

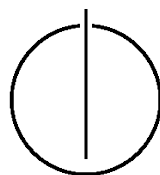
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

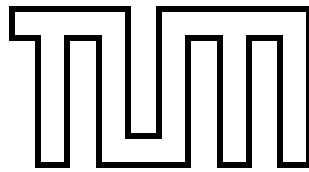
Habilitation in Informatik

**Approaches for Efficient and Autonomous  
Learning in Robotics and Computer Vision**

Dr. rer. nat. Rudolph Triebel







FAKULTÄT FÜR INFORMATIK

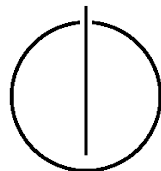
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Habilitation in Informatik

Approaches for Efficient and Autonomous Learning in  
Robotics and Computer Vision

Ansätze für Effizientes und Autonomes Lernen in  
Robotik und Computersehen

Author:	Dr. rer. nat. Rudolph Triebel
Supervisor:	Prof. Dr. Daniel Cremers
External Advisor:	Prof. Dr. Dieter Fox
Head of Committee:	Prof. Dr. Helmut Seidl
Date:	November, 2014





Ich versichere, dass ich diese Habilitationsschrift selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 12. November 2014

Dr. rer. nat. Rudolph Triebel



---

## Danksagung

An dieser Stelle möchte ich mich bei all den Menschen bedanken, die mich auf dem Weg während der Erstellung dieser Arbeit begleitet und unterstützt haben. Allen voran sind dies die Profs. Roland Siegwart (ETH Zürich), Paul Newman (Univ. Oxford) und Daniel Cremers (TU München), die es mir ermöglicht haben, in ihren jeweiligen Arbeitsgruppen zu forschen, und dabei interessante wissenschaftliche Fragestellungen mit konkreten Anwendung in Projekten auf international hohem Niveau zu verbinden. Ohne ihre Unterstützung wäre diese Arbeit nicht möglich gewesen. Auch möchte ich speziell Prof. Dieter Fox (Univ. Washington, Seattle) danken für die hilfreiche Zusammenarbeit und die guten Diskussionen während seines Aufenthaltes an der TU München. Ausserdem bedanke ich mich bei all meinen ehemaligen und aktuellen Kollegen aus allen drei Arbeitsgruppen, insbesondere aber bei Luciano Spinello, Jérôme Maye, Ralf Kästner, Jiwon Shin, Rohan Paul, Hugo Grimmett, Ingmar Posner, Jan Stühmer, Mohamed Souiai, Shoubhik Debnath und Shiv Sankar Baishya. Die Zusammenarbeit mit ihnen hat mir sehr viel Freude gemacht, und ich habe einige Erkenntnisse in dieser Arbeit erst durch diese Zusammenarbeit erlangt. Ein weiteres herzliches Dankeschön geht an Herrn Prof. Arras (Uni Freiburg) für die gute Zusammenarbeit im EU-Projekt SPENCER und für viele interessante und nicht nur fachliche Gespräche. Des weiteren bedanke ich mich bei meinen Eltern für die hilfreiche, tatkräftige Unterstützung während der Erstellung dieser Arbeit. Schließlich jedoch gilt der allergrößte Dank meiner Frau Rocio und meinen Söhnen Romeo und Ernesto, welche mich während der gesamten Zeit unterstützt haben und viel Geduld und Verständnis aufgebracht haben. Danke!

Meiner Frau Rocio und meinen Söhnen Romeo und Ernesto gewidmet.





---

## Abstract

This thesis investigates machine learning approaches for perception tasks in robotics and computer vision applications. The main focus of the developed techniques are the aspects of efficiency and autonomy, where the latter refers to the amount of interaction with a human user that is required to perform the learning task. Starting from empirical evaluations of standard supervised offline learning methods for classification in robot perception tasks, we extend and modify these approaches to unsupervised and online learning techniques. This reduces the number of user interactions in terms of queries for ground truth class labels and increases the efficiency with respect to computational resources, as we show experimentally. Furthermore, it enables the learning algorithms to adapt to new, unseen situations, which is a key requirement for systems that are designed to learn continuously, such as life-long learning robot systems.

Beyond that, we especially investigate supervised classification techniques that are able to provide reliable estimates of confidence in addition to the class label predictions they generate. We argue that classifiers which tend to make false predictions with a high confidence can cause severe effects, even if they generally perform well in classical measures such as precision and recall. In our analysis of such methods based on a notion of overconfidence, we conclude that the use of less overconfident methods such as the Gaussian Process Classifier (GPC) should be preferred over standard methods such as Support Vector Machines (SVM) if the confidence estimates are critical for further processing. Concretely, we demonstrate these effects for the task of Active Learning, where the learning algorithm itself chooses new training data to learn from. In that setting, an overconfident classifier has the major disadvantage that it can not improve on the samples where it produced a wrong prediction, as is shown in experiments, because it has no possibility to detect such misclassifications. Finally, an alternative learning method based on Boosting is developed and shown to reduce overconfidence over standard methods, resulting in an efficient active online learning framework with significantly steeper learning curves compared to the state of the art.

The approaches and techniques developed in the thesis are all backed up and motivated by practical applications in robotics and computer vision. Concretely, we investigate the tasks of the detection of cars, pedestrians, and traffic lights in urban environments, as well as the classification of road signs and objects in 3D indoor and outdoor environments. Specifically, we show that the developed Active Learning techniques are very effective methods for semantic mapping, 3D object classification and interactive image segmentation.



---

## Zusammenfassung

Diese Arbeit befasst sich mit Ansätzen des maschinellen Lernens für Aufgaben der automatischen Wahrnehmung in Robotik und Computersehen. Der Hauptaugenmerk der entwickelten Techniken sind die Aspekte der Effizienz und der Autonomie, wobei sich letztere auf den Grad der Interaktion mit einem menschlichen Benutzer bezieht, welche nötig ist, um die Lernaufgabe zu erfüllen. Ausgehend von empirischen Auswertungen anhand von häufig verwendeten, überwachten offline Lernmethoden zur Lösung von Klassifikationsaufgaben in der Roboterwahrnehmung, erweitern und verändern wir diese Ansätze in Richtung unüberwachtes und online Lernen. Dies verringert die Anzahl von Benutzerinteraktionen bezüglich der Anfragen nach Klassenbezeichnern aus der Grundwahrheit und erhöht die Effizienz im Sinne der Rechenintensität, wie wir experimentell zeigen. Weiterhin ermöglicht es dem Lernalgorithmus sich an neue, noch nicht beobachtete Situationen anzupassen, was eine besondere Voraussetzung für Systeme ist, die für andauerndes Lernen konzipiert sind, wie zum Beispiel lebenslang lernende Robotersysteme.

Darüber hinaus untersuchen wir speziell überwachte Klassifikationsmethoden, die in der Lage sind, eine zuverlässige Abschätzung der Konfidenz zu liefern, zusätzlich zu den vorhergesagten Klassenbezeichnern, die sie erzeugen. Wir argumentieren, dass Klassifikatoren, die dazu neigen falsche Vorhersagen mit hoher Konfidenz zu machen, ernsthafte Effekte hervorrufen können, sogar wenn sie generell sehr leistungsfähig sind gemessen an klassischen Kriterien wie Genauigkeit und Trefferquote. In unserer Untersuchung von solchen Methoden basierend auf dem Begriff der Hyperkonfidenz schließen wir, dass die Verwendung von weniger hyperkonfidenten Methoden wie zum Beispiel dem Gauß-Prozess Klassifikator (GPC) der von üblicherweise eingesetzten Verfahren wie der Support-Vektor Maschine (SVM) vorgezogen werden sollte, wenn die Konfidenzabschätzung kritisch für die weitere Verarbeitung ist. Konkrete zeigen wir diese Effekte am Beispiel des Aktiven Lernens, bei dem der Lernalgorithmus selbst entscheidet, aus welchen Trainingsdaten er lernt. In diesem Szenario hat ein hyperkonfidenter Klassifikator den großen Nachteil, dass er sich nicht auf den Datenstichproben verbessern kann, für die er eine falsche Vorhersage getroffen hat, wie wir in Experimenten zeigen, da er keine Möglichkeit hat, diese Fehlklassifikationen zu erkennen. Schließlich entwickeln wir eine alternative Lernmethode basierend auf dem Boosting-Schema, von welcher gezeigt wird, dass sie die Hyperkonfidenz im Vergleich mit Standardmethoden reduziert und zu einem effizienten aktiven online Lernverfahren führt, welches signifikant steilere Lernkurven erzeugt verglichen mit aktuellen Methoden.

Die Ansätze und Techniken, die in dieser Arbeit entwickelt werden, sind alle unterstützt und motiviert durch praktische Anwendungen in der Robotik und dem Computersehen. Konkret behandeln wir Aufgaben der Erkennung von Autos, Fußgängern und Ampeln in

---

Stadtumgebungen, sowie die Klassifikation von Straßenschildern und dreidimensionalen Objekten im Innen- und Außenbereich. Speziell zeigen wir, dass die entwickelten Aktiven Lernmethoden sehr effektiv sind für semantische Kartenerstellung, 3D Objektklassifikation und interaktive Bildsegmentierung.

# Contents

<b>Danksagung</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Overview of this Thesis . . . . .	2
1.2. Main Contributions . . . . .	3
<b>I. Supervised Offline Learning</b>	<b>5</b>
<b>2. A Short Summary of Common Supervised Learning Methods</b>	<b>7</b>
2.1. Support Vector Machines . . . . .	7
2.1.1. Training an SVM . . . . .	7
2.1.2. Inference with an SVM . . . . .	8
2.1.3. Relevance in the Context of this Thesis . . . . .	9
2.2. Boosting . . . . .	9
2.2.1. AdaBoost Training . . . . .	9
2.2.2. AdaBoost Inference . . . . .	11
2.2.3. Relevance in the Context of this Thesis . . . . .	11
2.3. Conditional Random Fields . . . . .	12
2.3.1. Probabilistic Formulation . . . . .	12
2.3.2. Training . . . . .	14
2.3.3. Inference . . . . .	15
2.3.4. Relevance in the Context of this Thesis . . . . .	16
<b>3. Learning to Detect Cars and Pedestrians</b>	<b>17</b>
3.1. Multi-modal Detection of Cars and Pedestrians . . . . .	17
3.1.1. Appearance-Based Detector . . . . .	18
3.1.2. Range-Based Detector . . . . .	18
3.1.3. Results . . . . .	20
3.2. Improvements using Ground Plane Extraction . . . . .	21
3.2.1. Ground Plane Computation from 3D Range Data . . . . .	21
3.2.2. Results . . . . .	22
3.3. Pedestrian Detection at Small Scales . . . . .	22
3.3.1. Descriptors Designed for Small Objects . . . . .	23
3.3.2. Results . . . . .	24
3.4. Detecting Pedestrians in 3D Range Data . . . . .	24
3.4.1. Classification by Splitting into Parts . . . . .	24

3.4.2. Results . . . . .	25
<b>4. Confidence-aware Classification</b>	<b>27</b>
4.1. Motivation . . . . .	27
4.1.1. Obtaining Uncertainty from an SVM . . . . .	28
4.1.2. Under- and Overconfidence . . . . .	28
4.2. A Less Overconfident Classifier . . . . .	29
4.2.1. The Formulation of the Gaussian Process Classifier (GPC) . . . . .	29
4.2.2. The GPC for Multiple Classes . . . . .	31
4.2.3. Application to 3D Point Cloud Classification . . . . .	32
4.3. The Importance of Confidence in Mission-Critical Classification . . . . .	34
4.3.1. Traffic Light Detection and Road Sign Classification . . . . .	34
4.3.2. Results . . . . .	35
4.4. An Alternative to the GPC . . . . .	37
4.4.1. Online Multi-Class Gradient Boost (OMCGB) . . . . .	37
4.4.2. Confidence Boosting . . . . .	38
4.4.3. Results . . . . .	38
<b>II. Unsupervised and Online Learning</b>	<b>41</b>
<b>5. Unsupervised Offline Learning</b>	<b>43</b>
5.1. Learning from Repetition by Discovering Regular Patterns . . . . .	43
5.1.1. Individual Steps of the Algorithm . . . . .	43
5.1.2. Structure Learning and Reasoning Using the CRF . . . . .	44
5.1.3. Results . . . . .	45
5.2. Learning from Other Sources by Relating Image with Motion Data . . . . .	46
5.2.1. A Model for Direction Signs . . . . .	46
5.2.2. Sign Detection and Arrow Learning . . . . .	47
5.2.3. Quantitative Results . . . . .	48
5.3. Learning from Similarity and Physical Closeness . . . . .	49
5.3.1. Clustering by Hierarchical Probabilistic Inference . . . . .	49
5.3.2. The Scene Graph and the Parts Graph . . . . .	50
5.3.3. Results . . . . .	51
5.3.4. Object Categorization Across Different Scenes . . . . .	52
<b>6. Unsupervised Online Learning</b>	<b>53</b>
6.1. Self-supervised Online Learning of Driving Behaviours . . . . .	53
6.1.1. A Bayesian Formulation of the Problem . . . . .	54
6.1.2. The Traffic Situation Model and the Action Model . . . . .	54
6.1.3. Results . . . . .	56
6.1.4. Online Estimation of Dynamic Objects . . . . .	57
6.2. Unsupervised Online Segmentation of 3D Scenes . . . . .	57
6.2.1. Online Clustering in Feature Space . . . . .	57
6.2.2. Online CRF Inference with Incremental Belief Updates . . . . .	58
6.2.3. Results . . . . .	59

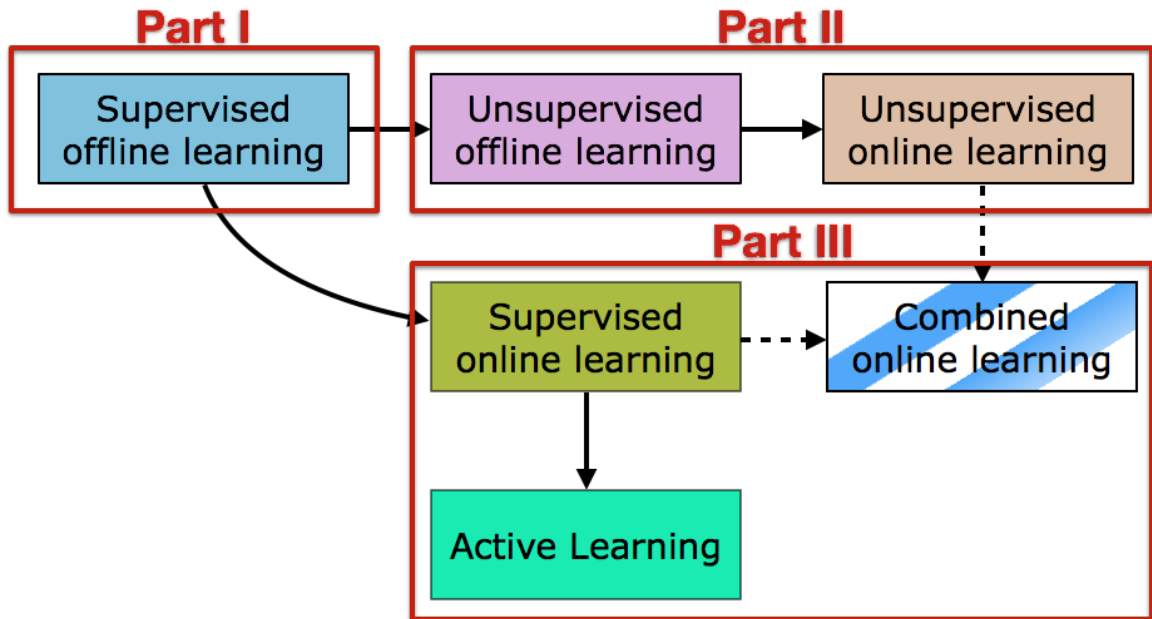
<b>III. Active Learning</b>	<b>61</b>
<b>7. Active Learning</b>	<b>63</b>
7.1. Combining Supervised and Unsupervised Learning . . . . .	63
7.1.1. From Clusters to Labels . . . . .	64
7.1.2. Results . . . . .	65
7.2. Active Learning for Semantic Mapping . . . . .	66
7.2.1. Active Learning in General . . . . .	66
7.2.2. Application to Semantic Mapping . . . . .	67
7.2.3. Results . . . . .	69
7.3. Active Learning for Interactive Image Segmentation . . . . .	70
7.3.1. Algorithm Overview . . . . .	70
7.3.2. Results . . . . .	72
7.4. Efficient Active Online Learning for 3D Object Classification . . . . .	73
<b>8. Conclusions</b>	<b>75</b>
<b>Bibliography</b>	<b>77</b>
<b>Publications</b>	<b>85</b>





# 1. Introduction

Machine learning algorithms have received an immensely growing interest in the areas of computer vision and robotics. Many problems in fields such as machine perception, robot navigation, control and others, which until recently seemed hard to solve, have been and are being addressed using techniques that were originally developed in the machine learning community. The success and the growing applicability and efficiency of these methods have convinced researchers and developers in computer vision and robotics of their benefit. However, a lot of challenges still remain. In particular, there are two objectives to be considered for machine learning techniques that are to be applied in a robotics or computer vision context: efficiency and autonomy. While the first objective is classically measured in terms of run time and memory requirements, the second is somewhat harder to measure. Also, compared to the goal of efficiency, the investigation of autonomous learning methods is a much more recent field, which makes questions such as comparability harder to address. Nevertheless, autonomy is a very attractive feature, in particular (although not exclusively) for machine *perception* tasks such as the ones we will investigate in this thesis. The reason for this is that in perception the aim is often to automatically associate an observation received from sensor input to some semantic interpretation, for example an object label (“chair”, “table”, etc.). However, the only reliable source of information, from which such semantics can be learned is a human supervisor, and providing such “ground truth” information to a learning algorithm is for most humans a tedious and annoying task. Therefore, in this thesis we will measure autonomy by the required amount of user interaction during the learning process. This means that in terms of our notion of autonomy the best learning algorithms are those that do not require any human user input, and we dedicate one significant part of the thesis to these unsupervised learning methods. We will show how certain features in the data such as regularity or similarity of constellations can be used to achieve segmentations that come very close to labelings that humans would do. Still, these methods can of course not provide any human description of the observations and should rather be considered as an attempt to group the data into meaningful clusters, so that a human annotation can be made on a higher, cluster-based level, and not per data sample. One example of such a method will be presented. Ultimately, however, our goal is to only reduce the amount of human interaction as much as possible while still retrieving enough information from the user to perform reliable predictions of class labels for newly observed data. This leads us to the field of *active learning*, which in a way combines the idea of autonomy with that of efficiency, because it requires less human input by querying selectively the data from which better learning can be expected, and it favours classification algorithms that learn fast and online. We will give concrete examples where active learning is used to achieve high classification rates with a comparably small amount of human data annotation.



**Figure 1.1.:** Overview of the learning methods developed in this thesis. Each colored box corresponds to a chapter, with the exception of supervised online learning and combined online learning. The former is mentioned here as a natural extension of standard supervised offline learning, but it is only applied within the context of active learning and not separately. The latter is generally considered to be too extensive to fit into the scope of this thesis. However, we mentioned it here as a motivation for our work, and we present some preliminary result on it.

## 1.1. Overview of this Thesis

To be able to develop the unsupervised and active learning techniques presented in this thesis, we first need to investigate and understand thoroughly the well-known standard supervised learning methods. There are at least two obvious reasons for that: first, by applying and refining the standard methods to concrete, relevant problems we obtain a notion of how well they perform for these tasks and where their drawbacks may be. And second, the development of novel learning algorithms will be easier and more reliable when we can rely on the established standard methods, for example by refining these rather than formulating new methods completely from scratch. Therefore, Part I of this thesis is mainly concerned with supervised learning methods such as Support Vector Machines, AdaBoost, and Conditional Random Fields, and how they can be used effectively to address the important problem of detecting cars and pedestrians from camera and laser range data in crowded urban environments. Then, in Chapter 4 we motivate the concept of *confidence-aware* supervised classification. This differs from the other learning techniques in that it also takes into account the uncertainty associated with a classification. Concretely, we investigate the use of a Gaussian Process classifier (GPC) to the problems of semantically segmenting 3D outdoor environments and for traffic light detection in urban traffic situations. As we will see, classification confidence can help to reduce false classifications, but more importantly, it will lay the foundation for an effective active learning approach later on.

In Part II, we then extend and modify some of the supervised methods so that they are either completely unsupervised or self-supervised, i.e. they relate data obtained from another source to the observations, so that predictions can be made for new observations. As an example for the former, we present in Chapter 5 a part-based segmentation and object discovery algorithm for 3D range data, which is based on the design of two inter-related undirected graphical models. With respect to self-supervised learning, we describe an approach to interpret the semantics of direction signs from previously recorded motion that is associated to input images containing the signs. Both ideas, the unsupervised and the self-supervised approach are then extended in Chapter 6 for online learning. This means, the algorithms are modified or redesigned so that they can make predictions *before* having observed the entire data set. Thus, they can be applied as filters, where new observations are used to incrementally refine an underlying model rather than completely recomputing it. This makes them much more efficient than the offline methods, which is – as mentioned – the other objective for this thesis apart from the autonomy.

Finally, in Part III we investigate active learning approaches and apply them to semantic mapping, interactive image segmentation, and 3D object recognition. Here, we make use of the findings from Chapter 4 and evaluate experimentally the benefits of confidence-aware classifiers such as the GPC over others that tend to be overconfident, i.e. they associate wrong classifications often with a high confidence. Here, in the context of active learning, overconfidence has the severe effect that it prevents the learning algorithm from improving its classification performance, because misclassified samples can often not be detected. Furthermore, we present a novel learning method that modifies a standard algorithm, namely GradientBoost, which also tends to overconfidence, in such a way that it is less overconfident. Again, this leads to a better performance in active learning, however with the additional advantage that the boosting method is much faster than the GPC.

## 1.2. Main Contributions

A total number of 20 scientific publications in robotics and computer vision has evolved from this thesis, and we refer to the Appendix on page 86 for a full list. Here, we name the most influential works from each chapter of this thesis:

- Detection of Pedestrians and Cars from Camera and Laser Range Data (Chapter 3):  
L. Spinello, R. Triebel, R. Siegwart: “Multiclass Multimodal Detection and Tracking in Urban Environments” in: The International Journal of Robotics Research (IJRR) 29 (12): 1498-1515, 2010 (see page 124)
- Confidence-Aware Classification of Outdoor Environments from 3D Laser Range Data (Chapter 4):  
R. Paul, R. Triebel, D. Rus, P. Newman: “Semantic Categorization of Outdoor Scenes with Uncertainty Estimates using Multi-Class Gaussian Process Classification” in: Proc. of the International Conference on Intelligent Robots and Systems (IROS) 2012 (see page 159)
- Unsupervised Segmentation and Object Discovery from 3D Point Clouds (Chapter 5):  
R. Triebel, J. Shin, R. Siegwart: “Segmentation and Unsupervised Part-based Discovery of Repetitive Objects” in: Robotics: Science and Systems (RSS) 2010 (see page 204)

- Unsupervised Online Learning for Segmentation of 3D Outdoor Environments (Chapter 6):  
R. Triebel, R. Paul, D. Rus, P. Newman: “Parsing Outdoor Scenes from Streamed 3D Laser Data Using Online Clustering and Incremental Belief Updates” in: Proc. of the Conference on Artificial Intelligence (AAAI) 2012 (see page 249)
- Active Learning for Semantic Mapping (Chapter 7):  
R. Triebel, H. Grimmett, R. Paul, I. Posner “Driven Learning for Driving: How Introspection Improves Semantic Mapping” in: Proc. of the International Symposium on Robotics Research (ISRR) 2013 (see page 257)

**Part I.**

# **Supervised Offline Learning**



## 2. A Short Summary of Common Supervised Learning Methods

In this chapter, we briefly review some of the most important supervised learning methods commonly used for classification tasks in robotics and computer vision. Throughout this thesis, these methods, along with some others, will be an important reference, be it for comparison with novel techniques or for application and adaptation to other contexts. In particular, we discuss Support Vector Machines (SVMs), Boosting methods, and Conditional Random Fields (CRFs). While the first two are typical classification methods, the latter is not usually denoted a classifier. Still, it requires labeled training data for parameter learning, and, if we consider node potentials as “local” classifiers, then CRFs can be used for *collective classification*. We will give more details below. Note that our description of these standard supervised learning methods is meant to be short, given that there is already a large literature available on these topics. Two very popular examples of comprehensive text books are those of Bishop [2006] and Murphy [2012], to which we refer for any further details on these topics.

### 2.1. Support Vector Machines

The most popular supervised learning method used for classification tasks in computer vision and robotics are Support Vector Machines (SVMs). This popularity mainly stems from their good performance in terms of classification rates and run time, but also from the availability of easy-to-use software packages. The literature available on SVMs is abundant, and a particularly detailed reference is the work of Schölkopf and Smola [2002]. Here, we only sketch the main principles of the SVM learning method, separated into the *training* phase, where a model is learned for a given training data set, and the *inference* phase, where a class label prediction is made based on a new test data point and the learned model.

#### 2.1.1. Training an SVM

Suppose we are given a set of  $N$  input training data points  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ , along with known ground truth labels  $y_1, \dots, y_N$ , where  $y_n \in \{-1, 1\}$ , i.e. we consider a binary classification problem. Our aim is to find model parameters  $(\mathbf{w}, b)$  so that the function

$$f(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \quad (2.1)$$

separates the two classes, i.e.  $f(\mathbf{x}_n) > 0$  if  $y_n = 1$  and  $f(\mathbf{x}_n) < 0$  if  $y_n = -1$ , where  $\phi$  is a *feature function* that maps the data into a feature space. This problem is equivalent to finding a hyper plane that separates all training data points in feature space. If such a hyperplane

exists, it may not be unique, and the idea is to find the hyperplane that *maximizes the margin*, i.e. the one that has the largest distance to any of the training data points. If there is no such hyperplane, then the problem can be relaxed so that the hyperplane at least separates a fraction of the training set correctly, but we will not consider this case here. The maximum-margin solution to Eq. (2.1) can be obtained from the following quadratic program formulation:

$$\arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b) \geq 1, \quad n = 1, \dots, N. \quad (2.2)$$

By introducing Lagrange multipliers  $a_n$  for each constraint and eliminating  $\mathbf{w}$  and  $b$  from the resulting Lagrangian, we obtain the *dual formulation* of the problem:

$$\begin{aligned} \arg \max_{\mathbf{a}} \left\{ \sum_{n=1}^N a_n - \frac{1}{2} \sum_{n=1}^N \sum_{m=1}^N a_n a_m y_n y_m k(\mathbf{x}_n, \mathbf{x}_m) \right\} \\ \text{s.t.} \quad a_n \geq 0, \quad n = 1, \dots, N \\ \sum_{n=1}^N a_n y_n = 0. \end{aligned} \quad (2.3)$$

Here, the function  $k(\mathbf{x}_n, \mathbf{x}_m) = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m)$  defines the *kernel function*. Thus training an SVM involves solving for the Lagrange multipliers  $\mathbf{a}$  in Eq. (2.3). This is again an instance of a quadratic program, for which there are several standard techniques in the literature. The most common one in the context of SVMs is the *sequential minimal optimization* (SMO) algorithm by Platt [1998]. Once a solution to (2.3) is found, it can be used to compute the threshold parameter  $b$  in (2.2).

The major benefit of the dual formulation (2.3) over Eq. 2.2 is that it is expressed only in terms of the kernel function  $k$ , and the feature function  $\phi$  has disappeared. This enables the applicability of the *kernel trick*, i.e.  $k$  can be replaced by any symmetric, positive-definite function (i.e. a so-called *Mercer kernel*), including cases where a corresponding feature function  $\phi$  can only be infinite-dimensional. The kernel trick makes the SVM a very powerful classifier, because it extends the notion of a linearly separating hyperplane into highly non-linear, and even infinite-dimensional feature spaces. In practice, very often the mostly used Gaussian kernel function is sufficient to obtain good classification results.

### 2.1.2. Inference with an SVM

After the model parameters  $(\mathbf{a}, b)$  are obtained from the training phase, class labels for new test data points  $\mathbf{x}^*$  are predicted in the inference phase. This is done by evaluating the decision function

$$f(\mathbf{x}^*) = \sum_{n=1}^N a_n y_n k(\mathbf{x}^*, \mathbf{x}_n) + b \quad (2.4)$$

and assigning a positive label to  $\mathbf{x}^*$  whenever  $f(\mathbf{x}^*) > 0$ . From a computational point of view, this means that for every new prediction step, all  $N$  training data points need to be



considered. Fortunately, it turns out that this is not necessary: It can be shown that for all training data points  $\mathbf{x}_n$  it either holds  $a_n = 0$  or  $y_n f(\mathbf{x}_n) = 1$ . In the former case, the corresponding term does not contribute to the sum in Eq. (2.4). Thus, the sum needs to be evaluated only for the latter case, i.e. when the corresponding training point lies on the boundaries of the margin. These points are called the *support vectors*. The fact that for prediction only the support vectors are needed makes the SVM classification method very efficient: usually the number of support vectors is much smaller than  $N$ , leading to a *sparse* formulation of the problem.

### 2.1.3. Relevance in the Context of this Thesis

Despite their broad use and their obvious benefits in terms of classification performance and run time efficiency, Support Vector Machines have at least one important drawback: they can not return a probabilistic prediction in terms of a class label probability for a new test point  $\mathbf{x}^*$ . However, this is often necessary, particularly if an estimate of uncertainty is required. Although there have been attempts to mitigate this problem, probabilistic estimates obtained from an SVM classifier are often not reliable enough for applications that require them. More details of this will be given in Chapters 4 and 7. Here, we summarize the two main statements regarding SVMs that will be backed up with experiments in later chapters of this thesis:

1. **SVMs yield very good classification rates.** As we will show in Chapter 3, SVMs are a very useful tool for important applications in mobile robotics, and we consider the classification of pedestrians and cars in an urban environment.
2. **SVMs are overconfident.** In situations where a probabilistic estimate of the classification is needed, SVMs perform poorly in the sense that they tend to return wrong class label predictions with low uncertainty. In Chapter 4, we will show examples of this. As a consequence, SVMs are not very useful in applications such as Active Learning, and we give experimental evidence of this in Chapter 7.

## 2.2. Boosting

Another very popular supervised learning framework for classification tasks are *boosting methods*, and in particular the AdaBoost algorithm by Freund and Schapire [1997]. This method received major attention in the computer vision community when it was very successfully employed to face detection by Viola and Jones [2002]. Since then, it has been used for many different problems in computer vision and robotics, and a large variety of extensions and improvements have been developed. A good overview and many detailed explanations are given by Schapire and Freund [2012]. Here, we will only give a summary on the standard AdaBoost algorithm. Again, we distinguish between the training and the inference phase of the learning algorithm.

### 2.2.1. AdaBoost Training

As above, we assume we are given a training data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  with ground truth labels  $\mathbf{t} = (y_1, \dots, y_N)$ , and again we consider only the binary case, i.e.  $y_n \in \{-1, 1\}$ . The

---

**Algorithm 1:** AdaBoost for binary classification

---

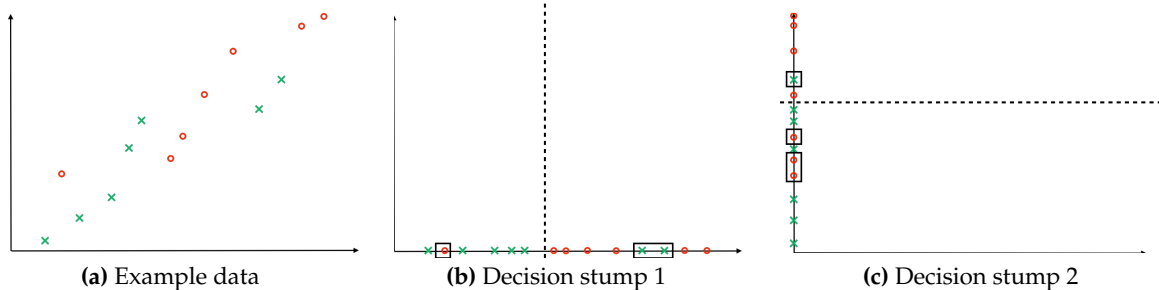
**Data:** training data  $(\mathcal{X}, \mathbf{y})$  consisting of  $N$  labeled feature vectors, where  
 $y_n \in \{-1, 1\}$   
**Input:** Number  $M$  of training rounds  
**Output:** weak classifiers  $(f_1, \dots, f_M)$ , coefficients  $(\alpha_1, \dots, \alpha_M)$

- 1  $\mathbf{w}^{(1)} \leftarrow (1/N, \dots, 1/N)$
- 2 **for**  $m \leftarrow 1$  **to**  $M$  **do**
- 3      $f_m \leftarrow \text{LearnWeakClassifier}(\mathbf{w}, \mathcal{X}, \mathbf{y})$
- 4      $\epsilon_m \leftarrow \frac{\sum_{n=1}^N w_n^{(m)} I(f_m(\mathbf{x}_n) \neq y_n)}{\sum_{n=1}^N w_n^{(m)}}$
- 5      $\alpha_m \leftarrow \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
- 6     **for**  $n \leftarrow 1$  **to**  $N$  **do**
- 7          $w_n^{(m+1)} \leftarrow w_n^{(m)} \exp(\alpha_m I(f_m(\mathbf{x}_n) \neq y_n))$
- 8     **end**
- 9 **end**

---

main idea of all boosting methods is to run a given number  $M$  of training rounds through the data, where in each training round a *weak classifier* is trained that particularly focusses on the misclassified samples from the previous rounds. To achieve this, *weights*  $w_1, \dots, w_N$  are assigned to the training data, and those  $w_i$  that correspond to misclassified training samples are increased after each training round. As a result, the next weak classifier to be trained will attach more importance to these misclassified samples. Important to note here is that a weak classifier is per definition only required to yield a training error that is better than a random “classifier” would give. This means for the two-class problem that we need to guarantee that the training error for any possible training set is always smaller than 0.5. A very simple and commonly used weak classifier is the *decision stump*, and Fig. 2.1 shows an illustration of this. Now, the main point of boosting algorithms is the fact that the overall training error decreases during training even though the underlying weak classifiers may be quite bad in performance (but at least better than random guessing). This behaviour justifies the name “Boosting Algorithms” for these kind of supervised learning methods.

A more explicit description of AdaBoost is given in Algorithm 1. In the first step (line 1) all training weights are equally initialized with  $1/N$ . Then, in each round a weak classifier  $f_m : \mathbb{R} \rightarrow \{-1, 1\}$  is learned from the weighted training data, and its weighted training error  $\epsilon_i$  is computed (line 4). Here,  $I()$  denotes the indicator function, which is 1 if the argument is true and 0 otherwise. From the training error  $\epsilon_m$  the coefficient  $\alpha_m$  is computed. This value is later used to weight the weak classifier  $f_m$  trained in round  $m$  within the ensemble. The intuition is that if  $f_m$  has a small training error,  $\alpha_m$  will be large and  $f_m$  will contribute more to the ensemble. Then, in the last steps, the data weights are updated so that the misclassified points obtain a higher weight while the weights of the other points remain unchanged (lines 6 – 8).



**Figure 2.1:** Decision stump as a weak classifier for AdaBoost. a) A simple 2D example data set with two classes. b) Projection of the data onto the first feature dimension. To learn a decision stump, we find the (hyper-) plane that separates the data into two classes with the smallest number of outliers. Here, we have three outliers (see boxes). These will receive a higher weight compared to the other data points in the next training round. c) Projection of the same data onto the second data dimension. Again, the dashed line shows the optimal hyper-plane separating the data. However, here the training error is higher. Therefore, AdaBoost selects the first decision stump from b) as a next weak classifier.

### 2.2.2. AdaBoost Inference

As soon as the weak classifiers  $f_1, \dots, f_M$  and their coefficients  $\alpha_1, \dots, \alpha_M$  are obtained from the training process, new test data points  $\mathbf{x}^*$  can be classified. To do this, AdaBoost evaluates the function

$$f(\mathbf{x}^*) = \sum_{m=1}^M \alpha_m f_m(\mathbf{x}^*) \quad (2.5)$$

and tests it for its sign. If the sign is positive, the predicted class label  $y^*$  is 1, otherwise it is  $-1$ . Usually, this operation can be done very quickly, which is also one reason why AdaBoost is often used in real-time applications.

### 2.2.3. Relevance in the Context of this Thesis

Boosting algorithms will be a topic of special relevance in later chapters of this thesis. However, in addition to the mere application and performance evaluation of the algorithms, we will present an extension that particularly aims at reducing the *overconfidence* of the classifier. This will be shown to be very useful in an active learning framework, where a reliable uncertainty estimate of the classifier is crucial. Thus, we will investigate these two statements regarding Boosting:

1. **Boosting is a very effective and flexible classification framework.** In Chapter 3 we give several examples of successful applications of AdaBoost to pedestrian and car detection, including a technique where AdaBoost was combined with an SVM.

2. **Boosting can be extended to reduce overconfidence.** Similar to SVMs, Boosting tends to be overconfident when used as a classifier with probabilistic output. We give experimental evidence of this in Chapter 4. However, in contrast to SVMs, Boosting can be modified such that it returns class label predictions with less overconfidence. This is denoted Confidence Boosting, and will be presented in Chapters 4 and 7.

## 2.3. Conditional Random Fields

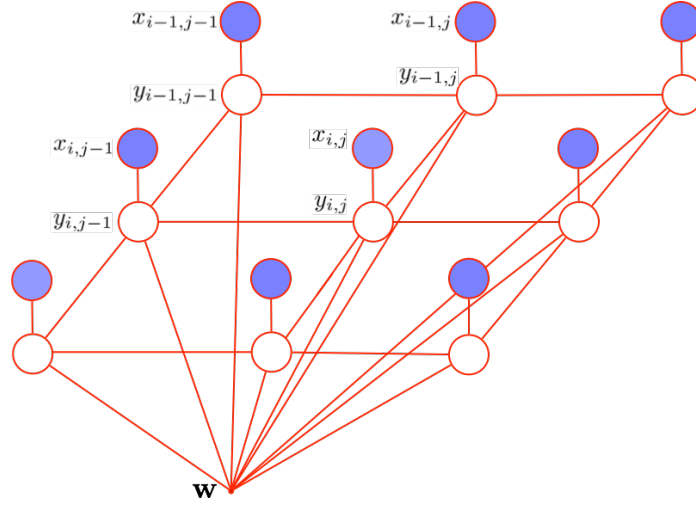
In contrast to SVMs and Boosting methods, Conditional Random Fields (CRFs) belong to the class of *probabilistic reasoning* methods. This means that the approach is formulated in terms of probability theory, and data points and class labels are modeled as random variables. As we will see later, this has major benefits when dealing with data that comes from noisy observations of a sensor such as a camera or a 2D laser scanner. Although the literature on CRFs is not as ample as it is for the other two classification methods in this chapter, there is a very detailed, recent article by Sutton and McCallum [2012], which we recommend for further reading on this topic. Again, we only present the principles of the method here to build the foundation for further references to the method within this thesis.

### 2.3.1. Probabilistic Formulation

As above, we start with a training data set which consists of observations  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  and corresponding ground truth labels  $\mathbf{y} = (y_1, \dots, y_N)$ . We note that there are at least two major differences to the formulation used above: First, all elements – except the model parameters  $\mathbf{w}$  – are modeled as collections of *random variables*, i.e. there is an underlying random distribution assumed for them. And second, the class labels  $y_n$  are elements of a set  $\{1, \dots, K\}$ ,  $K \in \mathbb{N}$  and not just 1 or  $-1$ . This allows classification into many classes instead of just two. With this notation, the conditional distribution modeled by a CRF can be formulated as:

$$p(\mathbf{y} \mid \mathcal{X}, \mathbf{w}) = \frac{1}{Z(\mathcal{X}, \mathbf{w})} \prod_{c \in \mathcal{C}} \varphi_c(\mathbf{y}_c \mid \mathcal{X}, \mathbf{w}), \quad (2.6)$$

where  $Z(\mathcal{X}, \mathbf{w}) = \sum_{\mathbf{y}'} \prod_{c \in \mathcal{C}} \varphi_c(\mathbf{y}_c \mid \mathcal{X}, \mathbf{w})$  is a normalizer, which turns the right hand side into a proper distribution. In the literature,  $Z$  is known as the *partition function*. The function  $\varphi_c$  is a non-negative *potential*, which is interpreted as an unnormalized probability. Furthermore, in Eq. (2.6), the symbol  $c$  denotes the index of a *clique*  $\mathbf{y}_c$  of random variables, i.e. a subset of  $\mathbf{y}$ , in which all pairs of variables directly depend on each other. The set of all cliques is denoted as  $\mathcal{C}$ . The concept of cliques becomes clearer when the CRF is represented as a *probabilistic graphical model*, in which each random variable corresponds to a node, and dependencies are modeled as edges (see Fig. 2.2 for an example). Here, a clique is determined as a fully connected subgraph. Usually, working with cliques of arbitrary size makes the formulation overly complex. Therefore, most often a maximum clique size of two is assumed, which leads to the so-called *pairwise* CRFs. The formulation of a pairwise CRF is somewhat simpler:



**Figure 2.2.:** Example of a CRF on a 2D grid. This model can be used for image segmentation in computer vision, where each pixel of an image is modelled as an observed feature  $x_{ij}$ , and the true label or segment identifier of a pixel is a hidden (unobserved) variable  $y_{ij}$ . According to the notation used by Bishop [2006], we denote observed variables with filled circles and hidden variables with open circles. The deterministic model parameters  $\mathbf{w}$  are depicted as a small point.

$$p(\mathbf{y} \mid \mathcal{X}, \mathbf{w}) = \frac{1}{Z(\mathcal{X}, \mathbf{w})} \prod_{n=1}^N \varphi(y_n \mid \mathbf{x}_n, \mathbf{w}) \prod_{(n,m) \in \mathcal{E}} \psi(y_n, y_m \mid \mathbf{x}_n, \mathbf{x}_m, \mathbf{w}), \quad (2.7)$$

where  $\mathcal{E}$  denotes the set of all edges in the graph, i.e. all pairs of indices of connected hidden nodes. This means, that there are *node potentials*  $\varphi$  and *edge potentials*  $\psi$ . Although these potentials are not specified further in the general setting, most authors use the so-called *log-linear* model. In this model the potentials are defined as

$$\varphi(y_n \mid \mathbf{x}_n, \mathbf{w}) = \exp(\mathbf{w}_\eta^T f_\eta(y_n, \mathbf{x}_n)), \quad n = 1, \dots, N \quad (2.8)$$

$$\psi(y_n, y_m \mid \mathbf{x}_n, \mathbf{x}_m, \mathbf{w}) = \exp(\mathbf{w}_\epsilon^T f_\epsilon(y_n, y_m, \mathbf{x}_n, \mathbf{x}_m)) \quad n, m = 1, \dots, N, \quad (2.9)$$

where the parameters  $\mathbf{w}$  are split into a vector of *node weights*  $\mathbf{w}_\eta$  and a vector of *edge weights*  $\mathbf{w}_\epsilon$ . Under this model, a CRF is then fully specified by a *node feature function*  $f_\eta$  and an *edge feature function*  $f_\epsilon$ . If the CRF is used for classification, as we do in this work, then we can think of the node feature function as the application of a classifier that only considers the local evidence, i.e. it is not influenced by the (labels of the) neighboring data points. If we denote such a classifier as  $\zeta$ , then we have

$$f_\eta(y_n, \mathbf{x}_n) \propto p(y_n = \zeta(\mathbf{x}_n) \mid \mathbf{x}_n). \quad (2.10)$$

The role of the edge feature function is to relate class labels of neighboring data points with each other. A commonly used edge feature function is defined as

$$f_\epsilon(y_n, y_m, \mathbf{x}_n, \mathbf{x}_m) = \begin{cases} c & \text{if } y_n = y_m \\ 0 & \text{otherwise} \end{cases}, \quad \text{where } c > 0. \quad (2.11)$$

This is often referred to as the generalized Potts model (see Potts [1952]).

### 2.3.2. Training

In the training phase, we are given a set of ground truth labels  $(y_1, \dots, y_N)$  with the training data, and our aim is to find node and edge weights  $\mathbf{w}_\eta$  and  $\mathbf{w}_\epsilon$  so that the data is best explained by the model. This is usually done using a maximum-likelihood approach, where the likelihood  $p(\mathbf{y} \mid \mathcal{X}, \mathbf{w})$  is maximized by  $\mathbf{w}$ . To simplify the notation, we actually maximize the log of the likelihood, which is given by

$$\log p(\mathbf{y} \mid \mathcal{X}, \mathbf{w}) = \sum_{n=1}^N \mathbf{w}_\eta^T f_\eta(y_n, \mathbf{x}_n) + \sum_{(n,m) \in \mathcal{E}} \mathbf{w}_\epsilon^T f_\epsilon(y_n, y_m, \mathbf{x}_n, \mathbf{x}_m) - \log Z(\mathcal{X}, \mathbf{w}). \quad (2.12)$$

To maximize this we first need to compute the derivative with respect to the node and edge weights. If we do this for the last term in (2.12), we obtain

$$\frac{\partial \log Z(\mathcal{X}, \mathbf{w})}{\partial \mathbf{w}_\eta} = Z^{-1}(\mathcal{X}, \mathbf{w}) \sum_{\mathbf{y}'} \phi'(\mathbf{y}' \mid \mathbf{x}, \mathbf{w}_\eta) \prod_{(n,m) \in \mathcal{E}} \psi(y'_n, y'_m \mid \mathbf{x}_n, \mathbf{x}_m, \mathbf{w}_\epsilon), \quad (2.13)$$

where the first term is the partition function, the second term is the derivative of all node potentials with respect to  $\mathbf{w}_\eta$ , and the last term represents the edge potentials, which remain unchanged by the partial derivative. To compute  $\phi'$  we need to apply the product rule, which results in

$$\phi'(\mathbf{y}' \mid \mathbf{x}, \mathbf{w}_\eta) = \prod_{n=1}^N \exp(\mathbf{w}_\eta^T f_\eta(y'_n, \mathbf{x}_n)) \sum_{n=1}^N f_\eta(y'_n, \mathbf{x}_n). \quad (2.14)$$

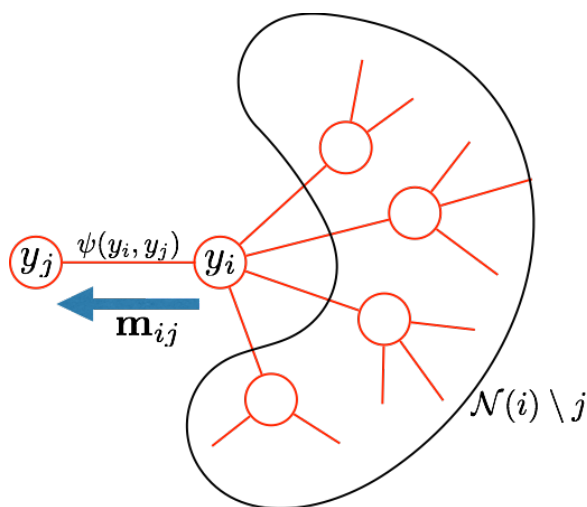
Pluggin this back into (2.13) gives

$$\frac{\partial \log Z(\mathcal{X}, \mathbf{w})}{\partial \mathbf{w}_\eta} = \sum_{\mathbf{y}'} \sum_{n=1}^N f_\eta(y'_n, \mathbf{x}_n) p(\mathbf{y}' \mid \mathcal{X}, \mathbf{w}), \quad (2.15)$$

where we have used our definition of the pairwise CRF from Eq. (2.6). This means that, in order to compute the gradient of the log-likelihood (2.12) we need to compute  $p(\mathbf{y}' \mid \mathcal{X}, \mathbf{w})$ , and this is done in the inference step, i.e. training *includes* inference. However, besides running an inference step, in training we also need to compute the partition function  $Z$ , but this is intractable in general. Therefore, we need to employ an approximation. One common way to approximate the likelihood (2.12) is by means of the *pseudo-likelihood*  $\ell_{PL}$ , which is defined as

$$\ell_{PL} = \prod_{n=1}^N p(y_n \mid \mathbf{y}_{\mathcal{N}(n)}, \mathcal{X}), \quad (2.16)$$

where  $\mathcal{N}(n)$  are all neighbors of node  $n$ . If we denote  $\mathbf{y}_{-n}$  as the vector of all labels with the exception of  $y_n$ , then we can say that the pseudo-likelihood is the product of all local likelihoods  $p(y_n \mid \mathbf{y}_{-n}, \mathcal{X})$ . To see this we note that, according to the structure of the CRF,  $y_n$  is conditionally independent of all other nodes given the *Markov blanket*  $\mathbf{y}_{\mathcal{N}(n)}$ . Thus,



**Figure 2.3.:** Message passing in the Belief Propagation algorithm. A message from node  $j$  to node  $i$  is the marginal of the subgraph at node  $i$ . It can be computed recursively from the potential at node  $i$ , the edge potential  $\psi_{ij}$ , and all messages sent from neighbors of node  $i$  except node  $j$ .

maximizing the pseudo-likelihood can be considered as matching the local likelihoods to the training data. Both the pseudo-likelihood and its derivative with respect to  $\mathbf{w}$  can be computed very efficiently, because they only involve the node potential of node  $y_n$  and the edge potentials of all adjacent edges. Normalization is done by summing over all possible labelings of node  $y_n$ . To optimize for  $\mathbf{w}$ , standard gradient descent methods can be used such as the L-BFGS method (see Liu and Nocedal [1989]).

### 2.3.3. Inference

As mentioned before, the inference step computes the conditional distribution  $p(\mathbf{y} \mid \mathcal{X}, \mathbf{w})$  for given (learned) model parameters  $\mathbf{w}$ . Depending on the application, there are two different variants of inference algorithms. If we are interested in the distribution itself, then we need to compute all node and edge marginals  $p(y_n \mid \mathbf{x}_n)$  and  $p(y_n, y_m \mid \mathbf{x}_n, \mathbf{x}_m)$ . In contrast, if we want to find the most likely labeling  $\mathbf{y}^*$  for a test data set  $\mathcal{X}^*$ , and we obtained  $\mathbf{w}$  already from training, then we need to compute  $\mathbf{y}^* = \arg \max_{\mathbf{y}} p(\mathbf{y} \mid \mathcal{X}^*, \mathbf{w})$ . It turns out that both inference problems are very closely related, and the algorithms used to compute them only differ in one operation. Therefore, we only present the algorithm for the maximization problem, as it is more relevant for our purpose.

If we assume for a moment that the structure of the CRF is a tree, i.e. there are no loops in the graph, then we can compute the node marginals  $p(y_n \mid \mathbf{x}_n)$  from the marginals that only correspond to each subgraph connected to  $y_n$ . These subgraph marginals are often interpreted as *messages* from the subgraph to node  $y_n$ . To simplify the notation, we will use the symbol  $\mathbf{m}_{ij}$  for the message that is sent to node  $y_j$  from the subgraph that starts at node  $y_i$ , which is connected to  $y_j$ . A visualization is shown in Fig. 2.3. This means that  $\mathbf{m}_{ij}$  is a distribution over all possible labels  $l_j$  at node  $y_j$ , and the probability  $m_{ij}(l_j)$  of each such label can be computed recursively using the update rule

$$m_{ij}(l_j) \leftarrow \sum_{l_i} \varphi_i \psi_{ij} \prod_{k \in \mathcal{N}(i) \setminus j} m_{ki}(l_i), \quad (2.17)$$

where we used the short hand notation  $\varphi_i = \varphi(y_i = l_i)$  and  $\psi_{ij} = \psi(y_i = l_i, y_j = l_j)$ . Equation (2.17) is usually called the *sum-product* rule. However, as mentioned we are only interested in the most likely label, therefore it is sufficient to turn the summation in Equation (2.17) into a maximization. Also, for numerical reasons it is much better to perform the computation in log-space. Therefore, the resulting update rule is

$$m_{ij}(l_j) \leftarrow \max_{l_i} \left\{ \log \varphi_i + \log \psi_{ij} + \sum_{k \in \mathcal{N}(i) \setminus j} m_{ki}(l_i) \right\}. \quad (2.18)$$

This is called the *max-sum* rule. For trees, the recursion always terminates and the messages are computed exactly. If there are loops in the graph, there are no guarantees that the algorithm converges, however, in practice convergence is mostly reached. The resulting inference algorithm is called *loopy belief propagation*. From the converged messages, the most likely label  $l_j^*$  at node  $y_j$  can then be computed as

$$l_j^* \leftarrow \arg \max_{l_j} \left\{ \nu \left( \log \varphi_j \sum_{i \in \mathcal{N}(j)} m_{ij}(l_j) \right) \right\}, \quad (2.19)$$

where  $\nu$  is the normalizer that ensures that the node marginal sums up to 1.

### 2.3.4. Relevance in the Context of this Thesis

Later in this thesis we will apply CRFs to the problem of *collective classification*, where a class prediction is made not only based on local characteristics of a data point, but also on its relationship to neighboring data points. Here, “neighboring” can refer to closeness in physical space, or in some feature space, where it can be interpreted as a similarity between data samples. This will prove useful when classifying objects in 2D and 3D laser range data. However, we will also show how this supervised, offline classification framework can be extended to unsupervised and online learning. In summary, we make the following two statements:

1. **CRFs are very useful for collective classification.** This will be shown in chapter 3, where we use a CRF to classify pedestrians and cars in 2D laser range scans. Here, the information obtained from the neighboring data samples is used to obtain smoother class label assignments.
2. **CRFs can be extended for unsupervised and online learning.** In chapter 6, we will present an algorithm for online, unsupervised segmentation of 3D point clouds. The method updates its internal representation incrementally with every new observed point cloud. It uses a combination of an online clustering algorithm to find similarities between 3D mesh segments, and a CRF that can grow incrementally, i.e. nodes and edges are added subsequently, and the inference step only considers those messages that are directly related to the newly added nodes and edges.



## 3. Learning to Detect Cars and Pedestrians

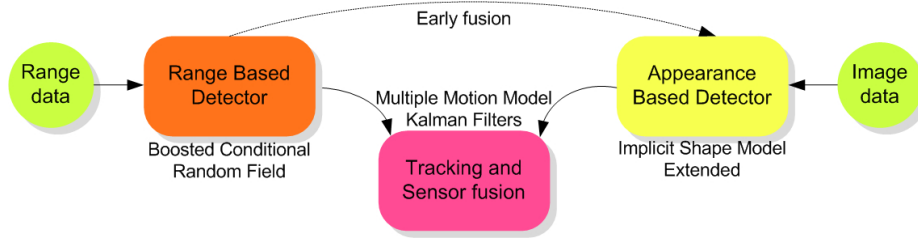
In this chapter, we present concrete applications of the supervised learning methods mentioned in chapter 2 to the concrete problem of detecting cars and pedestrians in an urban environment<sup>1</sup>. This is a very important task in mobile robotics, and in particular in the context of autonomous driving, which is one of the most studied topics currently in mobile robotics. For any autonomous or semi-autonomous driving system, the reliable detection of vehicles, pedestrians and moving objects from the surrounding traffic is crucial. Therefore, we investigate supervised learning techniques that are capable to robustly classify input data from different sensor modalities into classes such as “pedestrian” or “car”. Note that the purpose here is mainly to explore and apply existing learning techniques that use standard, offline supervised learning for a concrete and highly relevant problem, rather than investigating new learning methods. The latter will be the topic of later chapters, but the methods investigated here will serve as important foundations for these new techniques. In total, six scientific publications have evolved from these investigations, and we give a summary of them here. Further details can be found in the original works in the appendix (pages 88 ff.).

### 3.1. Multi-modal Detection of Cars and Pedestrians

The detection of cars and pedestrians is a very challenging task, particularly because it requires very reliable and informative input data. A single sensor modality is often not enough to obtain the required robustness for this task. For example, a single standard RGB camera can provide very detailed information about the environment at a comparably high resolution. However, the range at which moving objects can be detected is often not enough, especially when the camera itself moves at high speed. 2D laser range scanners have proved very useful here, because they yield measurements at very large distances and with a very wide angular field of view. However, their resolution is significantly lower compared to the camera, and important information such as color and texture are not available. Therefore, we consider a hybrid system that consists of these two modalities, i.e. a 2D laser scanner and a camera. This has the advantage that, given a good extrinsic calibration between these sensors, we can obtain accurate depth information and corresponding color information. We use this in our multi-modal detection framework (see Spinello et al. [2010b] on page 124), which is shown schematically in Fig. 3.1. In the following, we give more details about the appearance-based detection method, i.e. the one that operates on the camera images, and we present the range based detector.

---

<sup>1</sup>Note that in the publications on this topic (see appendix) we also addressed the tracking problem. However, this is of little relevance in the learning context, which is the major focus of this thesis.



**Figure 3.1.:** Schematic overview of the multimodal detection and tracking framework. We obtain data streams from two different sensors, namely the laser range scanner and the camera. For the range data, we apply a boosted Conditional Random Field, which was trained on a hand-labeled training set beforehand. The obtained detections additionally serve as regions of interest for the image-based detector by mapping them into the camera frame (“early fusion”). There, we use an improved version of the Implicit Shape Model (ISM) to detect pedestrians and cars, where the ISM was also trained on a labeled set of camera images. The detection results from both modalities are then fused in the tracking module.

#### 3.1.1. Appearance-Based Detector

Our vision-based people detector is mostly inspired by the work of Leibe et al. [2005] on scale-invariant Implicit Shape Models (ISM). In summary, an ISM consists in a set  $\mathcal{I}$  of local region descriptors called *codebook*, and a set  $\mathcal{V}$  of displacements and scale factors, usually named *votes*, for each descriptor. The idea is that each descriptor can be found at different positions inside an object and at different scales. Thus, a vote points from the position of the descriptor to the center of the object as it was found in the training data. To obtain an ISM from labeled training data, the descriptors are computed at interest point positions and then clustered, usually using agglomerative clustering with a maximal distance threshold  $\vartheta_d$ . Then, the votes are obtained by computing the scale and the displacement of the objects’ center to the descriptors. A training dataset consists in a collection of images and binary image masks defining the area and the position of the objects in each image. For the detection, new descriptors are computed on a test image and matched against the descriptors in the codebook. The votes that are cast by each matched descriptor are collected in a 3D *voting space*, and a maximum density estimator is used to find the most likely position and scale of an object.

In the publications relevant to this chapter (see appendix, page 88 ff.), several improvements to the ISM have been developed. However, for the development of unsupervised and online learning techniques presented later in this thesis, these methods are of little relevance. Therefore, we will at this point not focus further on them and refer the interested reader instead to the mentioned publications.

#### 3.1.2. Range-Based Detector

To detect cars and pedestrians from 2D laser range data (see Fig. 3.2-left), we propose a supervised learning method based on a Conditional Random Field (CRF) (see Lafferty et al. [2001]). The motivation of this is the fact that CRFs can handle dependencies between labels of neighboring data segments (see Section 2.3), which leads to a smoother and more natural assignment of labels. To formalize the problem, we use a discrete label  $l_i$  that



**Figure 3.2.:** An urban environment with cars, pedestrian and other objects as it is perceived by a 2D laser. **Left:** Laser beams are shown in red, circles represents the measured points. Gray beams indicate out of range data due to material reflections, sun related effects and particular object poses. **Center:** Resulting Jump-Distance Clustering of the scene. Orange lines depicts consecutive points segmented in the same cluster. **Right:** A Delaunay triangulation is computed on the centroids of the segments. This defines graph structure used for the CRF.

ranges over the 3 different classes “pedestrian”, “car” and “background”, as well as a feature vector  $\mathbf{s}_i$  that is extracted from a 2D data segment  $\mathcal{S}_i$  of the laser scan. We create these segments in a preprocessing step, which consists of a clustering technique to group nearby points, called Jump Distance Clustering (JDC). It is fast and simple to implement: if the Euclidean distance between two adjacent data points exceeds a given threshold, a new cluster is generated otherwise the point is added to the current cluster (see Fig. 3.2-center). Each cluster, or segment, is defined as the set of points  $\mathcal{S}_i$ . Then we compute a Delaunay triangulation between the centroids of each segment  $\mathcal{S}_i$  to obtain a graph that connects clusters (see Fig. 3.2-right). With this notation the conditional probability of the labels  $\mathbf{l}$  given the observations  $\mathbf{s}$  for a pairwise CRF (see Eq (2.7)) is

$$p(\mathbf{l} | \mathbf{s}) = \frac{1}{Z(\mathbf{s})} \prod_{i=1}^N \varphi(\mathbf{s}_i, l_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j), \quad (3.1)$$

As mentioned in Section 2.3, we use the log-linear model for the potentials and minimize the pseudo-likelihood to find the node and edge weights from training data. We use a set of statistical and geometrical features  $\mathbf{s}_i$  such as width, circularity, standard deviation, kurtosis, etc. (for a full list see Spinello and Siegwart [2008]). However, the features  $\mathbf{s}_i$  are not used directly within the node and edge feature functions  $\mathbf{f}_n$  and  $\mathbf{f}_e$ . Instead, we use AdaBoost to account for non-linear relations between observations and labels (see also the work of Ramos et al. [2007]). For our particular classification problem with multiple classes, we train one binary AdaBoost classifier for each class against the others. As a result, we obtain for each class  $c$  a set of  $M$  weak classifiers  $h_i^c$  (in our case decision stumps) and corresponding weight coefficients  $\alpha_i^c$  so that the sum

$$g_c(\mathbf{s}_i) := \sum_{i=1}^M \alpha_i^c h_i^c(\mathbf{s}_i) \quad (3.2)$$

is positive for observations assigned with the class label  $c$  and negative otherwise. To obtain a classification *likelihood*, we apply the logistic function  $\sigma(x) = (1 + e^{-x})^{-1}$  to  $g_c$ . This results in values between 0 and 1, which can be interpreted as the likelihood of correspondence to class  $c$ . We apply the same technique also for the edge features, i.e. the resulting potentials are comparable. Thus, the node feature function  $\mathbf{f}_n$  of the segment features  $\mathbf{s}_i$  and the label  $l_i$  is computed as

$$\mathbf{f}_n(\mathbf{s}_i, l_i) = \sigma(g_{l_i}(\mathbf{s}_i)). \quad (3.3)$$

For the edge features  $\mathbf{f}_e$  we compute two values, namely the Euclidean distance between the centroids  $\mathbf{c}_i$  and  $\mathbf{c}_j$  of the segments  $\mathcal{S}_i$  and  $\mathcal{S}_j$ , along with a value  $g_{ij}$  defined as

$$g_{ij}(\mathbf{s}_i, \mathbf{s}_j) = \text{sign}(g_i(\mathbf{s}_i)g_j(\mathbf{s}_j)) \cdot (|g_i(\mathbf{s}_i)| + |g_j(\mathbf{s}_j)|). \quad (3.4)$$

Thus, the value of  $g_{ij}$  has a positive sign if AdaBoost classifies  $\mathbf{s}_i$  and  $\mathbf{s}_j$  into the same class, and otherwise it is negative. The absolute value of  $g_{ij}$  is the sum of the classification qualities of AdaBoost. If  $g_i(\mathbf{s}_i)$  and  $g_j(\mathbf{s}_j)$  are far from 0 then  $g_{ij}$  has a high value, and viceversa. To summarize, we define the edge features as

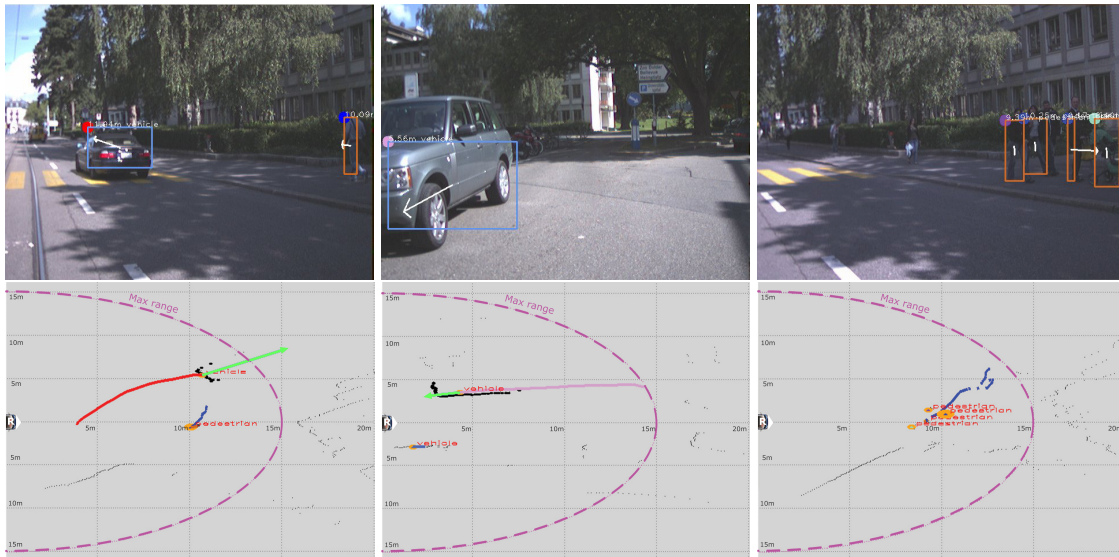
$$\mathbf{f}_e(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j) = \begin{cases} (\sigma(\|\mathbf{c}_i - \mathbf{c}_j\|) \quad \sigma(g_{i,j}(\mathbf{s}_i, \mathbf{s}_j)))^T & \text{if } l_i = l_j \\ (0 \quad 0)^T & \text{otherwise.} \end{cases} \quad (3.5)$$

The intuition behind equation (3.5) is that edges that connect segments with equal labels have a non-zero feature value and thus yield a higher potential.

### 3.1.3. Results

We evaluated our technique on a challenging urban scenario dataset collected in Zurich, Switzerland. It consists in a loop of about 1km length with cars and pedestrians in a busy urban environment. To obtain training data, the images were manually labeled with boxes indicating pedestrians and cars. Annotations in images are marked if at least half of an object is shown or the object width in the image is greater than 80 pixels. The laser range data was manually labeled using associated image frames as reference for the ground truth.

Fig. 3.3 shows an example for some qualitative results of our multi-modal pedestrian and car detector. Note that despite the challenging data due to low contrast, partial occlusions and clutter, the system manages to detect and track the objects in the scene. A quantitative analysis showed that the classification performance in terms of precision and recall is better than standard methods such as the one of Dalal and Triggs [2005], which is based on histogram of oriented gradients (HOG) computed on the camera images, or the AdaBoost classifier of Arras et al. [2007] for the laser data. The equal error rate (EER), i.e. the point where precision equals recall, is reached at 60.01% for our image-based pedestrian detector, while HOG reaches 52.21%. For the class “car” we obtained 72.54% EER. The laser-based classifier using our boosted CRF resulted in 64.23% EER on the pedestrian class, while standard AdaBoost yielded 57.09%. When applied to the class “car”, we obtained 74.89% compared to 70.58% with standard AdaBoost. More detailed analyses of our method can be found in Spinello et al. [2010b] on page 124.



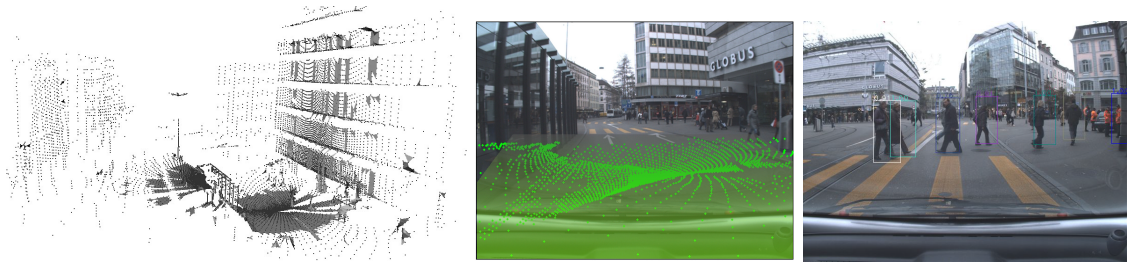
**Figure 3.3.:** Cars and pedestrian detected and tracked under occlusion, clutter and partial views. In the camera images (upper row), blue boxes indicate car detections and orange boxes pedestrian detections. The colored circles on the upper left corner of each box is the track identifier. Tracks are shown in color in the lower row and plotted with respect to the robot reference frame. Green vectors show the direction of motion for cars.

## 3.2. Improvements using Ground Plane Extraction

One problem with the ISM-based detection algorithm, which we use on the camera images, is that it tends to produce a large amount of false positive detections. Therefore, it is necessary to find appropriate criteria to discard candidates that are unlikely to correspond to foreground objects (i.e. cars or pedestrians). In the previous section, we used an “early fusion” step, in which the detection result from the 2D laser scanner is used to create a region of interest in the camera image. Concretely, the 2D laser segments that correspond to foreground are mapped into the camera frame using a previously performed extrinsic calibration between both sensor modalities. Then, a 3D region of interest (ROI) is created, where the width is defined by the segment width, and height and depth are chosen according to the object class (see Spinello et al. [2010b]). This turned out to reduce the number of false positives substantially. However, the vertical position of the ROI is often not very accurate due to the residual error in the calibration and the fact that the ground plane is slightly inclined in the presence of small slopes. This is particularly evident at far distances, which makes the detection of pedestrians there more challenging. Therefore, in Spinello et al. [2008] on page 94 we compute an estimate of the ground plane for alignment of the ROIs as described next.

### 3.2.1. Ground Plane Computation from 3D Range Data

To obtain a better estimate of the 3D ROIs, we use a third sensor modality, namely a 3D scanning device that produces point clouds from a laser scanner which rotates around the vertical axis. The advantage of this sensor is that it produces highly accurate 3D measure-



**Figure 3.4.:** Pedestrian detection with explicit ground plane estimation. **Left:** 3D point cloud of the environment, where the extracted ground plane is highlighted in a mesh representation. **Center:** 3D point cloud mapped into the camera frame. **Right:** Example result of pedestrian detection using ground plane estimation.

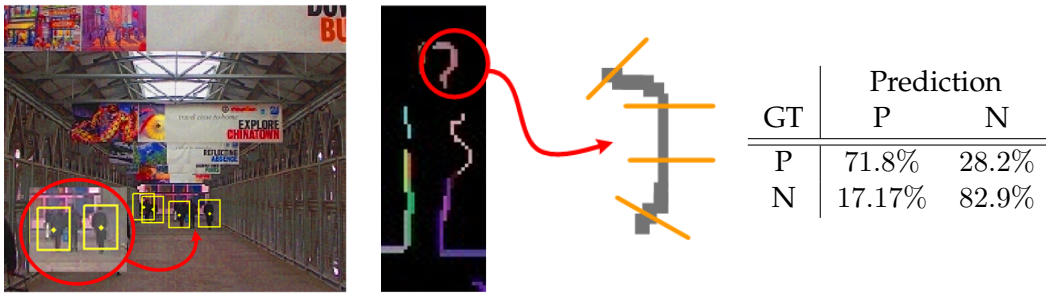
ments, which we exploit to obtain good estimates of the ground plane. An example of a point cloud produced by the 3D scanner is shown in Fig. 3.4 (left). To extract the ground plane from that data, we are mainly interested in a fast algorithm, because our application has to deal with a stream of input data in a quickly changing environment. Therefore, we decided to use a simple but time efficient region growing technique to detect the ground plane. The criterion we use for a scan point to belong to the ground plane is that its corresponding normal vector deviates only slightly (in our implementation by maximal  $25^\circ$ ) from the upright vector  $(0\ 0\ 1)^T$  and that it is not farther away from its closest neighbor than a given threshold (we use 1 m). The region growing is initiated always at the same fixed point right in front of the vehicle at the ground level. To efficiently compute the normal vectors, we exploit the fact that the point clouds are structured in slices, i.e. each scan line of the vertically mounted rotating laser scanner accounts for one slice. This facilitates a fast and simple mesh triangulation performed by connecting two consecutive points from one slice with one point of the consecutive slice. From this triangulation the normal vectors are easily computed using the normalized cross product of difference vectors. An example result of the ground plane extraction is shown in Figure 3.4 (center). This estimated plane is then used to align the 3D ROIs obtained from the 2D laser data vertically with the ground.

#### 3.2.2. Results

We evaluated the pedestrian detector with ground plane estimation on a similar data set as used in the previous section. An example result is shown in Fig. 3.4 (right). As can be seen, the detection result is very accurate even for pedestrians that are comparably far away. We observed that also in general for this data set. Quantitatively, we found that the multi-modal detection was about 10 – 20% better than detection using 2D laser alone. For further details on the experimental evaluation we refer to Spinello et al. [2008] (page 94).

### 3.3. Pedestrian Detection at Small Scales

Despite the improvements obtained from the ground plane estimation, the pedestrian detection method described so far still requires a certain minimal size at which the pedestrians have to be visible in the data. Thus, pedestrians that are far away, as well as children,



**Figure 3.5.:** Detecting pedestrians at very small scales. **Left:** Pedestrians in a walk way that are too far away for standard detection methods. The yellow boxes show the detection results from our algorithm. The yellow dots are the estimated object centers. **Center:** Graphical explanation of our descriptor. From a given set of edge segments (here colored inside the black box) we compute a histogram of gradient orientations (here shown in orange) on all edge pixels. **Right:** Confusion matrix of our detector for pedestrians at the size of  $16 \times 20$  pixels, evaluated on the NICTA data set provided by Overett et al. [2008]. Here, GT stands for “ground truth”.

can not be detected reliably. According to the rule of thumb from theoretical traffic lessons, a car that moves with 50km per hour needs 40m to come to a full stop. This is still far from the maximal distance at which pedestrians can be detected, using a lens that provides still an acceptable opening angle (above 90 degrees). Therefore, we investigate an approach to detect pedestrians that are up to 50m away while the lens still provides a wide field of view (see, e.g Fig. 3.5, left). To do this, we use descriptors that are particularly suited for objects in small scales, as described next.

### 3.3.1. Descriptors Designed for Small Objects

We designed two different descriptors that are particularly suited for the detection of objects at small scales. The first one is inspired by the HOG descriptor from Dalal and Triggs [2005], however applied to *edge segments*, i.e. consistent chunks of object edges (see Spinello et al. [2009] for more details). Along each edge segment, the local gradient orientations are computed and collected in a histogram, where each bin counts the number of edge points at which the image gradient has a certain orientation (see Fig. 3.5, center). For the second descriptor we compute for each edge segment a polyline approximation. Then, we collect all angles between the line segments and the horizontal axis in a vector. The final descriptor used is then a concatenation of both presented descriptors.

For classification we use a combination of AdaBoost and a voting method. First, the descriptors obtained from a test image are classified using AdaBoost. Then, those that are classified as foreground (i.e. “pedestrian”) are used for voting, i.e. they are matched to a previously generated codebook of descriptors with displacements from the object centers. The generated votes are then clustered using the mean-shift algorithm of Comaniciu et al. [2001]. Thus, the training phase consists of an AdaBoost training step and the collection of the codebook, as it is done in the original ISM approach, while in the inference phase positively classified descriptors vote for the occurrence of a pedestrian at a far distance.

#### 3.3.2. Results

We evaluated our method on the NICTA large pedestrian image dataset (see Overett et al. [2008]), which contains pictures with pedestrians taken in a typical urban environment. The pedestrians appear either alone or in crowded scenes with partial occlusions, in different poses (walking, standing, sitting) and in a broad range of lighting variations. Negative examples are represented by random crops of images from indoor and outdoor environments. We randomly selected 10000 positive images and 50000 negative images for each scale. In total we trained our algorithm with 120000 image samples. In each image we encountered between 10 and 20 edge segments, i.e. several million descriptors were used for training. We used 5 times more negative training examples to provide a large variety of background. To assess the quality of the AdaBoost training we used a leave-one-out cross validation, in which data is partitioned into subsets such that the analysis is initially performed on a single subset, while the other subsets are retained for confirming and validating the initial results. The test set is composed of 24000 images with 4000 and 20000 negative examples. Our results on images of size  $16 \times 20$  are shown in table 3.5 (right). As we can see, the approach is comparably robust. Compared to the approach of Kruppa et al. [2003], which uses AdaBoost on Haar features, we obtain a precision and recall of about 70%, while the latter resulted in 28% precision and 18% recall.

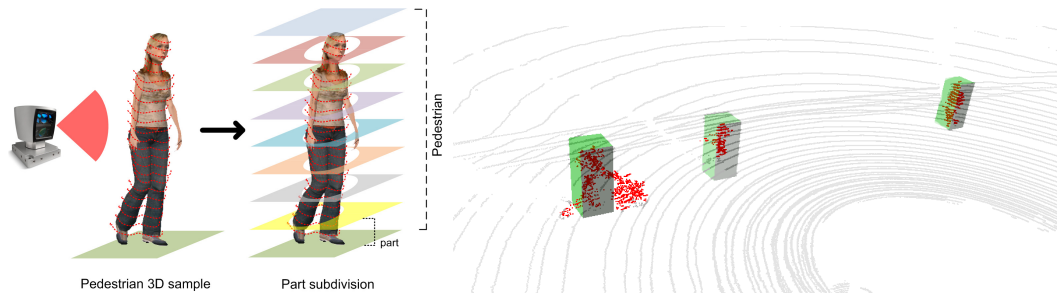
#### 3.4. Detecting Pedestrians in 3D Range Data

In Sec. 3.2 we saw how additional data obtained from a 3D range sensor can be used to further constrain the regions of interest, in which pedestrians are detected. The benefit of this method mainly stems from the ability of 3D range sensors to determine the correct scale of an object, because the sensor returns very accurate depth measurements. This raises the question whether we can use 3D range data directly to detect pedestrians *instead* of using a combination of different 2D sensor modalities. To answer this question we have to take into account that cameras still have the major advantage over laser sensors that their data is much denser and that color and texture is often much more informative than shape. However, the development of sensor systems for 3D perception has advanced very quickly in recent years, and modern 3D laser devices can already produce millions of data points at very high frame rates. This data density results in a very high resolution at which objects can be scanned even at distances of about  $10m$ , which somehow reduces the advantage of standard color cameras over these 3D scanners. Therefore, we develop in this section a pedestrian detection algorithm that only relies on such dense 3D point clouds. To do this, we first observe that 3D scanners which rotate around the vertical  $z$ -axis produce data that can be interpreted as a vertical collection of 2D scan lines. This means that we can subdivide the 3D data vertically and apply similar techniques as we did for the 2D case. The details of this are given in Spinello et al. [2010a] on page 118, here we present the main idea and the results.

##### 3.4.1. Classification by Splitting into Parts

In general, classification of humans in 3D data is very difficult, because the appearance of people is highly variable: Humans have different sizes and body shapes, wear clothes,



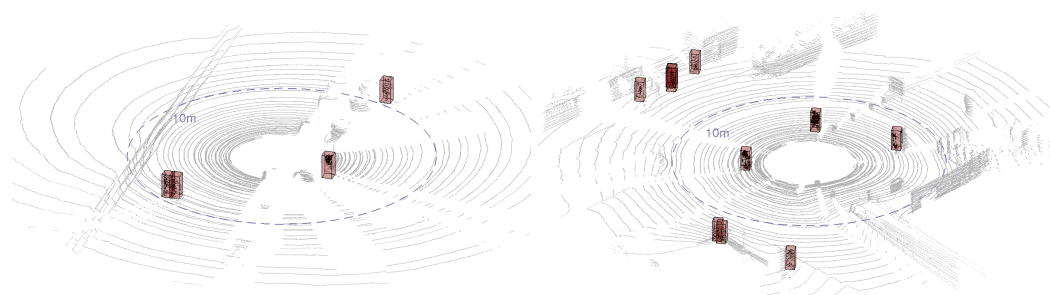


**Figure 3.6.:** **Left:** Learning a 3D pedestrian model. Objects are vertically divided into  $M$  parts. For each subpart an independent AdaBoost classifier is learned: all the segmented points contained on each layer are considered as positive samples for the  $M$  AdaBoost classifiers. **Right:** Detection result on a 3D point cloud. A person pushing a buggy, a child and a walking pedestrian are correctly identified in the point cloud.

carry bags, backpacks, or umbrellas, pull suitcases, or push buggies, making it hard to predefine models for their appearance. To simplify the problem, we therefore use a subdivision into  $M$  different height layers called *parts* (see Fig. 3.6, left). The subdivision is defined beforehand and does not follow any anatomical semantics such as legs, trunk, or head. Then, to learn a 3D person model, we create  $M$  different and independent classifiers, one for each part. We use AdaBoost as a classification algorithm, as it has shown to perform very well on similar data from 2D sensors [Arras et al., 2007]. To train the classifier, we first compute a set of descriptors on the previously segmented horizontal scan lines, where the descriptors include features such as width, mean curvature, or radius of a segment (the full list is given in Spinello et al. [2010a] on page 119). Then, from a given 3D data set of pedestrians where the parts are labeled, a classifier is trained for each part. Furthermore, displacement vectors pointing to the object center are stored for the parts. Then, in the inference step, scan lines are first classified into the parts, and the corresponding displacement vectors are used as votes for the position of the pedestrian.

### 3.4.2. Results

We evaluated our algorithm on two outdoor data sets collected with a Velodyne HDL 64E S2 laser scanner. The first data set, named *Polyterrasse*, has been collected in a large area in the front of the ETH Zurich main building, accessible only to people and bicycles. The second data set, named *Tannenstrasse*, has been collected on a busy street crossing in downtown Zurich with trams, cars, pedestrians, or bicycles. Training was done using 2592 background segments and 7075 people segments from the *Polyterrasse* data set with 203 subjects, which were standing still or walking. A qualitative result is shown in Fig. 3.7. Quantitatively we obtained for *Polyterrasse* an equal error rate (i.e. precision and recall) of 96%, 71%, and 65% where pedestrians were up to 10, 15, and 20m away from the sensor, respectively. For *Tannenstrasse*, these values were 95%, 76%, and 63%.



**Figure 3.7.:** Two example frames showing detection results as red boxes. **Left:** A frame of the *Polyterrasse* data set. Closely walking people and a partly visible individual standing close to the sensor are correctly found. **Right:** A frame of the *Tannenstrasse* data set showing a cluttered urban street crossing with people, pillars, street signs, and a tram just entering the scene. People are correctly detected while crossing the street and walking at a large distance to the sensor. There are two false positives in the lower left part of the picture caused by a glass window with vertical steel poles.

## 4. Confidence-aware Classification

As we have seen in the previous chapter, standard supervised learning methods such as AdaBoost and Support Vector Machines are very powerful methods for challenging tasks such as the detection of pedestrians and cars in camera and laser range data. In particular, we saw that by using informative data descriptors and an appropriate application of the learning methods, very high classification rates can be achieved. Thus, we could argue that these classifiers are the methods of choice for the kind of problems we are considering. Unfortunately, it turns out that these methods have some severe drawback, and this drawback is particularly evident in applications where safety-critical decisions depend on the return values obtained from the classifier. An example of this is the decision to stop an autonomous car when the classifier returns a detected pedestrian in front of the car. As we will argue in Sec. 4.1, standard classifiers can in this case lead to catastrophic situations, because their measure of *confidence* is unreliable. Therefore, we investigate alternative classifiers in this chapter, which provide better confidence estimates. In particular, in Sec. 4.2 we will briefly describe the Gaussian Process classifier (GPC) and present an example application. Then, in Sec. 4.3 we apply the GPC to traffic light detection and road sign classification, which are further examples of safety-critical problems. Finally, in Sec. 4.4 we develop a novel classification method, that is more efficient than the GPC but not much worse in terms of its confidence estimation.

### 4.1. Motivation

Consider again the scenario where an autonomous car operates in a busy urban environment. One of the main challenges in this scenario is the *reliable* detection of other road users such as cars, cyclists, and pedestrians. By reliable we mean that the detection method we use should return wrong detections only very rarely. However, it is clear that even the best detection method will always fail in some situations, even though this might happen in very few cases. Unfortunately, only one such failure case can already lead to disastrous situations, for example if a pedestrian is not detected and hit by the autonomous car. In contrast, if the detection method reports a pedestrian, where in fact there is free space, this is less serious, because although the car might decide to stop unnecessarily, it does at least not harm anyone. This means, that in principle there are two different kinds of failure cases with different levels of severity: the false positive and the false negative detections, while the latter are by far the more critical ones. Thus, the question is how the number of false negative detections can be reduced. To answer this, we first note that classifiers generally make wrong predictions particularly when they are presented with a test datum which is unlike anything they saw during training. In these cases, we argue that the appropriate response for a classifier is to respond with high uncertainty, or equivalently with low confidence. This means that we need to provide uncertainty estimates *in addition* to

the class label predictions produced by the classifier. For the case of the Support Vector machine (SVM), Platt [1999] proposed a method to compute probabilistic outputs, as will be described next.

#### 4.1.1. Obtaining Uncertainty from an SVM

As explained in Sec. 2.1.2 on page 8 the standard SVM inference algorithm computes for a new test datum  $\mathbf{x}^*$  the decision function  $f(\mathbf{x}^*)$  from Eq. (2.4) and returns the label  $y^* = 1$  if the sign of  $f$  is positive. To convert this binary output into a probabilistic formulation, Platt [1999] uses the formulation

$$p(y^* = 1 | f(\mathbf{x}^*)) = \frac{1}{1 + \exp(af(\mathbf{x}^*) + b)}. \quad (4.1)$$

This corresponds to a parametric sigmoid model that is fit directly to the class posterior. The parameters  $a$  and  $b$  of this sigmoid function are obtained by maximizing the likelihood of the training data using three-fold cross-validation. To further reduce the risk of overfitting, the training labels  $y_1, \dots, y_N$  are converted into *target probabilities*  $t_1, \dots, t_N$  as

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = 1 \\ \frac{1}{N_- + 2} & \text{if } y_i = -1 \end{cases}, \quad (4.2)$$

which follows from Bayes' rule applied to an out-of-sample data model with a uniform prior over the labels.

As we will see later, the problem with this method is that class likelihoods are derived using only a *single* estimate of the discriminative boundary obtained from training the classifier. No other feasible solutions are considered. In particular, the predictive variance of the discriminant  $f(\mathbf{x})$  is not taken into account while determining probabilistic output (see Sec. 6.4.1 on page 145 of Rasmussen and Williams [2006]). Furthermore, there are no guarantees that the optimisation itself is well-behaved.

#### 4.1.2. Under- and Overconfidence

A crucial point with the uncertainty estimates obtained from a classifier is the question, how much one can rely on these estimates. Intuitively, the uncertainty estimated for a classification output is reliable if it correlates with the incorrectness of the classifier. This means that an ideal classifier should be uncertain only if its predictions are wrong and certain only if it is correct. This intuition is used in Triebel et al. [2013b] on page 174, where the point-biserial correlation coefficient (PBCC) was used to quantify the relation between incorrect and uncertain classifications. Based on this measure, an algorithm was developed to reduce the number of wrong, but certain classifications (see Sec. 4.4 for details). However, it turns out that the PBCC is not very useful in cases where the number of correctly and incorrectly classified samples is unbalanced. Therefore, we propose a different approach. First, we define the *total average confidence*  $\bar{c}$  of a classifier  $f$  on a test set  $\mathcal{X}^*$  of size  $K$  as

$$\bar{c}(f, \mathcal{X}^*) := 1 - \frac{1}{K} \sum_{\mathbf{x}^* \in \mathcal{X}^*} h(f(\mathbf{x}^*)), \quad (4.3)$$

where  $h$  is the uncertainty of the prediction  $f(\mathbf{x}^*)$ , which can for example be defined as the *normalized entropy*

$$h(y) = - \sum_{i=1}^C p(y = c_i | \mathbf{x}) \log_C p(y = c_i | \mathbf{x}), \quad (4.4)$$

where  $C$  is the number of classes and  $c_i$  are the class labels. Thus,  $\bar{c}$  is a value that is 1 iff all prediction uncertainties are 0 and vice-versa. Next, we define two functions  $u$  and  $o$  as follows:

$$u(f, \mathcal{X}^*, \hat{\mathcal{Y}}) := \frac{\sum_{\mathbf{x}^* \in \mathcal{X}^*} I(y^* = \hat{y}) h(f(\mathbf{x}^*))}{K_c \bar{c}(f, \mathcal{X}^*)} \quad (4.5)$$

$$o(f, \mathcal{X}^*, \hat{\mathcal{Y}}) := \frac{\sum_{\mathbf{x}^* \in \mathcal{X}^*} I(y^* \neq \hat{y})(1 - h(f(\mathbf{x}^*)))}{K_f(1 - \bar{c}(f, \mathcal{X}^*))}, \quad (4.6)$$

where  $I$  is the indicator function,  $\hat{\mathcal{Y}}$  are the ground truth labels, and  $K_f$  and  $K_c$  are the number of incorrectly and correctly classified test samples, i.e.  $K_f + K_c = K$ . The interpretation of these functions is that  $u$  rates the average uncertainty of the correct classified samples over the total average confidence. Similarly,  $o$  sets the average confidence of the incorrectly classified samples in relation to the total average uncertainty. We will denote  $u$  as the *normalized underconfidence* and  $o$  as the *normalized overconfidence* of the classifier. Intuitively, if all incorrect classified samples have the maximum confidence of 1 assigned, then the overconfidence is very high. The other extreme case is that of maximal *underconfidence*  $u$ . Here, all uncertainty values  $h$  for correctly classified samples are 1, i.e. the classifier is always fully uncertain, although its predictions are correct. We note that with this definition under- and overconfidence are decoupled, as there may be cases where a classifier is under- and overconfident at the same time, namely when it is uncertain on the correct predictions and certain on the wrong ones. Based on this intuition of under- and overconfidence, we now turn to a classifier that performs better in this sense.

## 4.2. A Less Overconfident Classifier

One group of machine learning techniques that have particularly attracted the interest of researches in robotics are probabilistic reasoning techniques. They are used very often and with great success in all kinds of robotics applications, because they tend to allow for a more precise modelling. Conditional Random Fields are one example, and another very successful technique are Gaussian Processes (GP). Originally, GPs are used for regression problems, where the predictions made by the algorithm are real numbers rather than indices of class labels as in classification. However, there is also a classification framework based on GPs, which we will use here. Before, we give a brief explanation of the Gaussian Process Classifier. For more details on this topic we refer to Rasmussen and Williams [2006].

### 4.2.1. The Formulation of the Gaussian Process Classifier (GPC)

According to Rasmussen and Williams [2006], a Gaussian Process (GP) is defined as a collection of random variables, any finite number of which are jointly Gaussian distributed.

One can think of a GP as a distribution over functions rather than vectors, and to specify a GP we need to name a *mean function*  $m$ , which is unary, and a binary *covariance function*  $k$ . Then, we can write the distribution over functions  $f$  as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \quad (4.7)$$

Most often, and also in our case, the mean function is assumed to be the zero function, which means that the covariance function alone specifies the GP. Many different covariance functions have been proposed, but by far the most used one is the squared exponential function defined as

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp((\mathbf{x}_p - \mathbf{x}_q)^T D (\mathbf{x}_p - \mathbf{x}_q)) + \sigma_n^2 \delta_{pq} \quad p, q = 1, \dots, N, \quad (4.8)$$

where  $D$  is a  $d \times d$  diagonal matrix,  $d$  is the dimension of the feature vectors  $\mathbf{x}$ , and  $\delta$  is the Kronecker delta. The parameters  $D$ ,  $\sigma_f$ , and  $\sigma_n$  are known as the *hyper-parameters* of the model.

To use GPs for prediction, we assume we are given a set of training inputs  $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  and outputs  $f_1, \dots, f_N$ , where for now we consider  $f_i \in \mathbb{R}$ . Then, given a test sample  $(\mathbf{x}_i, f_i)$  we consider the joint distribution of all training and test samples, which is a multivariate Gaussian:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K & \mathbf{k}_* \\ \mathbf{k}_*^T & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right). \quad (4.9)$$

Here,  $K$  is the covariance matrix of all training inputs, i.e. the  $N \times N$  entries of  $K$  are equal to  $k(\mathbf{x}_i, \mathbf{x}_j)$ ,  $i, j = 1, \dots, N$ . Similarly, the vector  $\mathbf{k}_*$  corresponds to the covariance function applied to all training samples and the test sample  $\mathbf{x}^*$ . Using this joint distribution, the *predictive distribution* for a test output  $f^*$  can now be obtained by conditioning on the observations, i.e.

$$p(f^* | \mathbf{f}, X, \mathbf{x}^*) = \mathcal{N}(\mathbf{k}_* K^{-1} \mathbf{f}, k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}_*^T K^{-1} \mathbf{k}_*). \quad (4.10)$$

This formulation assumes that we know the exact values  $\mathbf{f}$  of the training output, which is, however, most often not the case. In fact, we usually only know a noisy version  $\mathbf{y}$  of  $\mathbf{f}$ , but we can assume that we know the noise model  $p(\mathbf{y} | \mathbf{f})$ . For example, in the standard case of regression, this noise model is a zero-mean multivariate Gaussian. Noting that the prior  $p(\mathbf{f} | X)$  is specified by the GP, we can compute the posterior as

$$p(\mathbf{f} | X, \mathbf{y}) = \frac{p(\mathbf{y} | \mathbf{f})p(\mathbf{f} | X)}{p(\mathbf{y} | X)}, \quad (4.11)$$

which is another multivariate Gaussian that can be computed in closed form. To perform predictions over new values  $y^*$ , we first need to marginalize over all  $\mathbf{f}$  using (4.10) and (4.11), and we obtain

$$p(f^* | X, \mathbf{y}, \mathbf{x}^*) = \int p(f^* | \mathbf{f}, X, \mathbf{x}^*) p(\mathbf{f} | X, \mathbf{y}) d\mathbf{f}. \quad (4.12)$$

This can again be done in closed form due to the nature of the multivariate Gaussians. Then, we need to marginalize over the (unknown)  $f_*$ , which results in

$$p(y^* | X, \mathbf{y}, \mathbf{x}^*) = \int p(y^* | f^*) p(f^* | X, \mathbf{y}, \mathbf{x}^*) df^*, \quad (4.13)$$

where we again made use of the Gaussian noise model  $p(y^* | f^*)$ . Now, to switch from the regression to the classification case, the only modification we make is that we assume  $y$  to be from the interval  $[0, 1]$ , and the noise model is a sigmoid function rather than a Gaussian. This has the drawback that (4.11) can not be computed in closed form. However, there are good approximation techniques such as the expectation-propagation (EP) algorithm by Minka [2001] that can be used to find mean and covariance matrix of a Gaussian that approximates (4.11). If EP is used in combination with a probit sigmoid function for the noise term, i.e.

$$p(y | f) = \Phi(yf) \quad \text{where} \quad \Phi(z) = \int_{-\infty}^z \mathcal{N}(x | 0, 1) dx, \quad (4.14)$$

then the prediction (4.13) can be computed in closed form and does not need to be approximated.

The training phase of the GPC consists of maximizing the marginal likelihood  $p(\mathbf{y} | X)$  with respect to the hyperparameters of the covariance function (see (4.8)). This can be done by standard gradient-based optimization methods. For more details we refer again to Rasmussen and Williams [2006].

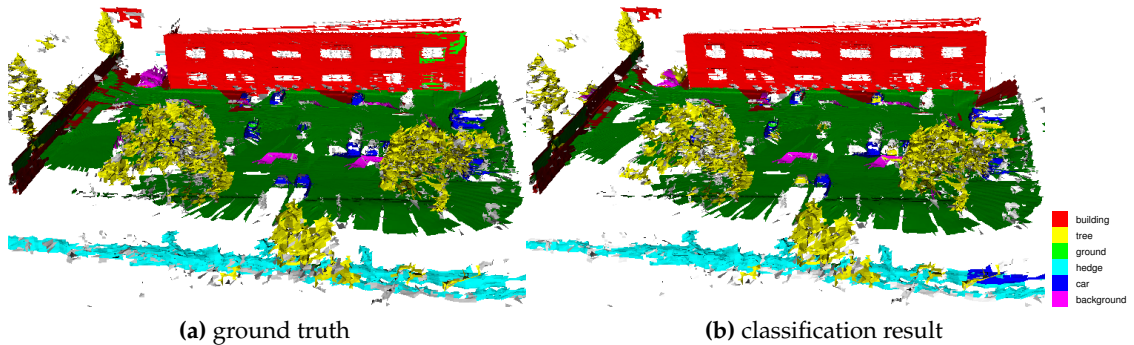
#### 4.2.2. The GPC for Multiple Classes

In the standard case, the Gaussian Process classifier is used for binary classification, i.e. the prediction (4.13) is interpreted as the probability for class 1 (or foreground). In general, however we have classification problems involving multiple classes. To do this with a GPC, several approaches have been proposed, and we use the one proposed by Girolami and Rogers [2006]. Again, we denote the  $d$ -dimensional feature vectors used for training with  $\mathbf{x}_1, \dots, \mathbf{x}_N$  and the corresponding class labels with  $y_1, \dots, y_N$ , however now  $y_i \in \{1, \dots, C\}$  and  $C$  is the number of classes. Furthermore, a *latent* function  $f_j(\mathbf{x})$  is introduced for each class along with the *probit regression* model. Thus, the probability of a class label  $y_i$  for a given feature vector  $\mathbf{x}_i$  is defined as:

$$p(y_i = j | \mathbf{x}_i) = \Phi(f_j(\mathbf{x}_i)) \quad i = 1, \dots, n, \quad j = 1, \dots, k. \quad (4.15)$$

Unfortunately, the EP approximation used in the binary case can not be applied in a straightforward way to the multi-class case. Therefore, Girolami and Rogers [2006] propose a variational Bayes formulation. Then, during training the hyperparameters are learned by gradient ascent on the estimated marginal likelihood. Once the hyperparameters and the latent posterior are obtained from training data, inference is performed on new test input by applying Equation (4.13), as in the binary case.

In terms of computation time, the full GP classification procedure scales as  $O(kN^3)$  where  $k$  is the number of classes and  $N$  is the total number of sample points. The scaling is dominated by the cubic dependence on  $N$  due to the matrix inversion required to obtain the posterior mean for the GP variables. The variational Bayes multi-class GP formulation [Girolami and Rogers, 2006] is amenable to a sparse approximation by constraining the maximum number of samples  $s$  included in the model. This results in an  $O(kns^2)$  scaling where  $s \ll n$ . The informative points are picked according to the posterior predictive probability of the target value, intuitively picking points from class boundary regions which are most influential in updating the approximation of the target posteriors.



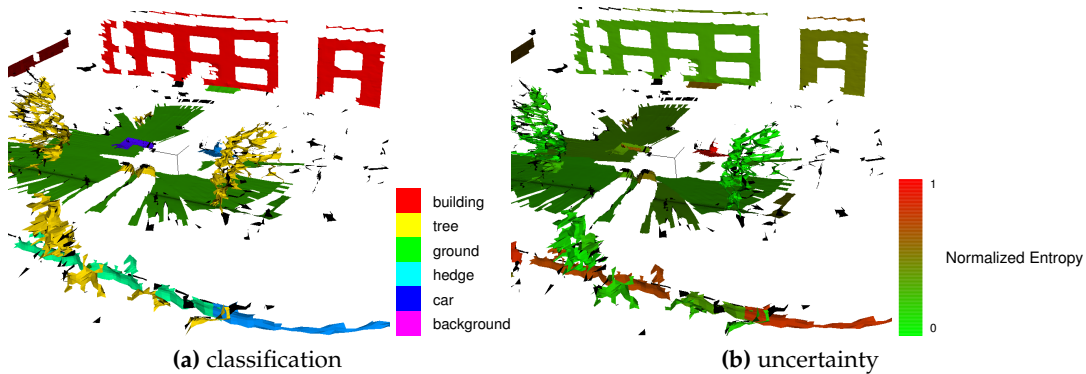
**Figure 4.1.:** Classification result after 9 point clouds. **Left:** Ground truth. Note that even in the ground truth some areas are not labeled correctly, e.g. on the ground close to the *building*. This is due to the fact that the mesh segmentation is not perfect and a correct manual labeling of segments that actually correspond to more than one class is not possible. In our evaluation we abstract from such segmentation errors. **Right:** Classification result. Only minor errors are visible. Note the *hedge* in the front, but also on some *cars*.

### 4.2.3. Application to 3D Point Cloud Classification

In Paul et al. [2012] on page 159, we applied the multi-class GPC to the problem of segmenting 3D point cloud data into classes such as building, tree, hedge, etc. The data was acquired with a rotating laser scanner device. In a preprocessing step, we first produce a triangular mesh from a 3D point cloud by connecting neighboring data points if they are closer than a given threshold. We then compute normal vectors for all triangles and apply a segmentation algorithm based on the work of Felzenszwalb and Huttenlocher [2004], where the similarity of two adjacent triangles is defined by the angles of their normal vectors. Each resulting mesh segment consists of a single connected component and is consistent with respect to the orientation of the triangles it contains. Thus, segments are consistently shaped, e.g. all triangles are all mostly co-planar or they are all similarly distributed in orientation. In the next step, we compute feature vectors for all mesh segments. We use similar features as in Triebel et al. [2010] (see page 204), namely *shape factors*, *shape distributions* based on Euclidean distance, on angles between normal vectors and on the elevation of the normal vectors, and finally *spin images*, where the latter are computed per data point and then an average is computed per mesh segment. As a result, we obtain a 113 dimensional feature vector for each mesh segment, where 50 account for the  $5 \times 10$  spin image, 20 for each shape distribution and 3 for the shape factors. These feature vectors, together with a set of ground truth class labels are then fed into the training algorithm of the GP multiclass classifier.

Figure 4.1 shows the classification result of 9 consecutive meshes in one common image. Note that there are slight labeling errors even in the ground truth (Fig. 4.1a). This is caused by imperfections during segmentation, which lead to under-segmentation. For example, few segments contain 3D points from the building and the ground. As it is impossible to determine a single *true* label for these segments, we decided to assign the ground truth label based on a majority voting. From Fig. 4.1b we can see that the classification is very good, only the under-represented classes such as *car* and *hedge* are classified slightly worse.



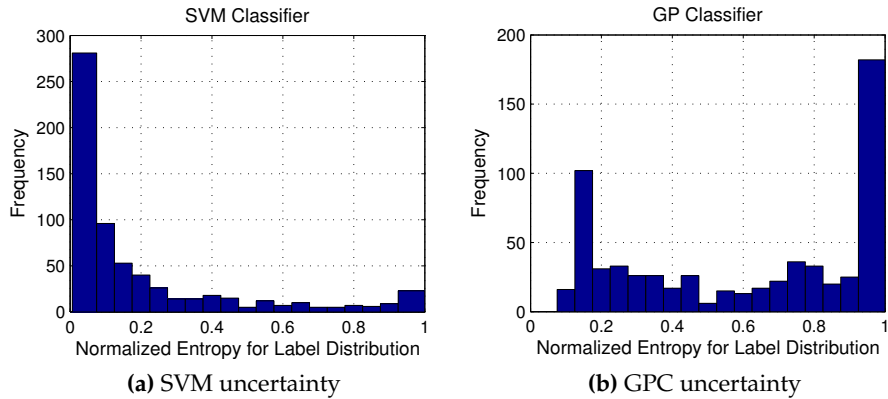


**Figure 4.2.:** Classification result and normalized entropy for one example mesh. (a) Classification result using multi-class GP classification. Note the classification error of the *hedge* in the front, which is classified as *car*. (b) Uncertainty in terms of normalized entropy of the class label distributions for each mesh segment. For most segments the classifier is very confident. For some, such as the (wrong classified) *hedge* in the front, the normalized entropy is high and the classification confidence low.

Fig. 4.2 shows an example of the classification result for one triangle mesh from our data set. Fig. 4.2a depicts our classification result using the multi-class GP classifier. One can clearly see that there are only minor classification errors. The most obvious ones are in the front on the hedge surrounding the car park. Here, the classifier generated the label *car*. However, the labels in that area are not very certain, which can be seen from the Fig. 4.2b. Here, the uncertainty in terms of the normalized entropy (see Eq. (4.4)) is visualized with color values between green (no entropy) and red (entropy equal to 1). We can see that the class label distributions of the segments in the front have a much higher entropy than others such as those on the ground. This shows evidence that the GP classifier tends to be uncertain on the wrong classifications and certain on the correct ones.

We compared the generative GP classifier with a discriminative SVM classifier using the LIBSVM implementation of Chang and Lin [2011]. In all cases, we employed the squared exponential kernel to facilitate comparison. Evaluated in terms of the  $F_{0.5}$ -measure, the performance for the GP and SVM classifiers was very similar, even with a sparser representation used for the GP. Then, we compared the uncertainty estimates of the probabilistic classification output of the two classifiers to new object classes not used in training. We trained the GP and SVM classifiers only on segments from two classes (randomly picked): *building* and *ground*. Data from the remaining *un-modeled* classes was presented to both classifiers for inference, resulting in a classification distribution over binary labels. The normalized entropy values measuring uncertainty in the classification decision were computed for each label distribution.

Fig. 4.3 presents the normalized entropy histograms for the inference set. The SVM classifier commits a large majority of the *un-modeled* points to one of the *modeled* classes with high certainty, resulting in a peaked distribution over one of the two labels. As a result, for a majority of the data points, the label distribution has lower normalized entropy. In contrast, the GP classifier assigns higher normalized entropy for a majority of the test points. The same pattern was found consistent for other choices of training and testing classes.



**Figure 4.3.:** Histogram of normalized entropy values of the label distribution for SVM and GP classifiers. Both classifiers were trained explicitly on two classes. The data from the remaining classes was presented for inference. (a) SVM classifier assigns a majority of the points to a particular class with high certainty. (b) As a contrast, GP classifier assigns greater classification uncertainty to a majority of the points, providing evidence for a potential new class. Note the scale on y-axis.

The classifier uncertainty for the test points from new classes is expressed as a more uncertain (uniform) distribution over labels, indicating the presence of one or more potentially *un-modeled* classes.

### 4.3. The Importance of Confidence in Mission-Critical Classification

As mentioned above, the use of a confidence-aware classifier is particularly important in situations where decisions depend on the outcome of the classification, which can have catastrophic consequences. A classifier that performs well in terms of precision and recall might return only few misclassifications, but if these few mistakes are made with high confidence, then it is not possible to mitigate them. Therefore, it could be advantageous to use a classifier that is slightly worse in terms of precision and recall, but that at least returns a low confidence on the wrong classified samples. This idea was followed in Grimmett et al. [2013] on page 166, where a binary GPC was used to detect traffic lights and classify road signs in camera images. We briefly explain the approach here.

#### 4.3.1. Traffic Light Detection and Road Sign Classification

Our aim is to address traffic light detection and road sign classification with different classification frameworks and to compare them in terms of their tendencies for overconfidence. In investigating both classification and detection we aim to address the full spectrum of applications commonly encountered in robotics. The two are distinct in that classification addresses the case where a decision is made between two, well-defined classes (e.g. two types of traffic signs) and investigates classifier performance as a third, previously unseen class is presented. The detection case is arguably the more common one in






semantic mapping where a single class is separated from a broad *background* class. Here, the concept of a previously unseen class does not exist but the inherent assumption is that the data representing the background class are sufficiently representative to capture any non-class object likely to be encountered. In practice, this is often not the case, leading to a significant number of misclassifications. While it could be argued that this problem is ameliorated somewhat by expanding the dataset used for training, we propose that the complexity of the workspaces encountered during persistent, long-term autonomy will keep perplexing even the most rigorously trained classifier.

For feature computation, we use the approach proposed by Torralba et al. [2007b], in which a dictionary of partial templates is constructed, against which test instances are matched. A single feature consists of an image patch (ranging in size from  $8 \times 8$  to  $14 \times 14$  pixels) and its location within the object as indicated by a binary mask ( $32 \times 32$  pixels). For any given test instance, the normalised cross-correlation is computed for each feature in the dictionary. Therefore, per instance (or window, in the detection case) a feature vector of length  $d$  is obtained, where  $d$  is the size of the dictionary. We found empirically that  $d > 200$  leads to negligible performance increase in classification. Throughout our experiments we therefore set  $d = 200$ .

#### 4.3.2. Results

We evaluated the performance for six different classifiers, both in terms of classification rate and with respect to their overconfidence. The classifiers used were a GPC with a linear covariance function (kernel), a GPC with a squared-exponential (SE) kernel, an SVM also with a linear and an SE kernel, the LogitBoost [Friedman et al., 1998] algorithm, and an Informative Vector Machine [Lawrence et al., 2002], which is a sparse version of the GPC. We applied these classifiers to a subset of the German Traffic Sign Benchmark (GTSRB) dataset [Stallkamp et al., 2012], which comprises over 50,000 loosely-cropped images of 42 classes of road signs, with associated bounding boxes and class labels. From this dataset we specifically focus on the seven classes *roadworks ahead*, *right ahead*, *stop*, *keep left*, *lorries prohibited*, *speed limit*, and *yield*. First we evaluated the standard classification performance in terms of precision and recall. For that, we trained all classifiers on 400 samples of the two classes *stop* and *lorries prohibited*, 200 per class. Classifier performance was evaluated using precision and recall on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The results for all classifiers where values of precision and recall of almost 1, from which we conclude that all classifiers are able to reach a high classification rate when they are tested on known classes.

Then, we performed an experiment similar to the one described in Fig. 4.3: we trained all classifiers on the two classes *stop* and *lorries prohibited*, and tested them on the other five classes. The result is shown in Table 4.1. We can see that the three GP classifiers always return a higher uncertainty in terms of normalized entropy than the other classifiers. This result further supports the findings from Sec. 4.2.3, namely the apparent ability of the GPC to associate incorrect classifications with a higher uncertainty. Later, in Chapter 7 we will exploit this ability when we use the GPC in an Active Learning framework.

Test Class	Classifier	Normalised Entropy	
		$\mu \pm \text{std. err.}$	$\sigma \pm \text{std. err.}$
	IVM	<b>0.776 <math>\pm</math> 0.081</b>	0.145 $\pm$ 0.030
	Non-linear GPC	0.751 $\pm$ 0.087	0.152 $\pm$ 0.029
	Linear GPC	<b>0.776 <math>\pm</math> 0.108</b>	0.150 $\pm$ 0.041
	Non-linear SVM	0.476 $\pm$ 0.101	0.089 $\pm$ 0.056
	Linear SVM	0.664 $\pm$ 0.122	0.250 $\pm$ 0.041
	LogitBoost	0.019 $\pm$ 0.025	0.041 $\pm$ 0.073
	IVM	<b>0.794 <math>\pm</math> 0.117</b>	0.106 $\pm$ 0.026
	Non-linear GPC	0.779 $\pm$ 0.124	0.107 $\pm$ 0.024
	Linear GPC	0.777 $\pm$ 0.202	0.124 $\pm$ 0.058
	Non-linear SVM	0.537 $\pm$ 0.126	0.028 $\pm$ 0.036
	Linear SVM	0.494 $\pm$ 0.239	0.222 $\pm$ 0.049
	LogitBoost	0.016 $\pm$ 0.022	0.031 $\pm$ 0.059
	IVM	0.539 $\pm$ 0.140	0.173 $\pm$ 0.023
	Non-linear GPC	0.546 $\pm$ 0.144	0.168 $\pm$ 0.023
	Linear GPC	<b>0.569 <math>\pm</math> 0.166</b>	0.177 $\pm$ 0.026
	Non-linear SVM	0.407 $\pm$ 0.077	0.076 $\pm$ 0.053
	Linear SVM	0.315 $\pm$ 0.195	0.197 $\pm$ 0.058
	LogitBoost	0.008 $\pm$ 0.004	0.012 $\pm$ 0.026
	IVM	0.579 $\pm$ 0.133	0.137 $\pm$ 0.020
	Non-linear GPC	0.577 $\pm$ 0.130	0.136 $\pm$ 0.019
	Linear GPC	<b>0.585 <math>\pm</math> 0.188</b>	0.151 $\pm$ 0.029
	Non-linear SVM	0.488 $\pm$ 0.111	0.039 $\pm$ 0.034
	Linear SVM	0.177 $\pm$ 0.127	0.155 $\pm$ 0.056
	LogitBoost	0.014 $\pm$ 0.019	0.030 $\pm$ 0.056
	IVM	<b>0.931 <math>\pm</math> 0.025</b>	0.080 $\pm$ 0.026
	Non-linear GPC	<b>0.934 <math>\pm</math> 0.021</b>	0.079 $\pm$ 0.023
	Linear GPC	0.925 $\pm$ 0.031	0.085 $\pm$ 0.027
	Non-linear SVM	0.641 $\pm$ 0.168	0.100 $\pm$ 0.047
	Linear SVM	0.705 $\pm$ 0.142	0.212 $\pm$ 0.049
	LogitBoost	0.059 $\pm$ 0.103	0.077 $\pm$ 0.127

**Table 4.1.:** Mean and standard deviation of normalised entropies (including standard errors) from 40 iterations of classifier training and testing, each with a randomly created dictionary and both training and test datasets resampled. Results are presented for classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on five different unseen classes as shown.

**Algorithm 2:** Online Multi-class Gradient Boost [Saffari et al., 2010]

---

**Data:** training data  $(\mathcal{X}, \mathbf{y})$  with  $C$  classes  
**Input:** number of weak learners  $M$ , loss function  $\ell$ , agreement function  $a$   
**Output:** weak learners  $f_1, \dots, f_M$

```

1 Initialize  $(f_1, \dots, f_M)$ 
2 for  $n = 1, \dots, N$  do
3    $w_n \leftarrow 1$ 
4    $g_n \leftarrow 0$ 
5   for  $m = 1, \dots, M$  do
6      $f_m \leftarrow \text{UpdateWeakLearner}(f_m, \mathbf{x}_n, y_n, w_n)$ 
7      $\mathbf{p}_{nm} \leftarrow f_m(\mathbf{x}_n)$ 
8      $\alpha_{nm} \leftarrow a(\mathbf{p}_{nm}, y_n)$ 
9      $g_n \leftarrow g_n + \alpha_{nm}$ 
10     $w_n \leftarrow -\nabla \ell(g_n)$ 
11  end
12 end
```

---

## 4.4. An Alternative to the GPC

From the experiments shown in the previous section, we conclude that the Gaussian Process classifier is well suited for applications where a reliable estimate of the classification confidence is required. However, one major problem with the GPC is its huge demand in run time and memory. Therefore, in this section, we investigate a classification framework that is known to be efficient and effective, but at the same time can be modified so that it is less overconfident. A recently developed method with these requirements is the Online Multi-Class Gradient Boost (OMCGB) algorithm of Saffari et al. [2010] (see Algorithm 2), which we describe next, before we develop our modification of the algorithm denoted as Confidence Boosting.

### 4.4.1. Online Multi-Class Gradient Boost (OMCGB)

We start again with a training data set  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ , for which we are given ground truth labels  $\mathbf{y} = (y_1, \dots, y_N)$ . As in Sec. 4.2.2, we will consider the multi-class case, i.e.  $y_i \in 1, \dots, C$ , where  $C$  is the number of classes. Then, in addition to the training data, the OMCGB algorithm requires three further input parameters: a fixed number  $M$  of weak learners, a loss function  $\ell : \mathbb{R} \rightarrow \mathbb{R}$ , and an *agreement function*  $a : \mathbb{R}^C \times \mathbb{N} \rightarrow \mathbb{R}$ , which quantifies the amount of agreement between a class label prediction  $f(\mathbf{x}_n)$  and the corresponding ground truth label  $y_n$ . After initialization of the weak classifiers, the algorithm loops over all training data points and updates all weak classifiers for every new training sample  $(\mathbf{x}_n, y_n)$ . This online behaviour of the algorithm will be particularly attractive for our Active Learning framework developed in Chapter 7, because it avoids a recomputation of the underlying representation whenever a new ground truth label is queried from the user and added to the existing training set. As in offline boosting methods, every training sample  $\mathbf{x}_n$  has an assigned weight  $w_n$ , which is first initialized to 1. Then, every weak

classifier is updated with the new sample, its weight  $w_n$  and its ground truth label  $y_n$ . Note that the weak learner itself also must be an online algorithm, because otherwise the overall boosting method would not be online. In our experiments we used online random forests as weak classifiers.

The next step (line 7) is to obtain a label prediction  $\mathbf{p}_{nm}$  for the new training sample, which we represent as a distribution over the class labels. Then, the agreement with the ground truth label is computed. In standard OMCGB, this is defined as

$$a_g(\mathbf{p}_{nm}, y_n) = p_{nm}^{(y_n)} - 1/C, \quad (4.16)$$

i.e. it is directly related to the prediction for class  $y_n$ , here denoted as an upper index into the prediction vector  $\mathbf{p}_{nm}$ . The resulting agreement  $\alpha_{nm}$  is then accumulated, and a new weight  $w_n$  is computed for the sample from the negative gradient of the loss function of the sum of agreements. In Saffari et al. [2010], two different loss functions are investigated, but with little performance difference, so we decided to use the standard exponential loss  $\ell(g) = \exp(-g)$  known from AdaBoost. Concretely, the computation in line 10 results in higher weights for samples that disagree with the ground truth and lower weights for those that do agree.

#### 4.4.2. Confidence Boosting

As can be seen from Eq. (4.16), the agreement  $a_g$  used by standard gradient boost is only related to the prediction itself, but not to the confidence of the prediction. To build a classifier that takes both prediction and confidence into account, we propose to use this agreement function:

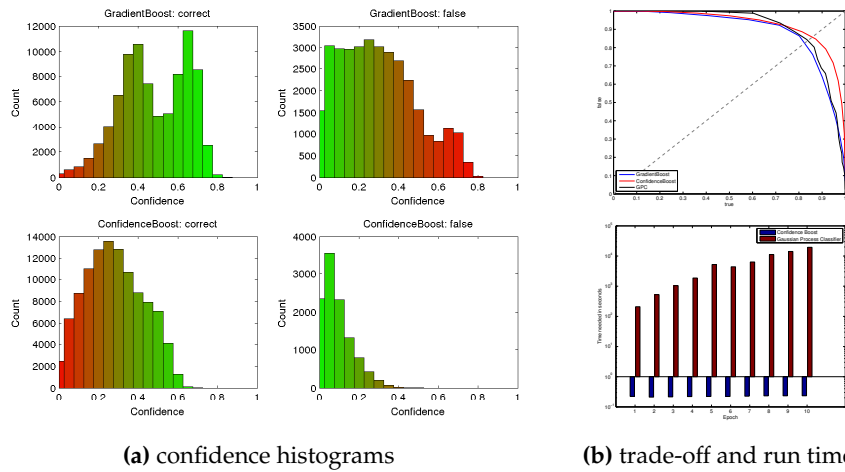
$$a_c(\mathbf{p}_{nm}, y_n) = (-1)^\xi \left( 1 - \frac{h(\mathbf{p}_{nm})}{(C-1)^\xi} \right), \quad (4.17)$$

$$\text{where } \xi = I(\arg \max_i \mathbf{p}_{nm}^{(i)} \neq y_n). \quad (4.18)$$

This means, that we measure agreement by the amount of confidence, which is equal to one minus the uncertainty  $h$ . In case of a correct classification, i.e. when  $\xi = 0$ , the agreement simply amounts to the confidence of the current weak classifier  $f_m$ . However, if the classification is incorrect, we actually have a disagreement, and we express this with the – slightly modified – *negative* confidence. Our modification is the term  $(C-1)$ , by which we divide the uncertainty. This has empirically shown to improve the classification results substantially. To summarize, our agreement function is high if the classification is correct and certain, and it is low if we have an incorrect, but certain classification. Also note that if the uncertainty is zero, i.e. when we completely trust the classification, then the agreement is 1 for correct and  $-1$  for incorrectly classified samples. Thus, in this case, our agreement function is even simpler than the original one given in Eq. (4.16).

#### 4.4.3. Results

To evaluate our algorithm, we performed experiments on six different data sets, where the aim was to show how much Confidence Boosting actually reduces the overconfidence in



**Figure 4.4.:** (a) Confidence histograms of gradient boosting and Confidence Boosting on the ‘DNA’ dataset. The left plots show the histograms for the correctly classified samples, the right ones those for the incorrect samples. Confidence Boosting shifts all histograms to the left, resulting in an overall decreased confidence. Note, however that the false samples are shifted even further, which leads to a lower normalized overconfidence. (b) **Top:** Trade-off curves for gradient boosting, Confidence Boosting, and GPC on the ‘Pendigits’ data set. Higher curves correspond to less overconfident classifiers, curves that are further to the right represent a lower underconfidence. We see that Confidence Boosting generally improves over- and underconfidence compared to gradient boosting, and it is even less underconfident than the GPC. **Bottom:** Average run times of Confidence Boosting (blue) and the GPC (red) for each epoch (note the log scale).

the class label predictions. We used four data sets from the UCI machine learning repository, and two sets from robotics. The UCI data sets are ‘USPS’, ‘Pendigits’, ‘Letter’, and ‘DNA’. We used these because they were also used for evaluation by Saffari et al. [2010], and our aim is to compare Confidence Boosting with gradient boosting. The robotics data sets we used were an RGB-D set provided by Lai et al. [2011], and the 3D point cloud data from Paul et al. [2012] (see page 159). From the first one, for which we use the identifier ‘RGBD’, we extracted 89 pre-segmented objects of 17 object classes, resulting in a total of 58372 RGB-D images. Then, we computed Hierarchical Matching Pursuit (HMP) descriptors [Bo et al., 2011] on the depth channel. The dictionary needed for the HMP features was learned on 5 classes out of 17, mainly for memory reasons. Then, the data was split into a training set of 90% of the data and an evaluation set of the remaining 10%. The other robotics data set, which we denote as ‘Begbroke’, consists of 3D point clouds from a car park with 6 classes. The data was segmented automatically, and features were computed for each segment. In total, there were 1496 segments, out of which we took 1000 for training and the rest for evaluation. We used this data to be able to compare with the multi-class GP classifier used in Paul et al. [2012], both in terms of run-time and classification performance.

To measure how much Confidence Boosting actually reduces overconfidence, we computed histograms over the confidences for correctly and incorrectly classified samples, both for gradient boosting and for Confidence Boosting. Fig. 4.4a shows the resulting histograms for gradient boosting (GB) and Confidence Boosting (CB) on the ‘DNA’ data set.

#### 4. Confidence-aware Classification

---

	RGBD	Begbroke	USPS	Letter	Pendigits	DNA
GradientBoost	0.1969	0.3899	0.2964	0.2611	0.3878	0.1988
Confidence Boosting	0.1125	0.2246	0.2308	0.1995	0.2462	0.1136

**Table 4.2.:** Normalized overconfidence averaged over 100 runs.

The plots on the left show the confidence histograms for the correct classified samples, the right ones for the incorrect samples (red bars depict either over- or underconfident regions). As we can see, Confidence Boosting tends to shift both histograms to the left, which means that in general classification is more uncertain. This implies that more false classifications are uncertain as well. Thus, if we measured overconfidence only by the false classified samples, then increasing the uncertainty in general would already reduce overconfidence. This underlines our definition of the normalized overconfidence, which also takes the overall confidence into account. To quantify this on the data sets used, we show the normalized overconfidences in Table 4.2. We see that Confidence Boosting reduces the normalized overconfidence on all six data sets.

To visualize the trade-off between over- and underconfidence, we use a plot type similar to a precision-recall curve. On the x-axis, we plot for a given number of different confidence thresholds  $\theta_1, \theta_2, \dots$  the fraction of false classified samples that have a confidence below the thresholds. On the y-axis, we plot for the same thresholds the fraction of correct classified samples for which the classification was more confident than  $\theta_i$ . Ideally, this curve stays close towards the upper right corner of the plot. Fig. 4.4b (top) shows an example of such a plot for the 'Pendigits' data set, both for gradient boosting and for Confidence Boosting. We see that Confidence Boosting gives the opportunity to 'detect' more false classifications while at the same time not losing too many correct classifications. The bottom part of Fig. 4.4b shows the average run times of Confidence Boosting and GPC, here for each epoch of Active Learning (see Chapter 7). As we can see, Confidence Boosting is orders of magnitude faster while not being more overconfident than the GPC.



## **Part II.**

# **Unsupervised and Online Learning**



## 5. Unsupervised Offline Learning

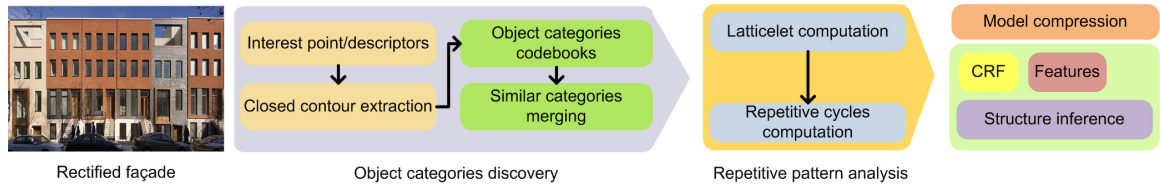
So far, we have only used learning methods that require ground truth information from a supervisor for the training data. For many applications, particularly when the task is to predict semantic information such as class labels for a given input data set, this ground truth information is indispensable, because without it a correct prediction is impossible. However, the process to obtain this ground truth labeling often signifies a huge effort for the human supervisor. Therefore, there is a growing interest in unsupervised machine learning methods, which rely on different sources of information than a hand-labeled training data set. In this chapter, we investigate such methods. In particular, we present algorithms that group the input data into meaningful clusters based on different criteria. In Sec. 5.1, we present a method that automatically discovers regular patterns as they occur for example in facades of buildings. In Sec. 5.2 we use data from motion commands and relate them with observed direction signs for semantic interpretation of the signs. And in Sec. 5.3 we use information about similarity and physical closeness of data segments to reason about their correspondence to potential object classes. Note that in all these purely unsupervised learning methods a semantic annotation that could be understood by a human can not be obtained. However, as we will see in Sec. 7.1, an unsupervised learning step can reduce the amount of required user interaction in a later supervised or semi-supervised learning step.

### 5.1. Learning from Repetition by Discovering Regular Patterns

As mentioned above, unsupervised learning methods can not rely on hand-labeled data to perform classification tasks. However, there are other sources of information that can be exploited, and one idea is to find repetitive patterns in the data (e.g. a camera image) and to infer from these patterns the existence of object instances. In Spinello et al. [2010c] (page 178), this idea is used to find, for example, windows in a facade of a building. As this approach is completely unsupervised, no human label such as “window” is returned by the algorithm. Instead, cluster ids are obtained along with correspondences of data segments to clusters. We give some more details in the following.

#### 5.1.1. Individual Steps of the Algorithm

Fig. 5.1 shows a flow chart of our unsupervised object discovery algorithm. The first step is to compute a set of descriptors and closed contours as candidates for repetitive objects (e.g. windows or pillars). The idea is to obtain evidence of object occurrence by extracting similarities directly from the given scene. Apart from not requiring training data, this has the advantage that we can filter out outlier categories for which no repetitive pattern can be found. Our similarity measure is based on the ISM approach (see Leibe et al.



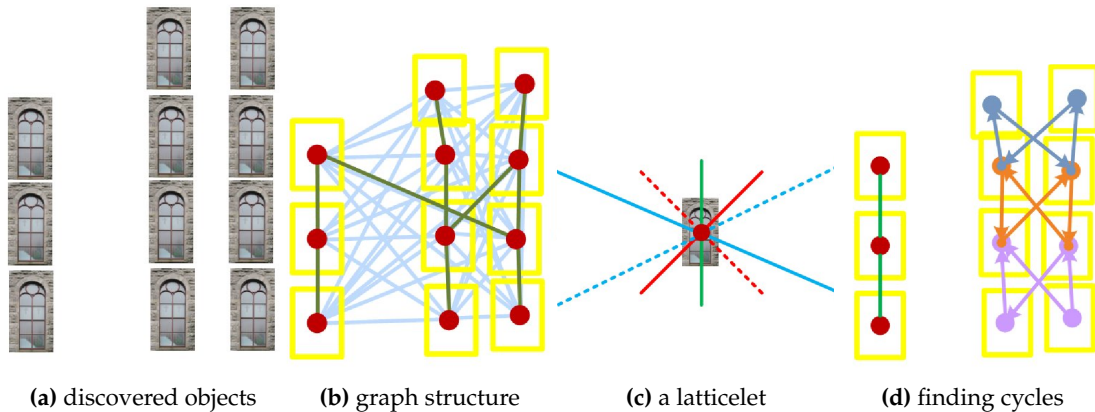
**Figure 5.1.:** Schematic overview of the unsupervised object discovery algorithm. The initial object discovery step consists of matching descriptors inside pairs of closed contours using the ISM approach of Leibe et al. [2005]. Then, the repetitive structure of the objects in the scene is analysed. Finally, this structure is used to reason on the existence and location of further object instances in the scene.

[2005]). An example result of this first step is shown in Fig. 5.2a. Then, we analyze repetitive patterns inside each category by looking at the Euclidean distances between elements in the image accumulated in a frequency map. These relative positions are represented as edges in a lattice graph, in which nodes represent objects positions (see Fig. 5.2b). The most dominant edges by which all nodes in this graph can be connected are found using a Minimum Spanning Tree algorithm and grouped into a set, which we call a *latticelet* (Fig. 5.2c). To reason on higher-level repetitions we first find cycles of repetitions of object candidates in the lattice graph, which can consist of simple polygonal structures such as triangles, quadrangles or pentagons (see Fig. 5.2d). Then, we extrapolate the graph structure by repeating the cycles at the graph boundaries. A probabilistic inference method using a CRF (see Sec. 2.3) is then used to determine if the occurrence of an object instance at a predicted position is likely or not. These last two steps will be explained next.

### 5.1.2. Structure Learning and Reasoning Using the CRF

To reason about the existence of repetitive objects we use a probabilistic model: each possible location of an object of a given category  $\tau$  is represented as a binary random variable  $l_\tau(\mathbf{x})$  which is true if an object of category  $\tau$  occurs at position  $\mathbf{x}$  and false otherwise. In general, the state of these random variables can not be observed, i.e. they are *hidden*, but we can observe a set of features  $\mathbf{z}(\mathbf{x})$  at the given position  $\mathbf{x}$ , which in this case correspond to the detection quality  $q$  of the objects. Now we need to find states of all binary variables  $\mathbf{l}_\tau = \{l_\tau(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$  so that the likelihood  $p(\mathbf{l}_\tau \mid \mathbf{z})$  is maximized, taking into account the *conditional dependence* between variables labels  $l_\tau(\mathbf{x}_1)$  and  $l_\tau(\mathbf{x}_2)$  of neighbors  $\mathbf{x}_1$  and  $\mathbf{x}_2$ . This is modeled using a CRF, where the node features are  $\mathbf{f}_n(q_i, l_{\tau,i}) = 1 - l_{\tau,i} + (2l_{\tau,i} - 1)q_i$ , and the edge features are based on the detection quality of the neighboring nodes (see Spinello et al. [2010c], page 185).

To learn node and edge weights  $\mathbf{w}_n$  and  $\mathbf{w}_e$ , we observe that they do not depend on the network geometry but only on its topology, thus we can artificially generate training instances by setting up networks with a given topology and assigning combinations of low and high detection qualities  $q_i$  to the nodes. This way we can create a higher variability of possible situations than seen in real data and thus obtain a higher generalization of the algorithm. The topology we use for training has a girth of 3 but other topologies can be used, e.g. using squared or hexagonal cycles, although from our experiments these topologies do not increase the classification result. We also found that the number of outgoing



**Figure 5.2.:** Analysing the structure of the repetitive pattern. From the initial object detections in (a), we build a graph structure and find the minimum spanning tree (b). From this, we create a set of edges called a latticelet (c) and find circular structures (d), which are then used to extrapolate latticelets at the borders of the network.

edges per node, i.e. the *connectivity*, has a strong influence on the learned weights. Thus, we use a training instance where all possible connectivities from 2 to 6 are considered.

In the inference phase, we create a CRF by growing an initial network. From the analysis of repetitive patterns described above, we obtain the set of cycles  $\mathcal{G}$  for each category, the topology and edge lengths of the lattice. By subsequently adding cycles from  $\mathcal{G}$  to the network we grow it beyond its current borders. After each growing step, we run loopy belief propagation to infer the occurrence of objects with low detection quality. The growth of the network is stopped as soon as no new objects are detected in any of the 4 directions from the last inference steps.

### 5.1.3. Results

Fig. 5.3 shows some results of our method on typical input images. In the upper row of the figure, we see the results of the object discovery step before applying structure learning and the CRF inference. Thus, this is the result of a standard similarity based clustering step where similarity is measured by the number of descriptor matches inside two object candidates. As we can see, many object instances are clustered correctly, but some further occurrences of objects (here mostly windows) have been missed. This is due to a too low detection score obtained for these instances. However, if we apply our structure learning method and run CRF inference on the obtained graph structure, we can infer the existence of further object instances in the scene. This is shown in the lower row of the figure, where additional red boxes are shown at locations where the CRF predicts an object instance. Note that this prediction is based only on the structure of the regular pattern and that it returns no object occurrence at locations where there is a too low local evidence for it.



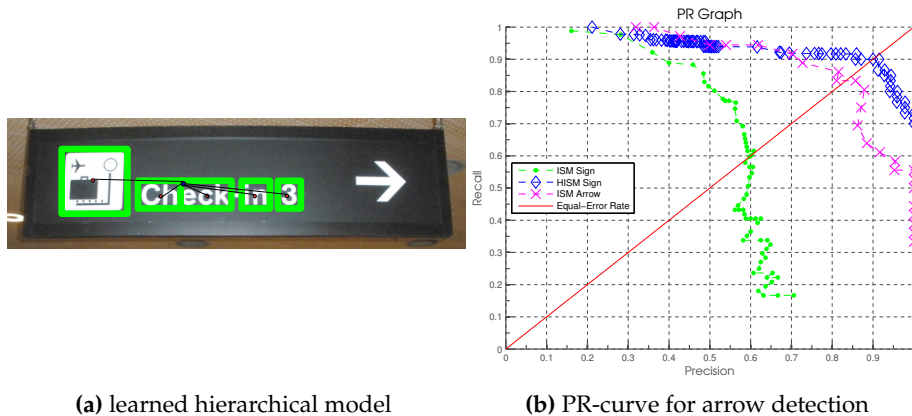
**Figure 5.3.:** Results of the unsupervised object discovery algorithm. The first row shows the discovered object candidates as they are obtained from clustering based on similarities. Note that not all instances of a true class are retrieved due to occlusions or low detection scores. However, using our structure learning approach we can reason about the existence of further instances of each class, as shown in the lower row. Here, the pink dots depict nodes of the CRF, while the yellow lines are the edges. Note how some nodes have not enough local evidence for the existence of an object instance.

## 5.2. Learning from Other Sources by Relating Image with Motion Data

Apart from the information we get from the relations of samples within a given data set, e.g. regular patterns, we can also use other sources of information, for example from different kinds of data. This is sometimes called *self-supervised learning* and particularly relevant in mobile robotics. For example, one can relate the data obtained from a camera or a laser with that of a proprioceptive sensor, such as an inertial measurement unit (IMU) or a localization sensor such as a GPS receiver. This idea is used in Maye et al. [2010] (page 192) to automatically interpret the semantics of direction signs by relating it with an associated motion and finding similarities in the observed data. The idea of this approach is to first learn a model for direction signs, which consist of text strings and arrows, and then find similar models in observations that are all related to a similar motion. This way, the algorithm finds for all images related to, say, a right turn, the similar elements, which turn out to be an arrow to the right. We will give more details and examples in the following.

### 5.2.1. A Model for Direction Signs

The first step of our algorithm is to learn a model for a direction sign without supervision. To do this, it first performs a low-level segmentation and computes descriptors for each segment. Then it collects the descriptors along with displacement vectors towards the



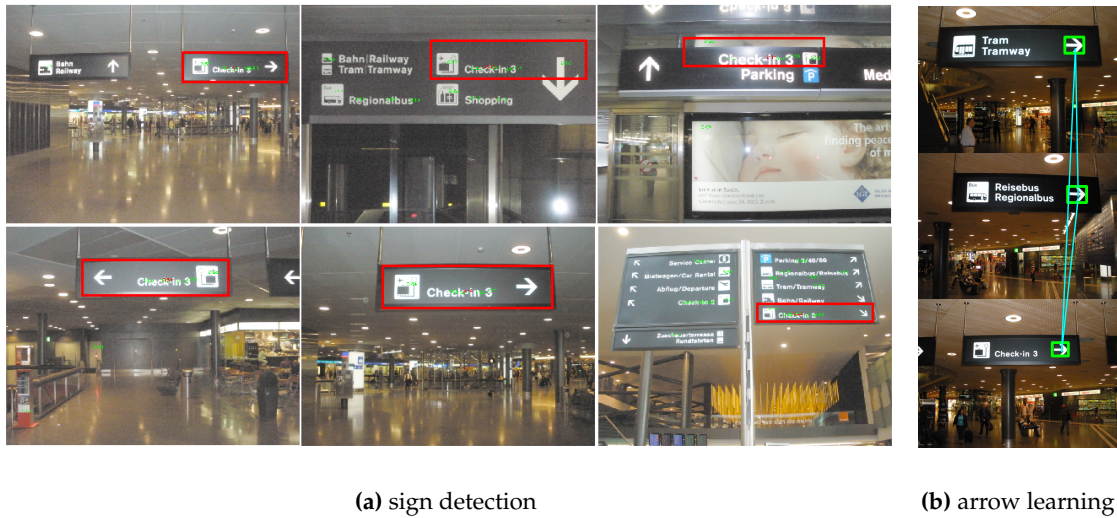
**Figure 5.4.:** (a) From a given input sign (here “Check-in 3”) we build a hierarchical model of parts called tokens (green boxes). Each token consists of a segment with descriptors and displacements pointing to the token center. In addition, all tokens and their associated displacements to the sign center (black lines) are collected in the model. (b) Precision-recall graph for our sign and arrow detection method. The hierarchical approach (HISM, blue curve) is significantly better than the standard ISM method (green curve).

segment center in a codebook in the same way as is done in the ISM approach (see Leibe et al. [2005]). The segment and the codebook together are denoted a *token*. These tokens are later used for matching with other signs, but they are too local and do not contain enough information. Therefore, we use a second codebook of tokens which consists of displacement vectors between a token center and the center of the sign. An example of this is shown in Fig. 5.4a. The benefit of this hierarchical approach is that it is more expressive because the model contains more global information. In addition to this, each token has an assigned weight, which resembles its distinctiveness within a larger set of signs. These weights are learned from occurrence frequencies in a given data set. For example, in a set where signs of Check-in 1 through Check-in 3 are present, the token corresponding to the last digit is very distinctive and therefore receives a higher weight.

### 5.2.2. Sign Detection and Arrow Learning

Using our hierarchical model for direction signs, new images can be used to find occurrences of a given target sign  $S$ . For example, in a mobile robot application the user can present the sign  $S$  to the robot, which then matches  $S$  with all newly observed images. This matching is done hierarchically: first descriptors are computed for a new image  $\mathcal{I}$ . Then, the descriptors are matched against those in  $S$  and votes are computed to find the occurrence of matching tokens, just as in standard ISM. Finally, the tokens themselves cast votes for the occurrence of the sign. An example result of this technique is shown in Fig. 5.5a, where the Check-in 3 sign of Fig. 5.4a is detected in six different images.

Using this sign detector, we can now formulate a self-supervised learning method to infer the semantics of distinctive parts of the signs. We do this by assigning one or more motion directions to each input image. These motion directions can for example be obtained from proprioceptive sensors while a robot observes an image and performs a given



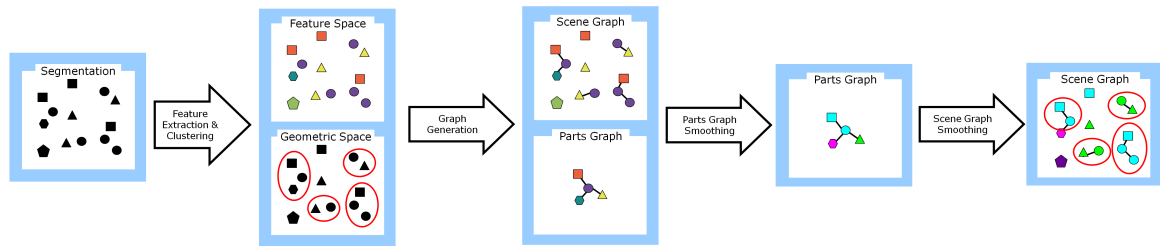
**Figure 5.5.:** Unsupervised sign detection and arrow learning. (a) six examples of detected matches between the sign shown in Fig. 5.4a and new input images. (b)

motion at the same time. Thus, information is not given by human hand-labeling but by steering the robot around within its environment. We note that, although this was not explicitly done for the evaluations in this section, a concrete application to such a self-supervised learning problem is straightforward, and we show a more explicit example in the next chapter. Here, we show how the information given by the associated motion directions can be used to learn the shape of an arrow. Fig. 5.5b shows an example of three images that are associated with a right turn. By matching these images with the described technique, we can distinguish the parts that match across all images, which in this case is the right arrow.

### 5.2.3. Quantitative Results

To quantify the results of our approach, we used a data set of signs from the airport of Zurich. Example images are already shown in Fig. 5.5. We hand-labeled these images to obtain ground truth information for evaluation. Then, we matched the `Check-in 3` sign with all 124 images of the data base where 57 actually contained the `Check-in 3` sign. The resulting precision-recall curve is shown in Fig. 5.4b. As we can see the hierarchical approach performs much better than the standard ISM method, reaching an equal error rate of 90% as opposed to 60.25% for the ISM. For arrow detection, we matched 23 images with an associated right turn to each other. Then, the resulting common token was matched to all images from the data base where 35 actually contained right arrows. The resulting PR-curve in Fig. 5.4b shows that the approach is fairly robust, reaching an equal error rate of 83.33%.





**Figure 5.6.:** Flowchart of our unsupervised object discovery algorithm. First, the input data is segmented and 3D features are computed for each segment. Then, the segments are clustered in feature space and in geometrical space. Next, the likelihood of correspondence to the same class is modeled using two graph structures, one connects physical neighbors (the scene graph) and one clusters of features (the parts graph). Probabilistic inference is then run on both graphs to find part and object labels for each segment.

### 5.3. Learning from Similarity and Physical Closeness

In this section, we return again to unsupervised learning methods that rely only on the data obtained from exteroceptive sensors such as cameras or laser scanners. In Sec. 5.1, we already saw how regular structures in the data can be used to group segments into clusters, and how these clusters can give evidence for the existence of objects of a given class. Here, we will handle the more general case where similar object instances appear within the same scene, but they are not aligned according to any regular pattern. Thus, we can not infer the existence or the location of other objects of the same kind by extrapolating a pattern. However, we can reason on the correspondence of data segments to clusters representing similar objects. Furthermore, in addition to the similarity, we will also use the physical closeness of segments to each other to find meaningful cluster assignments for a given data segment. The latter is used in a part-based approach where segments actually correspond to object parts rather than an entire object. We mainly present concepts and results from Triebel et al. [2010] on page 204 and Shin et al. [2011] on page 212, while some ideas are also taken from Shin et al. [2010] (page 198).

#### 5.3.1. Clustering by Hierarchical Probabilistic Inference

Our aim is to derive a part-based algorithm to discover instances of object classes from 3D point cloud data. Conceptually, this can be stated as the problem of *counting unknown objects*, i.e. that despite the fact that we have no semantic labels given for the observed data, we can find evidence of the existence of an object class by associating instances with each other. We do this not only based on their similarities, but also using the occurrence of similar *constellations* of object instances, which leads to a reasoning on a higher semantic level.

A flowchart of our algorithm is given in Fig. 5.6. Given an input 3D point cloud, the first step is a low-level segmentation based on the superpixel algorithm by Felzenszwalb and Huttenlocher [2004], where we use the dot product of the normal vectors of two 3D points (sometimes denoted the cosine distance) as a similarity criterion. Then, for each resulting segment we compute a set of 3D features, in particular these are spin images [Johnson, 1997], shape distributions [Osada et al., 2002], and shape factors [Westin et al., 1997]. In

---

**Algorithm 3:** Segmentation and smoothed clustering in geometric and feature space (SSCGF).

---

**Data:** Point Cloud  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$

**Input:**

- Segmentation parameters  $\kappa$  and  $\tau$
- Cluster parameters  $\vartheta_f$  and  $\vartheta_g$

**Output:**

- low-level segmentation  $\mathcal{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_M\}, \mathbf{s}_i \subset \mathcal{P}$
- feature-space clusters of segments  $\mathcal{F}_1, \dots, \mathcal{F}_C$
- geometric clusters of segments  $\mathcal{G}_1, \dots, \mathcal{G}_K$
- class label distributions  $\mathbf{d}_1, \dots, \mathbf{d}_M, \mathbf{d}_i \in [0, 1]^K$

$\mathcal{S} \leftarrow \text{SuperPixelSegmentation}(\mathcal{P}, \kappa, \tau)$   
 $\mathbf{f}_1, \dots, \mathbf{f}_M \leftarrow \text{FeatureExtraction}(\mathcal{S})$   
 $\mathcal{F}_1, \dots, \mathcal{F}_C \leftarrow \text{ClusterInFeatureSpace}(\mathcal{S}, \vartheta_f, \{\mathbf{f}_i\})$   
 $\mathcal{G}_1, \dots, \mathcal{G}_K \leftarrow \text{ClusterInGeometricSpace}(\mathcal{S}, \vartheta_g)$   
 $\mathfrak{P} \leftarrow \text{MakePartsGraph}(\{\mathcal{F}_i\}, \{\mathcal{G}_i\}, \mathcal{S})$   
 $\mathfrak{P} \leftarrow \text{SmoothPartsGraph}(\mathfrak{P}, K)$   
 $\mathfrak{S} \leftarrow \text{MakeSceneGraph}(\mathfrak{P}, \{\mathcal{G}_i\}, \mathcal{S})$   
 $\mathfrak{S} \leftarrow \text{SmoothSceneGraph}(\mathfrak{S}, K)$   
 $\mathbf{d}_1, \dots, \mathbf{d}_M \leftarrow \text{ReadFromGraphNode}(\mathfrak{S})$

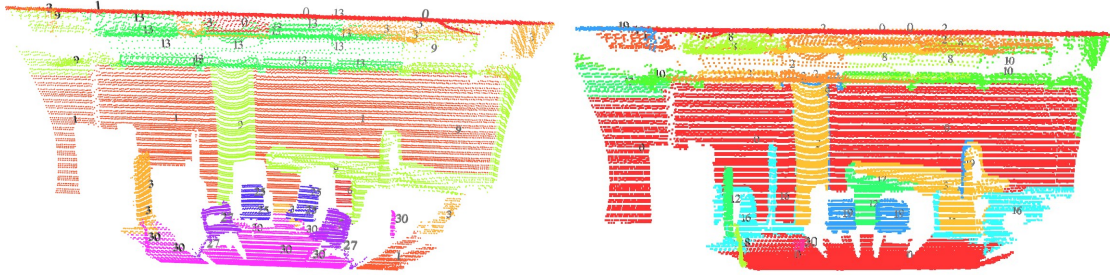
---

the next step, we cluster the segments both in feature space and in the geometric space, where for the former we use affinity propagation [Frey and Dueck, 2007] and for the latter a nearest-neighbor approach with a distance threshold  $\vartheta_g$ . Next, we create two graph structures, one is denoted the scene graph connecting physically close segments with each other, while the other is the parts graph, which draws edges between feature space clusters. Both graphs are explained in more detail in the next section. Then, probabilistic reasoning is done on the parts represented by feature clusters and on the actual part instances in the geometric space. The result is a distribution of cluster labels for each segment, from which the most likely part labels and object labels can be obtained. Note that both the part labels and the object labels consist of natural numbers rather than human-like annotations such as “chair” or “table”, as we do not have such annotations available in the data.

A more formal description of our method, which also denotes the required parameters explicitly, is shown in Alg. 3. Note that the number of segments  $M$ , as well as the numbers  $C$  and  $K$  of clusters are computed inside the particular subroutines.

### 5.3.2. The Scene Graph and the Parts Graph

The key component of our method are two probabilistic graph structures, and we refer to them as the scene graph  $\mathfrak{S}$  and the parts graph  $\mathfrak{P}$ . While  $\mathfrak{S}$  is used to model dependencies between physically close segments,  $\mathfrak{P}$  expresses a general dependency of *types* of segments. Concretely, the nodes of  $\mathfrak{P}$  are defined by clusters  $\mathcal{F}$  of segments in feature space, and two nodes  $\mathcal{F}_i$  and  $\mathcal{F}_j$  are connected whenever there exists a pair of segments  $(\mathbf{s}_k, \mathbf{s}_l)$  so that  $\mathbf{s}_k \in \mathcal{F}_i$  and  $\mathbf{s}_l \in \mathcal{F}_j$ . This means that each such pair raises the evidence that



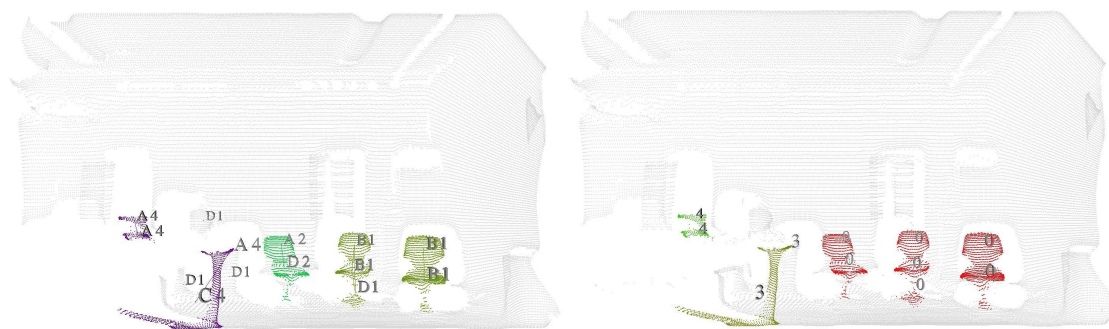
**Figure 5.7.:** Two example scenes observed with a “nodding SICK” laser scanner. Objects that are discovered by the algorithm are colored, where points that correspond to the same object class are assigned to the same color. In addition, the discovered class label identifiers are also shown.

there exists an object class, in which these two types of potential object parts are close to each other. We express this in the edge feature function of  $\mathfrak{B}$ , which yields a high potential if there are many such pairs, while the node features model the relation between the centers of the clusters  $\mathcal{F}_i$  and their potential class labels  $y_i$  (see Triebel et al. [2010] on page 209 for more details).

Once inference is done in the parts graph, the result is used to define the node potentials in the scene graph. As for the parts graph, the edge potentials are designed so that a common label for two connected nodes is preferred over different labels. The motivation behind this two-step inference procedure is that the parts graph only returns general label distributions for all segments of a given type, but the scene graph also takes the local geometric neighborhood into account. Expressing this with a concrete example, one can say that the parts graph reasons that, say chair legs and chair seats are often observed together, which raises the evidence that they have the same class label, however when considering a concrete instance of a chair leg, it could be observed very close to a table top, which increases the probability that it is actually a leg of a table.

### 5.3.3. Results

We tested the algorithm on data acquired from real-world scenes using a nodding SICK laser scanner with a horizontal opening angle of 100 degrees and a nodding range of 90 degrees. Each set was captured at a horizontal resolution of 0.25 degrees and a vertical resolution of 0.2 degrees. We evaluated 50 data sets from four different rooms, each room containing some number of chairs, trash cans, flip charts, plants, etc. Objects were placed up to 90 degrees of rotation from each other. Most scenes contained two or three objects of the same type, but some scenes contained up to four objects of three different kinds. Fig. 5.7 shows some results of our object discovery algorithm. Points that belong to the same object have the same color, and the numbers represent the class label assigned to each segment. For instance, the scene on the left contains four chairs of two different kinds, and they are correctly labeled as 25 (blue) and 27 (violet). In a quantitative analysis we computed the conditional entropy [Tuytelaars et al., 2009] of a hand labeled “ground truth” and the class labels that resulted from the discovery algorithm. We obtained values around 1, which is very good compared to similar algorithms described by Tuytelaars et al. [2009].



**Figure 5.8.:** Object categorization across different scenes. **Left:** Result without using categorization. The object discovery returns a different class for the left chair than for the other two chairs. **Right:** Result using categorization. Here, a reasoning across different scenes helps to associate all three chairs with each other and to assign them the same class label.

### 5.3.4. Object Categorization Across Different Scenes

Despite its ability to discover object classes without supervision, the algorithm described in the previous section has two disadvantages: First, it considers all segments as potential object parts resulting in many false neighborhood connections between foreground and background segments. This results in object candidates composed of real object parts and background parts. And second, it can not associate two instances of the same class with each other when they appear in different scenes. The reason for this is that, as the discovery process is unsupervised, the resulting local class labels are not unique over a given number of scenes. This means that an object class might be associated with a class label  $G_1$  when one scene is observed, but the same object class might have a different class label  $G_2$  if observed in a different scene. Therefore, in Shin et al. [2011] (page 212) we extend the approach in two aspects: First, we employ a saliency-based foreground extraction algorithm, and second we learn *object categories*, i.e., object classes that are consistent across a sequence of input scenes. To do this, we need to solve the data association problem, which we achieve by introducing a third level of reasoning named the *class graph*. The key idea behind the class graph is to find a mapping from local class labels to global category labels. Unlike the parts graph and the scene graph, the class graph models the statistical dependencies between labels of object class instances rather than object parts. Its nodes correspond of discovered object classes and the edges connect classes that have a similar appearance. In the same way as above, we use probabilistic inference to smooth the graph, and we obtain a most likely assignment of class labels across different scenes. An example is shown in Fig. 5.8, where the left image shows the result of object discovery without the categorization step. We see that the algorithm incorrectly assigns one of the three chairs to a different class. However, by reasoning across various scenes using our categorization step, all three chairs are correctly associated with the same class label.

## 6. Unsupervised Online Learning

In the previous chapter, we have seen how additional sources of information such as similarities or the structure of regular patterns can be used to infer the existence and the location of instances of object classes. This was done completely unsupervised, i.e. there was no hand-labeled training data set given. In this chapter, we elaborate on this idea even further, but now we additionally investigate approaches that can update the learned representation *before* all input data has been observed. Concretely, this means that the underlying model learned by the algorithm is updated and refined whenever a new set of observations is available. This is often denoted as *online learning*, which stands in contrast to the standard offline learning approach where no model updates are done once the learning step has finished. Online learning is particularly useful when learning should be adaptive, i.e. it can handle new, unobserved situations. However, this benefit also comes with the drawback that online methods can never be better than their offline counterparts, because they have to rely on a smaller subset of the data. We will see an example of this with an experimental comparison in Sec. 6.2. Before, we present a self-supervised online learning method in Sec. 6.1, where the information used for learning comes from a proprioceptive sensor in a robotic car. Sec. 6.1.4 shows an online learning approach to find dynamic objects in 3D range scans.

### 6.1. Self-supervised Online Learning of Driving Behaviours

In this section, we consider again the application of a perception system for autonomous cars. However, in contrast to Chapter 3, where we investigated supervised learning methods to classify pedestrians and cars, we will now discuss an algorithm that is able to perform online learning from an additional data source instead of human inputs. Concretely, our input data consists of an image sequence from a camera that was mounted inside a car, and a stream of acceleration measurements acquired from an inertial measurement unit (IMU), which produces data that is synchronized with the camera. The aim now is to segment the data stream online and to learn a correlation between the traffic situation observed with the camera and the corresponding vehicle motion in terms of IMU measurements for each such segment. For example, the system should then be able to relate a deceleration measured by the IMU to a red traffic light observed with the camera, and then predict such a deceleration the next time a red light is detected. The two major benefits of this learning method over those given in Chapter 3 are that we do not need any human interaction for learning and that the algorithm is completely adaptive to new situations. All details of the method are presented in Maye et al. [2011] on page 243, here we explain the key ideas.

### 6.1.1. A Bayesian Formulation of the Problem

As mentioned, for a given sequence of camera images and associated acceleration measurements from an IMU sensor, our aim is to automatically segment the data into motion segments, find a label that identifies the traffic situation of each segment, and predict a motion based on the currently inferred traffic situation. To do this, we use a Bayesian filter approach that estimates all model parameters unsupervised and online, i.e. each new observation refines the model and yields a new prediction for the next time step. In our formulation we use the following random variables:  $r_t$  represents the length of the current motion segment at time  $t$ ;  $l_t$  is the label of the traffic situation at time  $t$ ;  $\mathbf{a}_t$  is the predicted action at time  $t$ ;  $\mathbf{z}_{1:t}$  are the IMU measurements up to time  $t$  and  $\mathbf{c}_{1:t}$  the camera measurements up to time  $t$ . With this, we state the problem as an estimation of the joint distribution over  $r_t$ ,  $l_t$ , and  $\mathbf{a}_t$  given the data up to time  $t$ , which can be factorized as

$$p(r_t, l_t, \mathbf{a}_t \mid \mathbf{z}_{1:t}, \mathbf{c}_{1:t}) = p(r_t \mid \mathbf{z}_{1:t})p(l_t \mid r_t, \mathbf{c}_{1:t})p(\mathbf{a}_t \mid r_t, l_t, \mathbf{z}_{1:t}). \quad (6.1)$$

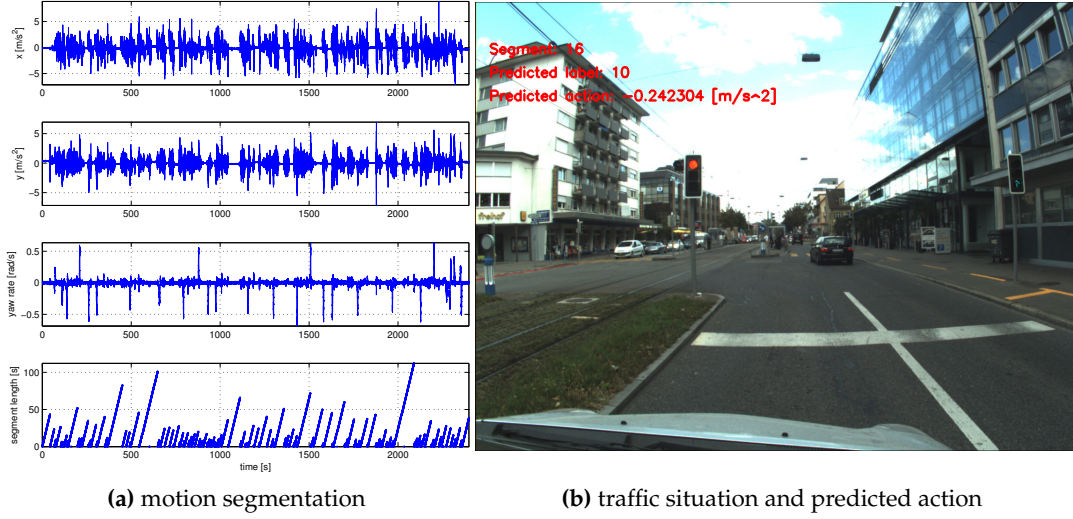
Here, we made the reasonable assumption that the segment length  $r_t$  and the action  $\mathbf{a}_t$  are conditionally independent of  $\mathbf{c}_{1:t}$  given  $\mathbf{z}_{1:t}$ , and that the scene label  $l_t$  is conditionally independent of  $\mathbf{z}_{1:t}$  given  $\mathbf{c}_{1:t}$ . Thus, we have divided our problem into the estimation of the three factors in Eq. (6.1), which can be described as follows:  $p(r_t \mid \mathbf{z}_{1:t})$  corresponds to the model of the motion segmentation,  $p(l_t \mid r_t, \mathbf{c}_{1:t})$  to the traffic situation model, and  $p(\mathbf{a}_t \mid r_t, l_t, \mathbf{z}_{1:t})$  to the action prediction model. The estimation of the latter two will be described in the next section. For the motion segmentation model  $p(r_t \mid \mathbf{z}_{1:t})$  we note that the joint  $p(r_t, \mathbf{z}_{1:t})$  can be factorized as

$$p(r_t, \mathbf{z}_{1:t}) = \sum_{r_{t-1}} p(r_t \mid r_{t-1})p(\mathbf{z}_t \mid r_{t-1}, \mathbf{z}_{1:t-1})p(r_{t-1}, \mathbf{z}_{1:t-1}), \quad (6.2)$$

which consists of a transition probability  $p(r_t \mid r_{t-1})$ , the *predictive distribution* given by  $p(\mathbf{z}_t \mid r_{t-1}, \mathbf{z}_{1:t-1})$ , and the posterior  $p(r_{t-1}, \mathbf{z}_{1:t-1})$  from the previous time step. While the transition probability can be set to a simple binary distribution, the predictive distribution for the new measurements  $\mathbf{z}_t$  is modeled as a multivariate Gaussian, whose parameters are marginalized out using the conjugate Normal-Wishart prior. Furthermore, to overcome the large computational cost that results from storing and computing  $p(r_t \mid \mathbf{z}_{1:t})$  for  $t$  and all previous time steps, we use a particle filter in the implementation.

### 6.1.2. The Traffic Situation Model and the Action Model

The discrete labels  $l_t$  in Eq. (6.1) correspond in our formulation to indices into a range  $\{1, 2, \dots, N\}$  of potential traffic situations. These traffic situations are modeled as generative models  $p(\mathbf{c}_t \mid \boldsymbol{\eta}_i)$ , which for given parameters  $\boldsymbol{\eta}_i$  return a distribution over potential camera measurements. Here, a camera measurement is represented using a bag-of-words approach where descriptors extracted from the images are clustered into  $K$  clusters and a histogram is computed that counts the number of descriptors per cluster. Thus, the model  $p(\mathbf{c}_t \mid \boldsymbol{\eta}_i)$  can be described as a multinomial distribution, and we use again a conjugate prior, which in this case is the Dirichlet distribution with hyperparameters  $\psi_i$ , to marginalize over the model parameters  $\boldsymbol{\eta}_i$ .



**Figure 6.1.:** Self-supervised online learning of driving behaviours. (a) Result of our motion segmentation algorithm. (b) Example result of a discovered traffic situation with a predicted action. Here, the system correctly predicts a braking maneuver.

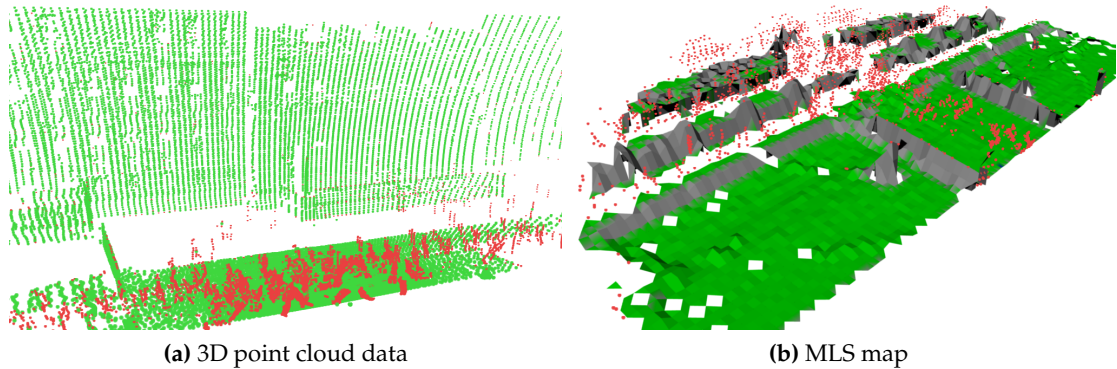
To estimate the number  $N$  of potential traffic situations found in the images we use Bayesian hypothesis testing, which means that we compute the Bayes factor  $B_i$

$$B_i = \frac{p(\mathbf{c}_t \mid l_t = i, \psi_i)}{p(\mathbf{c}_t \mid l_t, r_t, \mathbf{c}_{t-1}, \psi^{r_{t-1}})} \quad (6.3)$$

for each of the  $N$  existing models, where  $\psi^{r_{t-1}}$  are the hyperparameters learned over the current segment. If  $B_i$  is below a threshold  $\xi$  for all models  $i$ , we add a new model with initial hyperparameters. Thus, to summarize we compute the likelihood  $p(\mathbf{c}_t \mid l_t, r_t, \mathbf{c}_{t-1})$  of the current label  $l_t$  and obtain an estimate of the new label distribution using Bayes' law:

$$p(l_t \mid r_t, \mathbf{c}_{1:t}) \propto p(\mathbf{c}_t \mid l_t, r_t, \mathbf{c}_{t-1})p(l_t \mid r_t, \mathbf{c}_{1:t-1}). \quad (6.4)$$

The remaining term in Eq. (6.1) is the action model  $p(\mathbf{a}_t \mid r_t, l_t, \mathbf{z}_{1:t})$ . Here, we proceed in a similar way as before. To each traffic situation model  $M_i$  we associate an action model  $A_i$ , which is represented as a Gaussian Mixture Model (GMM). This enables us to assign different possible actions to the same traffic situation. For instance, when a driver reaches a traffic light, they might brake even when the light is green, e.g. due to a jam. To estimate the number of Gaussian components we again compute the Bayes factor and introduce a new Gaussian when all old Gaussians are too unlikely. Also, as above we use the conjugate prior of the Gaussians to marginalize out the parameters and compute the hyperparameters using online updates. In the implementation we attach both the action and the traffic situation model to each particle representing a hypothesis over the current segment length.



**Figure 6.2.:** Online estimation of dynamic objects. (a) Point cloud data produced by an up- and down sweeping 2D laser. The red dots correspond to the dynamic objects that are estimated online by the algorithm. (b) A different scenario, where the data is represented using an MLS map [Triebel et al., 2006]

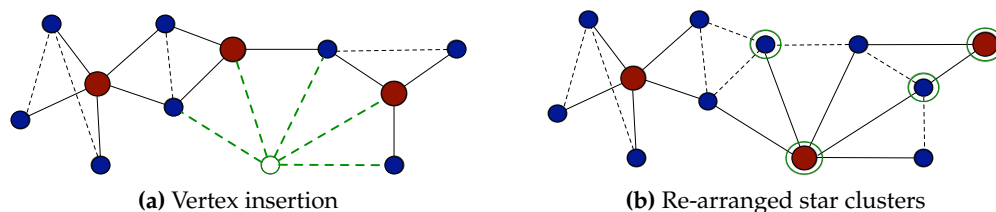
### 6.1.3. Results

To evaluate the approach, we collected a dataset with a car in an urban setting. Our car is equipped with a Sony XCD-SX910 camera recording 1280x960 images at 3.75 frames per second and an XSens MTi IMU running at 100 Hz with  $x$  pointing forward,  $y$  to the right, and  $z$  upward. The sequence contains 8218 images and is about 40 minutes long. We encountered different scenes comprising of traffic lights, crosswalks, or changes of speed limit. The car was driven in a loop so that it passes several times through the same situation, which gives a better estimation of the quality of our solution.

We estimated the quality of our motion segmentation algorithm from Section 6.1.1 on the recorded data and performed inference on the final posterior distribution (6.2) to get the optimal sequence of segment lengths representing our motion segments. We set the hazard rate to  $\lambda = 1/10$ , the number of particles to  $P = 100$ , and the prior hyperparameters of the normal-Wishart to  $\kappa_0 = 1$ ,  $\rho_0 = \mathbf{0}$ ,  $\nu_0 = 3$ ,  $\Lambda_0 = \mathbf{I}$ . We only considered IMU data at 10 Hz. Fig. 6.1a shows the extracted motion segments along with the corresponding IMU data. Our algorithm identified 165 segments which are validated by visual inspection of the IMU data. Furthermore, the segmentation has been compared to a manual annotation of our image sequence and exhibited an accuracy of approximately 92%. For the ground truth labeling of the change-points, we watched the video and noted down where a human would expect a change of driving behavior.

To evaluate the traffic situation labeling we created a ground truth by manually annotating the image sequence. Then, we learned traffic situation models with our approach from an explicitly selected subset of images containing traffic lights, yield signs and pedestrian crossings. We obtained an accuracy of 93% for traffic light scenes, 99% for yield scenes, and 91% for pedestrian crossings scenes. An example for a qualitative result is shown in Fig. 6.1b. Here, the currently estimated traffic situation label and the predicted action are visualized inside the current image from the video sequence. As we can see, the situation is a red traffic light, and the algorithm correctly predicts a negative acceleration, which corresponds to a braking maneuver.





**Figure 6.3.:** Insertion of a data point with star clustering. (a) The new data point may introduce additional links in the similarity graph (green) affecting adjacency and hence the validity of the current minimal star cover. (b) Inconsistent stars are re-arranged (green circles). The number of broken stars largely determine the running time. On real graphs, the avg. number of stars broken is usually small (experimentally verified) yielding an efficient incremental approach.

#### 6.1.4. Online Estimation of Dynamic Objects

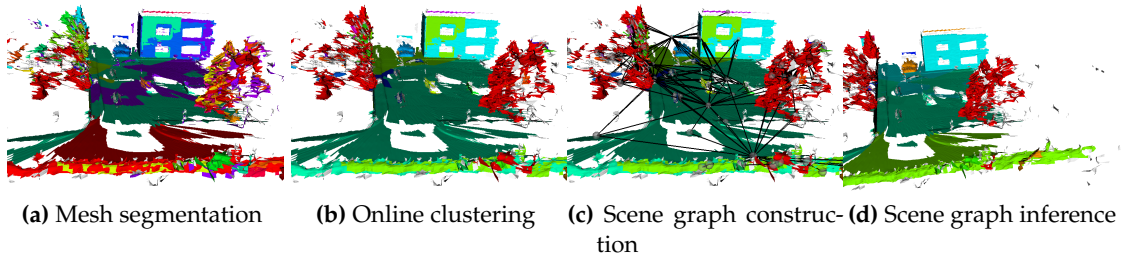
Another example of an unsupervised online learning framework is presented in Kästner et al. [2010] on page 228. Here, a 2D laser scanner is used to collect 3D point cloud data while moving up and down, and at the same time measurements corresponding to dynamic objects are identified and their model is incrementally updated. Again, we use a Bayesian filter approach which estimates the parameters of a Gaussian Mixture Model for each laser measurement online. Each mixture component has an associated likelihood of corresponding to a dynamic object, which is also estimated online. Two example results of this method are shown in Fig. 6.2.

## 6.2. Unsupervised Online Segmentation of 3D Scenes

In Sec. 5.3 we presented an unsupervised learning algorithm for segmenting 3D scenes into meaningful clusters. We saw that by exploiting similarities and physical closeness, a reasonable labeling of a 3D point cloud can be achieved. However, the method presented there was an offline algorithm, i.e. for every new observed 3D scene, the low-level segmentation, the underlying graph structure and the similarities between the segments have to be computed again. This leads to a significant overhead in computational resources. Particularly if we are concerned with a mobile robot platform, new observations are often not significantly different from the observations made previously. Therefore, we investigate in this section an online variant of the method presented in Sec. 5.3. The details of this online method are given in Triebel et al. [2012] on page 249, here we will describe the major ideas.

### 6.2.1. Online Clustering in Feature Space

Our goal is to modify the SSCGF algorithm (see Algorithm 3 on page 50) in such a way that it can perform the inference step incrementally, i.e. without having to re-compute the structure and the marginals of the CRF. First, we simplify the problem by only considering objects that do not consist of multiple parts. This makes the use of a parts graph unnecessary. Then, we observe that there are two main components in the algorithm: the clustering step, and the smoothing step, and we need to find online variants for both of them. For



**Figure 6.4.:** Key steps of the processing pipeline. (a) Result after segmenting the triangle mesh. Each color represents a different segment. (b) Result after online clustering in feature space. Each color represents a different feature cluster. (c) Construction of the scene graph. Nodes are centers of oriented bounding boxes (OBBs) around each segment. Edges connect segments with overlapping OBBs. (d) Result after inference in the scene graph. The class label distribution is smoother compared to (b), as can be seen, e.g., in the upper left corner of the building.

the clustering step, we use the star clustering algorithm by Aslam et al. [2004], because it is particularly made for online computation. Clustering is performed by building a similarity graph and determining star and satellite nodes so that the data is represented by a minimal number of maximal star-shaped subgraphs (“min-max condition”). An example of this is shown in Fig. 6.3a. Then, for a new data point the similarities to the existing nodes are computed and the star and satellite nodes are rearranged so that the min-max condition is again valid. In the worst case, all nodes need to be rearranged, but this occurs in practice only very rarely. Most often, only a few nodes change from center to satellite nodes.

Thus, in our application, we first compute a low-level segmentation of a new 3D mesh (see Fig. 6.4a), extract features as was done in offline SSCGF, and insert all new segments according to their similarities into the existing star graph, which results in a new clustering (see Fig. 6.4b). Then, after rearranging, we re-compute the node potentials of the scene graph for those nodes that were involved in the changed star clusters. Next, we insert the new segments also into the scene graph and connect them with already existing segments that are close enough (Fig. 6.4c). Finally, we re-run loopy belief propagation, but only for those messages that are directly influenced by the new nodes and the changed potentials (Fig. 6.4d). This is described next.

### 6.2.2. Online CRF Inference with Incremental Belief Updates

If we used standard loopy belief propagation (LBP) for the inference (see Sec. 2.3.3 on page 15), this would require a re-initialization of all messages every time a new scan is observed. Thus, the number of message updates grows at least linearly with the number of totally observed mesh segments. To avoid this, we perform the message update *online*, i.e. we only update messages that are affected by a change in the cluster graph  $\mathcal{G}$  and the messages that depend on them. First, we note that in the CRF, nodes are never removed, and a change in  $\mathcal{G}$  can affect nodes from earlier points in time. Thus, we need to provide two kinds of update operations: inserting a new node into the CRF, and changing the feature function of an existing node. In the first one, new messages are added, in the second,

existing messages need to be recomputed, which is essentially the same as removing the old message and adding a new one. The major problem here is however, that a newly inserted and initialised message has maximal entropy and can not propagate the same amount of information as the existing messages obtained after LBP convergence earlier. This leads to an "over-voting" of the potentials of the new nodes from the existing nodes.

To overcome this problem, we store all messages computed in each LBP iteration in a *message history*  $\mathbf{m}_{ij} = m_{ij}^{(1)}, m_{ij}^{(2)}, \dots$ . Then, before computing (2.18), we determine the minimal history length  $\mu$  of all message histories  $\mathbf{m}_{ki}$  where  $k \in \mathcal{N}(i) \setminus j$ , and the max-sum rule (2.18) turns into

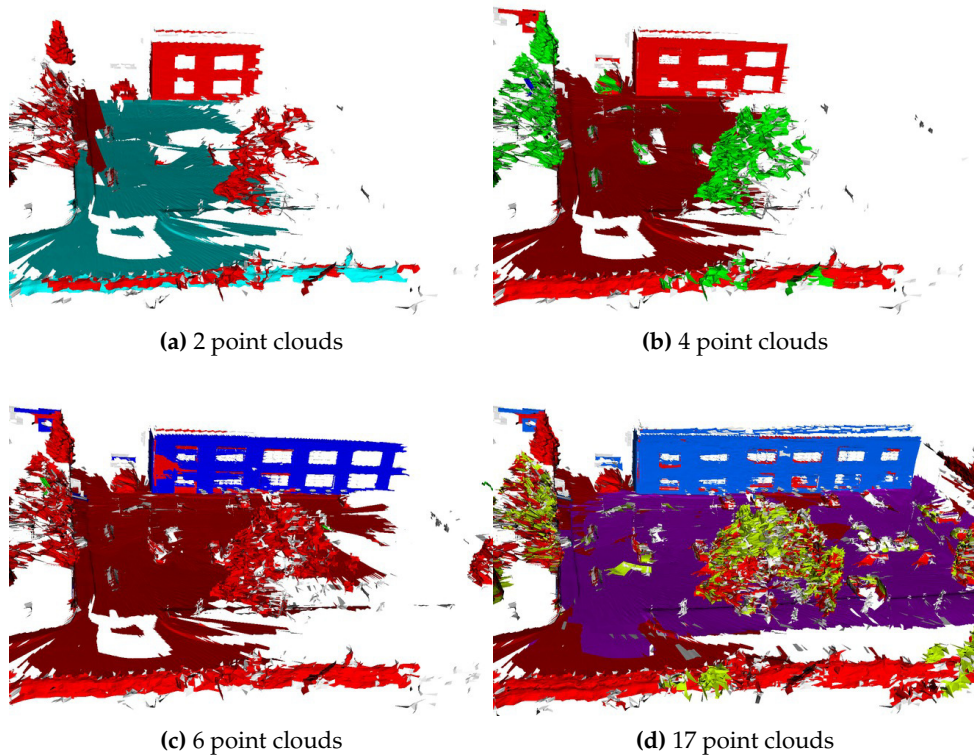
$$m_{ij}^{(\mu+1)}(l_j) \leftarrow \max_{l_i} \log \varphi_i + \log \psi_{ij} + \sum_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{(\mu)}(l_i). \quad (6.5)$$

Some care has to be taken here: to avoid inconsistencies, all messages in the history  $\mathbf{m}_{ij}$  later than  $\mu$  need to be removed. Also, all message histories that depend on  $\mathbf{m}_{ij}$  need to be updated as well. However, the amount of change caused by these updates decreases with every set of successor messages to be updated. To avoid an entire update of all message histories, we determine a threshold  $\epsilon$  and stop updating message histories when the change drops below  $\epsilon$ . Note that this is different from the threshold  $\xi$  for LBP convergence: while  $\epsilon$  determines the number of messages updated after an online update – and thus the performance difference between online and offline processing,  $\xi$  influences the amount of smoothing. By changing  $\epsilon$  gradually towards 0, the online LBP algorithm turns into its standard offline version.

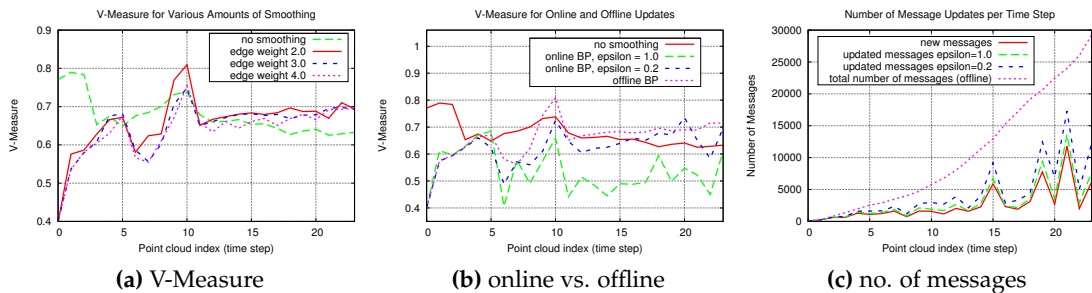
### 6.2.3. Results

For evaluation, we ran experiments on streamed 3D laser range data acquired with an autonomous car. The sensor consists of three SICK LMS-151 laser scanners mounted vertically on a turn table. The rotation frequency was set to  $0.1 Hz$ . We drove the car slowly ( $\approx 15 km/h$ ) around a car park in front of a building. We use qualitative and quantitative measures for evaluation. The qualitative evaluation is done by visualizing the discovery results with different colors for each category (see Fig. 6.5). The quantitative measures are: number of resulting categories, number of update steps, and the entropy-based *v-measure* [Rosenberg and Hirschberg, 2007], which is defined as the harmonic mean of *homogeneity* and *completeness* of the obtained labelling compared to a human-labeled ground-truth.

Fig. 6.6 shows a performance comparison with respect to different edge weight parameters  $w_e$  and online message update thresholds  $\epsilon$ . The left and middle figure show the V-measure compared to a hand-labeled ground truth over time. We can see that the performance increases over time and that the online LBP version for  $\epsilon = 0.2$  is only slightly worse than the offline version. However, as shown in Fig. 6.6(right), there is a significant reduction in the number of updated messages compared to the offline LBP. A smaller  $\epsilon$  improves the V-measure performance, but it also increases the message passing horizon causing more message updates and thus a longer computation time.



**Figure 6.5.:** Results of online scene segmentation. Results are shown after (a) 2, (b) 4, (c) 6, and (d) 17 point clouds. Note that initially only two categories are discovered, and the categorization is incorrect (e.g. the tree and the building are assigned the same label). However, as the algorithm evolves over time, the categorization improves, and the number of classes is increased.



**Figure 6.6.:** Quantitative results. (a) V-Measure compared to ground truth for each time step with different values of  $w_e$  (all offline). In the beginning, smoothing makes the result worse, as the number of clusters is reduced too much. Later, smoothing improves the result. The amount of smoothing has not a strong influence. (b) Comparison between online and offline LBP. With decreasing value of  $\epsilon$ , online performance approaches the offline quality, with some random effects. (c) Number of messages updated in online and offline LBP. The red line shows the number of new messages introduced at each time step, which is the minimum number of necessary updates. A smaller  $\epsilon$  leads to more message updates.

## **Part III.**

# **Active Learning**

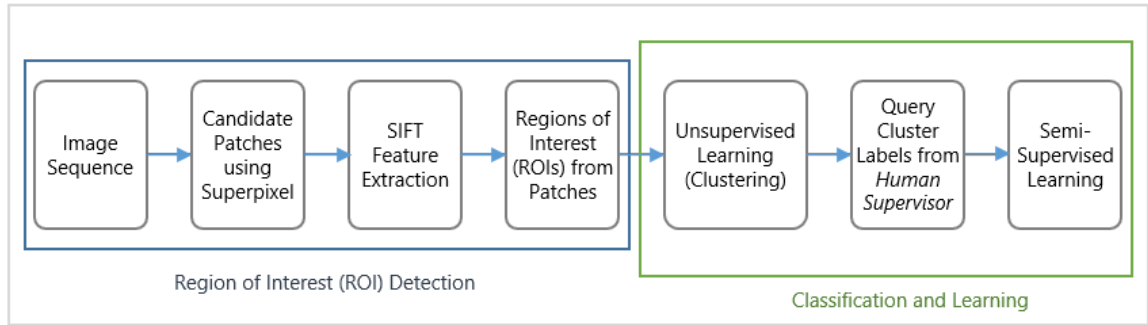


## 7. Active Learning

At this point, we have seen a number of techniques to make the learning process more autonomous by not requiring user interaction and more efficient by formulating online algorithms so that only incremental updates are needed instead of a complete re-computation. However, it is clear that only relying on completely unsupervised learning will not be sufficient to solve problems such as semantic mapping, in which a human-interpretable annotation of a given environment representation is sought. Here, some kind of human interaction is needed, although we still aim for approaches that reduce the number of required user interactions as much as possible. One idea to do this is to leverage the ideas from unsupervised learning and combine them with supervised learning methods, so that the human user is asked to give one label only for a whole sub-class of similar instances instead of one label for every instance. This idea is closely related to *semi-supervised* learning [Chapelle et al., 2006], and we will give a proof-of-concept example in Sec. 7.1. However, as mentioned in the introduction (see Fig 1.1), a thorough investigation of such combined techniques, particularly if they are supposed to be online methods, is out of the scope of this thesis. Instead, we try to do an intermediate step by focussing on supervised online learning methods which require less user interactions by actively selecting data points, from which learning can be achieved better. This concept is called *Active Learning* and is motivated by the insight that, given sufficient initial data, the learner itself should have a model that allows to distinguish between new data points, that could help to improve further predictions and those that are likely not to improve them. We give a more detailed explanation of Active Learning in Sec. 7.2, and we propose a concrete Active Learning algorithm for a semantic mapping task. Then, in Sec. 7.3 we apply Active Learning for interactive image segmentation, where one property of this particular problem can be used to formulate a very efficient online method. Finally, in Sec. 7.4 we apply the Confidence Boosting algorithm developed in Sec. 4.4 to Active Learning, which leads to an efficient online learning method with very steep learning curves.

### 7.1. Combining Supervised and Unsupervised Learning

As mentioned above, for tasks that require semantic information we have to use some kind of supervision during learning, and one possibility to reduce the required supervision is to use unsupervised learning before user interaction. This idea is used in Debnath et al. [2014] on page 284 to learn a classifier for indoor objects such as chairs and monitors. Concretely, before asking the human supervisor for a semantic label for training, we group the observed data into clusters using unsupervised learning. Then, our algorithm queries one common label for each cluster from the supervisor and uses the so obtained training data in a semi-supervised learning step. This approach has two major advantages: first, it reduces the amount of human intervention significantly by asking labels for multiple in-



**Figure 7.1.:** Flow chart of our system. From a sequence of images, regions of interest are detected using super pixel segmentation and by comparing the segments based on SIFT features. Then the resulting patches are clustered. From each cluster, a subset of patches is used to query object labels from a human supervisor. The resulting hand-labelled data together with some unlabelled samples is then used to train a semi-supervised classifier.

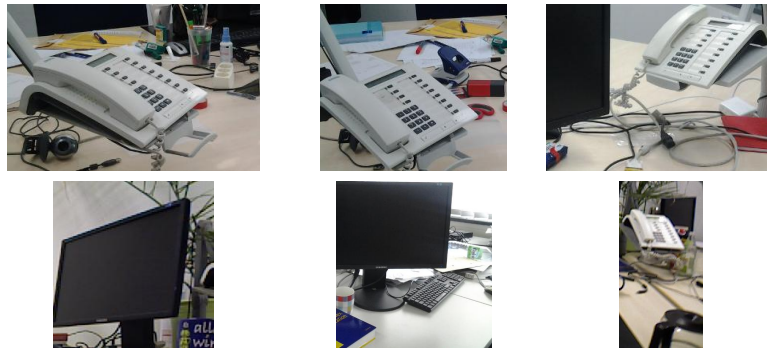
stances at the same time. And second, it gives us the potential to pre-select interesting data to train on, for example by asking labels only for clusters that are significantly represented. More details of the method are given in the following.

### 7.1.1. From Clusters to Labels

A flow chart of our complete semi-supervised learning method is given in Fig. 7.1. We start with a sequence of input images and determine first an appropriate set of rectangular regions of interest named *patches*. Then, we extract SIFT features [Lowe, 1999] and define a similarity measure between patches based on the Euclidean distance between matching descriptors. Based on these similarities, we cluster the patches using spectral clustering. The motivation to apply clustering before querying a class label is two-fold: first, the number of required user interactions, is reduced, because we query only one common label for an entire group of data instances. And second, the clustering step gives us the opportunity to pre-select interesting data to train on, because typically some clusters can be easily identified as more relevant for the learning task based on characteristics such as cluster size or similarities of elements within a cluster. The intuition here is that only those data instances should be learned by the classifier, for which there is enough evidence that they correspond to a meaningful object class. In the next step, we select a subset of appropriate patches from each cluster and query object labels from a human supervisor. Here, we aim to produce only very few label queries on one side, on the other side we need to make sure that the data we provide as training samples to the semi-supervised learning method is as *pure* as possible, i.e. ideally there should be no instances of different objects labelled by the human with the same label. Therefore, we propose to use a quality measure  $q$  for all patches within a cluster, which is based on the similarities  $s$  computed earlier. Concretely, for every patch  $A$  of a given cluster  $C$ , we compute  $q$  as the sum of similarities *within the cluster*:

$$q(A) = \sum_{B \in C} s(A, B). \quad (7.1)$$





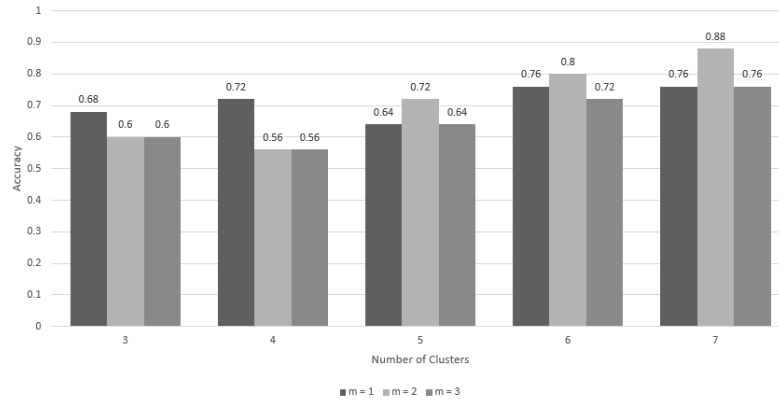
**Figure 7.2.:** Examples of clusters obtained from the clustering algorithm (rows corresponds to different clusters). For both clusters, we show the first three elements according to the quality measure defined in (7.1).

After computing the  $q$ -values, we sort all elements within a cluster in descending order of  $q$  and ask one common label from the user for the first  $m$  such elements of each cluster. This policy gives a good trade-off between the two opposing objectives of generating few label queries and providing pure training data. To illustrate this step, Fig. 7.2 shows an example result of the clustering step, where rows corresponds to different clusters and only the first 3 elements according to the quality measure  $q$  are shown.

Then, the final step in our approach uses the labelled data to learn a classifier for the objects discovered in the environment. For this, we first compute the Hierarchical Matching Pursuit (HMP) descriptor introduced by Bo et al. [2011] on every patch. Then, we performed experiments with three different classifiers: a standard Support Vector Machine (SVM), a nearest-neighbor classifier and a transductive SVM, which is a semi-supervised learning method. Thus, in addition to the labelled training set  $\mathcal{D}$  of size  $l$ , the algorithm is also given an unlabelled set  $\mathcal{D}^* = \{\mathbf{x}_i^* \in \mathbb{R}^p\}_{i=1}^k$  of test examples to be classified. From these three methods the worst in our experiments was the standard supervised SVM, and we did not consider this further. The highest classification performance was obtained with the transductive SVM.

### 7.1.2. Results

In our experiments, we investigated the correspondence of the number of label queries made by the algorithm and the classification accuracy. There are two parameters that can be set: the number of clusters  $c$  and the number  $m$  of patches per cluster, which receive a label after the query (see above). On one side, we want to have few clusters, i.e.  $c$  should be low. However, if there are more clusters, then the clusters are smaller and therefore *purier*, i.e. there are more elements that agree on the true class label. Purer clusters means that we can increase  $m$ , without assigning wrong labels to patches, thus we obtain better training data. This relationship is shown in Fig. 7.3. If the number of clusters is small, we get the best accuracy for  $m = 1$ . But for more clusters,  $m = 2$  is better, because by assigning the same label to the first  $m$  elements of each cluster, we get fewer wrong labels. In general we found that having less labels for training is better than having more, but wrong labels.



**Figure 7.3.:** Accuracy vs. number of clusters and number  $m$  ( $m = 1, 2, 3$ ) of patches receiving a label from the query. More clusters lead to a higher cluster purity. Then, higher values of  $m$  are more effective, because the tSVM receives better training data.

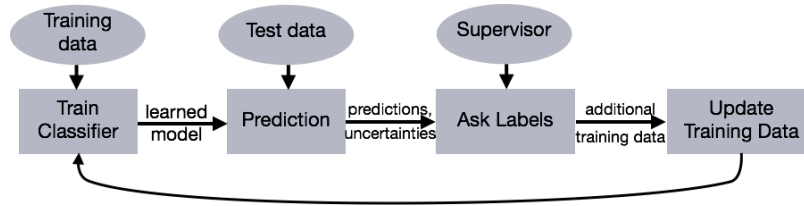
## 7.2. Active Learning for Semantic Mapping

Another strategy to reduce the number of required label queries is to selectively choose those data points, which are expected to give a large amount of information for refining the currently learned model. This idea is called Active Learning, and it stands in contrast to passive learning methods in that it performs learning in cycles of training and testing instead of clearly separating between a learning and a prediction phase. We explain Active Learning more detailed in Sec. 7.2.1 and refer to the survey of Settles [2012] for a deeper analysis. In Sec. 7.2 we then apply this idea to the problem of semantic mapping, where the underlying classifier is a sparse variant of the Gaussian Process Classifier.

### 7.2.1. Active Learning in General

Fig. 7.4 shows a general flow chart of an Active Learning framework. First, we start with an initial training set  $(\mathcal{X}_0, \mathcal{Y}_0)$ , where  $\mathcal{X}_0 = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  are the  $N$  feature vectors and  $\mathcal{Y}_0 = \{y_1, \dots, y_N\}$  are the corresponding class labels. For binary classification we have  $y_i \in \{-1, 1\}$  and for multiple classes we usually define the  $y_i$  as indices into the  $C$  classes. Then we train a classifier with  $(\mathcal{X}_0, \mathcal{Y}_0)$ , which we model as a function  $f : \mathbb{R}^d \rightarrow \mathbb{R}^C$ , i.e. each input feature vector  $\mathbf{x}$  is mapped to a prediction vector  $\mathbf{p} \in \mathbb{R}^C$ . Next, a set of  $K$  data samples  $\{\mathbf{x}_1^*, \dots, \mathbf{x}_K^*\}$  is drawn from the test data set  $\mathcal{X}^*$  and classified using  $f$ . Here, the nature of  $\mathcal{X}^*$  defines the type and complexity of the active learning problem: if  $\mathcal{X}^*$  is given beforehand and its size does not change during learning, then we are concerned with *pool-based* active learning. If, however  $\mathcal{X}^*$  is a potentially infinite stream of data, then we have *stream-based* active learning, which is significantly harder. In the applications presented here, we consider the pool-based variant.

Then, in the classification step we obtain the *label predictions*  $y_1^*, \dots, y_K^*$ , i.e. the class indices for which the classifier returned the highest probability in the prediction vectors  $\mathbf{p} \in \mathbb{R}^C$ . Here, the prediction vectors are important, because they are used to compute the *confidence* of the classifier, e.g. based on the normalized entropy as in Eqn. (4.4). Now, the key element of active learning is the ability of the learner to *query* new class labels from



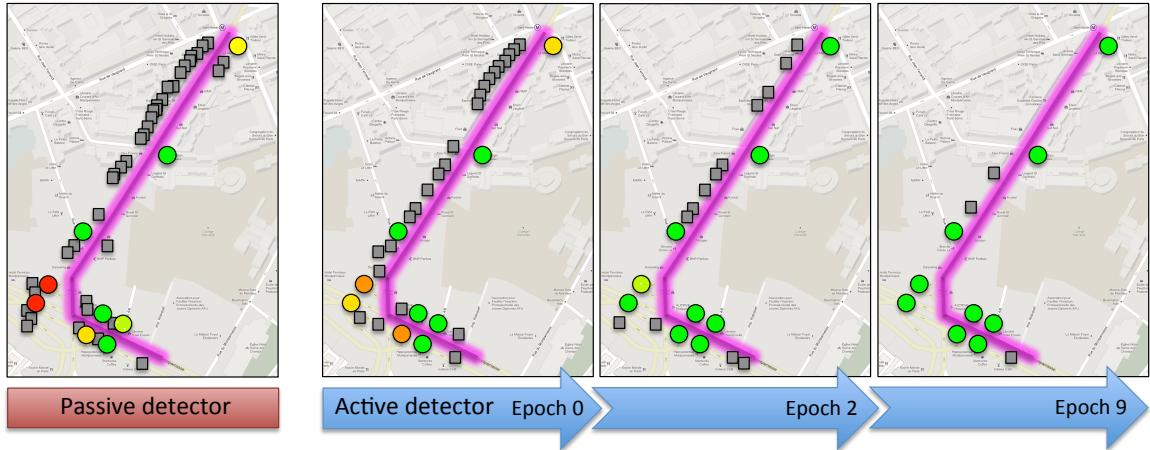
**Figure 7.4.:** Active Learning flow chart. After an initial training step, the classifier is presented new test data and reports label predictions and confidence values (here: uncertainties). These are used to ask a human supervisor for new ground truth labels, which subsequently are added to the current training data. Then, the training process is repeated with the extended training data until a stopping criterion is met.

the human supervisor. This is usually done by selecting those test points  $\mathbf{x}_i^*$ , for which the classifier has a low confidence and asking a ground truth label  $\hat{y}_i$  for them. These new data-label pairs  $(\mathbf{x}_i^*, \hat{y}_i)$  are then added to the current training data  $(\mathcal{X}_{j-1}, \mathcal{Y}_{j-1})$ , where  $j$  is the index of the current learning round, and the learning process starts again until an appropriate stopping criterion is reached. In our applications, we use a fixed number of learning rounds.

One important question in Active Learning is how to select the data samples for which semantic information, i.e. in our case class labels, are queried from the human supervisor. We refer to this as the *which-question problem*. Settles [2012] summarizes the following *query strategies*: uncertainty sampling, query-by-committee, expected model change, expected error reduction, variance reduction, and density weighting. Among these, the most used method is the uncertainty sampling, and we also use it in our framework. In the literature, there are at least two common ways to compute uncertainty from a prediction vector  $\mathbf{p}$ : the normalized entropy given in Eq. 4.4 and the *best-vs-second-best* (BVSB) method, which computes the quotient of the second-largest entry of  $\mathbf{p}$  and the largest one. Now, to address the which-question problem, the standard uncertainty sampling approach uses a *confidence threshold*  $\vartheta_c$  and decides to ask for a ground truth label  $\hat{y}$  for all those data samples which, in the current learning epoch, have been classified with a confidence lower than  $\vartheta_c$ . This strategy raises the following question: How do we know that the classifier gives meaningful uncertainty estimates so that uncertainty sampling actually makes sense? To answer this question, we will use our findings from Chapter 4 and use preferably those classifiers for Active Learning that tend to show a low level of overconfidence. A concrete example of this is given in the following.

### 7.2.2. Application to Semantic Mapping

We want to apply the general idea of Active Learning to the concrete problem of semantic mapping. Specifically, in Triebel et al. [2013a] on page 257 we consider the problem of first detecting all traffic lights from a given sequence of camera images recorded in an urban environment, and then annotating a given map automatically with this information. To do this, we use a variant of the Gaussian Process Classifier (GPC), because, as we have seen in Sec. 4.2 the GPC tends to be less overconfident in its predictions than other classifiers such as the SVM. This is a very important observation, because an overconfident classifier can not be used effectively for Active Learning, as it makes wrong predictions



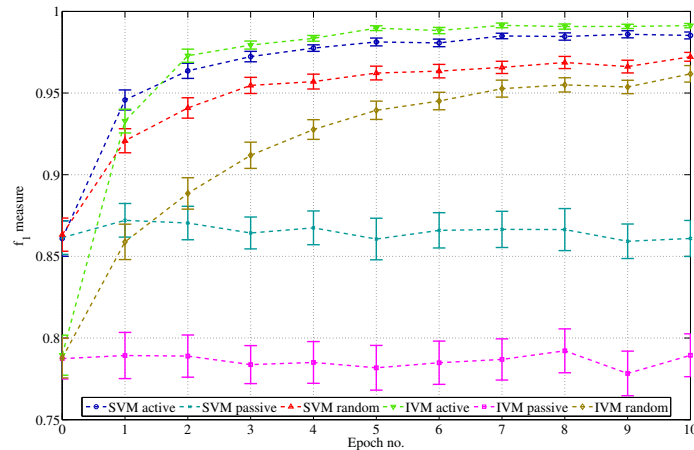
**Figure 7.5.:** Active learning in a semantic mapping context. This figure shows semantic maps indicating the positions of traffic lights along a street in Paris. Circles denote the locations of ground-truth traffic lights. The shading encodes the correctness of the classification output as provided by a probabilistic classifier: red denotes no detections, and green denotes all possible traffic light detections being found. False positives are shown as grey squares. From left to right, we first see a typical passive detector, followed by our active-learning framework at epochs 0, 2, and 9 respectively. Note that in the active learning setting the shading of the circles progresses from red to green as a greater proportion of traffic lights are correctly detected with increasing confidence. Similarly the number of false positives reduces dramatically. By epoch 2 the active learning framework already outperforms the passive detector.

with high confidence. As a result, when selecting the next data points for the label query, it is unlikely that these samples were the misclassified ones, preventing the classifier from improving on them in the next step. Note however, that an *underconfident* classifier causes less problems, because it only results in too many generated label queries, including those for which a correct classification was already found. This reduces the efficiency and increases the amount of label queries, but it does not prevent the classifier from improving its classification performance.

As already mentioned in Sec. 4.4, the GPC has the disadvantage that it requires huge computational resources, both in memory and run time. This is due to the fact that it maintains a mean  $\mu$ , as well as a covariance matrix  $\Sigma$ , which is computed from a kernel function and has size  $|\mathbf{y}|^2$ . A number of sparsification methods have been proposed in order to mitigate this computational burden, and we adopt one such sparsification method: the Informative Vector Machine (IVM) [Lawrence et al., 2002]. The main idea of this algorithm is to only use a subset of the training points denoted the *active set*,  $\mathcal{I}$ , from which an approximation  $q(f | X, \mathbf{y}) = \mathcal{N}(f | \mu, \Sigma)$  of the posterior distribution  $p(f | X, \mathbf{y})$  is computed. The IVM algorithm computes  $\mu$  and  $\Sigma$  incrementally, and at every iteration  $j$  selects the training point  $(\mathbf{x}_k, y_k)$  which maximizes the entropy difference  $\Delta H_{jk}$  between  $q_{j-1}$  and  $q_j$  for inclusion into the active set. As  $q$  is Gaussian,  $\Delta H_{jk}$  can be computed by

$$\Delta H_{jk} = -\frac{1}{2} \log |\Sigma_{jk}| + \frac{1}{2} \log |\Sigma_{j-1}|. \quad (7.2)$$

The details of the implementation can be found in Lawrence et al. [2005]. The algorithm



**Figure 7.6.:** Learning rates for the sparse Gaussian Process classifier (IVM) and a support vector machine (SVM) as indicated by the  $f_1$ -measure after each learning epoch. Measurements are averaged over 100 runs on a publicly available data set. Error bars indicate one standard error of the mean. The IVM using a normalised entropy-based data selection strategy (IVM+NE) consistently outperforms all other active learning variants in terms of overall performance and learning rate.

stops when the active set has reached a desired size. In our implementation, we choose this size to be a fixed fraction  $\gamma$  of the training set.

### 7.2.3. Results

To evaluate our approach we use the publicly available Traffic Lights Recognition (TLR) data set of Mines ParisTech, which comprises 11,179 colour images taken at 25 Hz from a car driven through central Paris at speeds under 31 mph and ground-truth labels for traffic light positions. As recommended by the authors of the dataset, we exclude sections where the vehicle was stationary for long periods of time. We use data from the first 5,800 frames for training and the remainder for testing. We compute a template-based feature set inspired by Torralba et al. [2007a], as was already used in Sec. 4.3. Each training or test window is represented by a feature vector of length 200.

When training the IVM we used an active set fraction  $\gamma$  of 0.2, which means that informative points will be added to the active set until its size is 20% of the training set size. We use a Squared Exponential (SE) kernel with white noise. Training such a classifier takes approximately 1.5 seconds on a single 3.4GHz core. The SVMs used here are trained using *libsvm* [Chang and Lin, 2011], and use the same kernel as used by the IVM. They are trained using 10-fold cross-validation on top of a grid-search over the two parameters, both in the space  $2^k$  where  $k = \{-7, -6, \dots, +4\}$ . Training takes approximately 10 minutes.

A qualitative result of our approach is given in Fig. 7.5. We see, that the Active Learner is able to improve its classification in every training epoch, while the passive learner only gives one result with many false positive and false negative detections. We note that both the passive and the active learner were provided with the same number of training samples. For a quantitative result, we refer to Fig. 7.6, which shows the learning rates in terms of  $f_1$ -measure for six different algorithms averaged over 100 runs. In particular, we used

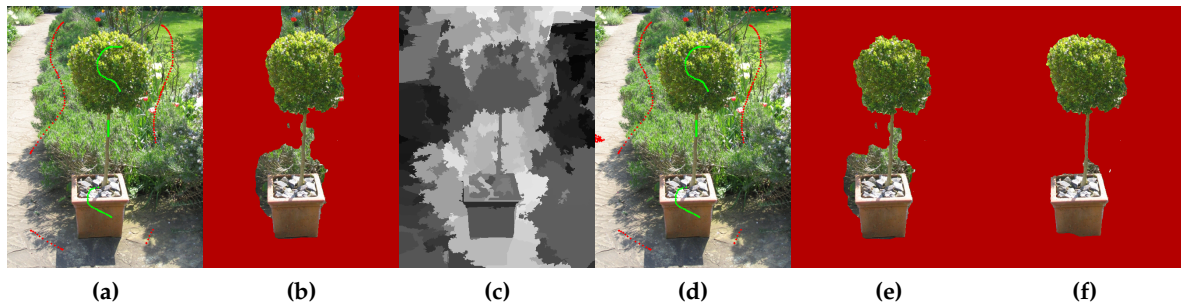
the IVM and SVM classifiers both in an active and in a passive learning setup. As we can see from the plot, the IVM learner starts off with a worse  $f_1$  measure at epoch 0 but has already exceeded the SVM by epoch 2, and is better (with non-overlapping 95% confidence bounds) in the steady state from then onwards. In addition, we performed Active Learning with both classifiers, but with a random selection of the new training samples rather than using the normalized entropy criterion. The result justifies empirically our choice of normalised entropy as a valid criterion for data selection compared to a random selection. Intuitively, both methods should improve classification by virtue of the fact that they increase the training set size. However, the results indicate that for both the IVM and the SVM, using normalised entropy leads to more rapidly improving classification performance.

### 7.3. Active Learning for Interactive Image Segmentation

In a further application example of Active Learning, we consider the problem of interactive image segmentation. In general, image segmentation is concerned with the task to separate the pixels of an image into foreground and background, or likewise into a number of different classes. Here, we will consider the former case, i.e. we do binary segmentation. In general the image segmentation problem is ill-posed, because a correct segmentation depends strongly on the application. Therefore, we focus on the *interactive* segmentation problem, where the user provides information about the regions to be segmented, e.g. by manually sampling image pixels and assigning them to a predefined region class. These user scribbles are used as ground truth information, and the aim is to infer a good segmentation using these scribbles as constraints on the labelling. To do this, many approaches have been presented in the literature with impressive results. However, current methods can reach high classification rates only by requiring comparably many user scribbles, and the number of user scribbles needed usually grows very fast as the segmentation quality approaches 100%. In the next section, we briefly describe the approach. For further details we refer to Triebel et al. [2014] on page 273.

#### 7.3.1. Algorithm Overview

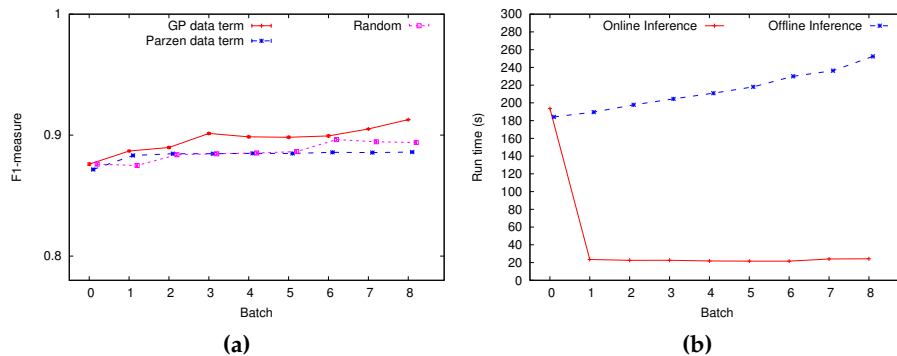
Fig. 7.7 shows an example sequence of our active learning framework for interactive image segmentation. From a set of initial user scribbles from both foreground and background regions (Fig 7.7a), our algorithm learns an IVM as in Sec. 7.2 and classifies the remaining pixels. Then, a segmentation is obtained using regularization (Fig. 7.7b), and an uncertainty measure is computed from the predictive variance returned by the IVM. As above, we use an IVM, because its uncertainty estimates are more reliable than those produced by other learning methods such as Support Vector Machines. Then, we perform an over-segmentation of the original image based on superpixels [Felzenszwalb and Huttenlocher, 2004] and compute the average classification uncertainty (entropy) for each segment (see Fig. 7.7c). In the next step, the algorithm selects the segment with the highest uncertainty to query a ground truth label from the user, samples pixels uniformly from the segment, and adds the samples with the obtained labels to the training data set (see Fig. 7.7d). Note that, due to imperfections in the segmentation, some segments can contain both foreground



**Figure 7.7.:** Example sequence of our proposed active learning framework. The algorithm starts with initial user scribbles as shown in (a). It then learns a sparse GP classifier and segments the image using the GP prediction and a regularization term (b). Then, candidate regions for new, informative user scribbles are computed (c). These are based on the normalized entropy of the GP prediction, i.e. bright regions represent a higher classification uncertainty than darker regions. In this case, a segment at the upper right border is chosen. A label is queried for these pixels (here it is background), and a sub-set of uniformly sampled pixels together with the class labels is added to the training data (d). In the next round, the classification is improved and the result is refined (e). After a few rounds (here 4 in total), the final segmentation is obtained (f).

and background pixels. In that case, the user can select a “don’t know” option, and the next segment is chosen in the order of decreasing entropies. This however, occurs only rarely when the segmentation is done sufficiently fine-grained. The whole learning and classification process is then repeated for a fixed number of times or until an appropriate stopping criterion is met (Fig. 7.7e and 7.7f).

As mentioned, we use again an IVM instead of a standard GPC, due to efficiency reasons. However, in addition to that the IVM has another advantageous property, namely its ability to compute the posterior distribution  $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})$  *incrementally*. Concretely, the algorithm loops over all active points and updates mean vector  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$  by increasing their lengths in every iteration. In particular, it keeps the lower triangular matrix  $L_d$  of a Cholesky decomposition in memory and updates it using rank-1 Cholesky updates, where  $L_d$  is of size  $d \times d$  and  $d = 1, \dots, D$ . Further details of this procedure are given in Algorithm 1 of Lawrence et al. [2005]. For our purpose, this incremental scheme is particularly useful, because it avoids the complete re-computation of the GP parameters in every training round and adds only a fixed number of rows and columns to  $L_d$ . This decreases the training time substantially. For an efficient *class prediction*, we furthermore use an online update rule, which is based on the fact that in every training round of Active Learning the same test data are considered. This is a particular property of interactive image segmentation, and is not true in general. As a consequence, it is possible to use the information from the previous round, were a correlation of each pixel with the old training data was already computed, in the current round. This can be done by splitting the Cholesky decomposition into blocks and applying the Schur complement for the inversion. Details are given in Triebel et al. [2014] (page 273).



**Figure 7.8.:** (a) Average f-measure over 8 active learning rounds. The GPC steadily improves the segmentation, because its label queries are more informative for classification. In contrast, the Parzen window only improves slightly and then remains at a lower performance level. We also show GPC results where new user scribbles are chosen randomly and not based on the entropy. This also improves the segmentation, as it increases the training data, but it is worse than the entropy-based method. (b) Run time of online and offline inference, averaged over all images. Note that in batch 0, the online and the offline method take the same time, because they both build up the initial covariance matrix. However, in later steps the online computation time drops down significantly.

### 7.3.2. Results

We evaluate our active learning approach on the benchmark data set from the University of Graz [Santner et al., 2011]. It consists of images with ground truth segmentations and user scribbles. As our method applies for foreground and background segmentation we chose a subset of 44 images from the dataset which contain only two object classes. As performance measure for this benchmark we use the  $f_1$  measure, which is defined as the harmonic mean of precision and recall.

We compare our approach with the method of Nieuwenhuis and Cremers [2013]. There, the data term is computed using a Parzen window (PW) estimator, and the training data consists of color information and positions of user scribbles. We use the same idea, but employ a GPC instead of the PW. For a quantitative evaluation, we ran active learning with the GPC and the PW on the Graz data set (Fig. 7.8a). Both approaches perform equally well in the first rounds, but then the GPC (red curve) outperforms the PW (blue curve), because it asks more informed label queries, while the PW tends to be overconfident. We also show the results for randomly selected scribbles (magenta curve) instead of those with the highest uncertainty. We see that random sampling also improves the classification, as it provides more training data in every round, but the improvement is smaller compared to selecting the most uncertain segments. This is because the GP requests the more informative user scribbles.

In Fig. 7.8b, we show the benefit of our online prediction step over the standard offline technique in every active learning round. Observe that for all but the first learning round the average run time drops from the order of minutes to the order of seconds. Also note that the increase in run time over the learning rounds is super-linear in the offline case,



	RGBD	Begbroke	USPS	Letter	Pendigits	DNA
pGB	0.2704	0.2668	0.1536	0.2628	0.1589	0.1410
aGB	0.1824	0.1463	0.1123	0.1876	0.0983	0.0867
pCB	0.1248	0.0568	0.0898	0.1161	0.0559	0.0811
aCB	0.1165	0.0517	0.0726	0.0840	0.0341	0.0724

**Table 7.1.:** Average classification errors over 100 runs.

where for the online method it is roughly linear. In the first round, the online and the offline method perform the same steps, because every pixel is compared to all training points. Currently, we compute this in parallel on 8 CPU threads, but we expect a substantial speed-up when using a GPU implementation.

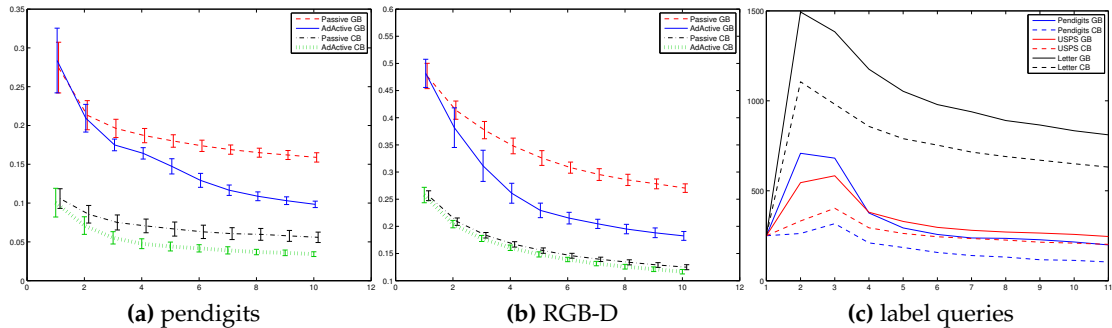
## 7.4. Efficient Active Online Learning for 3D Object Classification

As we have seen in Chapter 4, the Gaussian Process Classifier tends to be less overconfident compared to the Support Vector Machine or Boosting techniques, and we have used that fact in the previous sections to formulate Active Learning methods with faster learning rates, whenever a GPC (or an IVM) was used. However, as we mentioned already in Sec. 4.4, the GPC has huge computational demands, and even the IVM requires a lot of time for training the kernel hyperparameters. With that motivation, we developed the Confidence Boosting algorithm in Sec. 4.4.2 and showed that it is not worse in terms of overconfidence compared to the GPC, but it is much more efficient. Here, we evaluate Confidence Boosting further by applying it in an Active Learning framework.

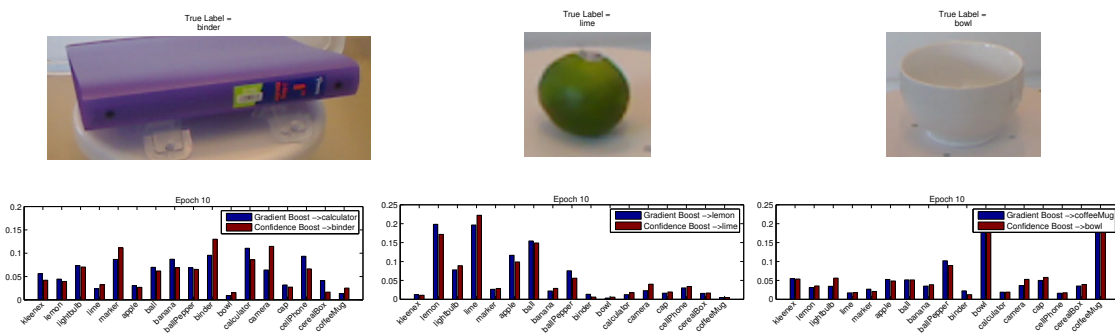
To do this, we ran active learning on the same six data sets as described in Sec. 4.4.3, once with standard gradient boosting and once with Confidence Boosting. We repeated this 100 times with the training sets randomly shuffled to obtain results that are independent on the data ordering. The mean classification errors are given in Table 7.1. Apart from the fact that active learning performs better on all data sets than passive learning, where new training data was added randomly and not based on its confidence, we see that Confidence Boosting results in lower classification errors in all cases. In particular, Confidence Boosting with active learning gave the best results for all data sets. The progress of the learning process for two data sets is depicted in Fig. 7.9 (a) and (b). Here, the means and standard deviations of the classification error are shown for every epoch, again averaged over 100 randomly shuffled runs through the data. We see that active learning improves the classification faster than passive learning. Furthermore, Confidence Boosting generally leads to better results, particularly when combined with active learning.

Fig. 7.9c shows the number of label queries for newly added training samples in every epoch of active learning. In general, we see that this number decreases for all data sets with the number of learning epochs. We also see that Confidence Boosting decreases this number further. We relate this to the fact that Confidence Boosting results in a higher correlation between uncertain and incorrect classified samples, which enables the classifier to find more incorrect samples for re-training while at the same time not losing many correct classified samples. Some qualitative results of our approach are shown in Fig. 7.10.

## 7. Active Learning



**Figure 7.9.:** (a) and (b): Learning curves for two sample data sets (left: ‘pendigits’, center: ‘RGB-D’). The x-axis shows the learning epochs, the y-axis represents the classification error on the test set. We compare standard online gradient boost (GB) with our online Confidence Boosting (CB), both using passive and active learning and show mean and standard deviations over 100 runs, where the training data was shuffled randomly. As we can see, active learning reduces the test error faster than passive learning, where new training samples are added randomly. Also, Confidence Boosting leads to steeper learning curves and better classification results. (c): Number of new label queries used per epoch. All curves start with a pool of 250 samples. The remaining training data are presented to the classifier in subsequent epochs for prediction and re-training. We see that in each epoch less additional queries are needed than in the previous one and that Confidence Boosting needs less label queries than gradient boosting.



**Figure 7.10.:** Qualitative results of our experiments. We show three different objects from the RGBD data set, for which we computed HMP descriptors on the depth information, i.e. colour is not used for classification. After 10 rounds of active learning, Confidence Boosting (CB) returned the correct label, while gradient boosting did not. Note that CB even distinguishes the lime correctly from a lemon although there was no colour information used, i.e. even such small differences in shape can be detected with our approach.

## 8. Conclusions

In this thesis, we have presented a number of different learning techniques for classification tasks in robotics and computer vision. The motivation behind these techniques was to develop efficient learning algorithms that at the same time facilitate the process for the human user by requiring as little user input as possible. To do this, we saw on one side that algorithms that operate unsupervised can still extract enough information from the data to achieve a reasonable segmentation result, albeit these results can not be used directly for tasks where semantic information is required. On the other side, we developed efficient on-line learning methods that do use human input for learning, although with the aim to do this with the least necessary effort for the user. In particular, the Active Learning methods presented here seem to be a very promising compromise between a low work load for a human supervisor and the requirement for the algorithm to receive semantic information so that certain tasks, for which such information is needed, can be performed. The further investigation of these techniques therefore seems to be a reasonable motivation for future research.



# Bibliography

- K. O. Arras, Ó. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2007.
- J. Aslam, E. Pelekhev, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications*, 8(1):95–129, Jan 2004.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- L. Bo, X. Ren, and D. Fox. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. In *Advances in neural information processing systems*, pages 2115–2123, 2011.
- C. Chang and C. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-supervised Learning*. MIT Press, 2006.
- D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 438–445, 2001.
- N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.
- S. Debnath, S. S. Baishya, R. Triebel, V. Dutt, and D. Cremers. Environment-adaptive learning: How clustering helps to obtain good training data. In *Advances in Artificial Intelligence (KI)*, 2014.
- P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004. ISSN 0920-5691.
- Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55, 1997.
- B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315(5814):972–976, February 2007.
- J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.
- M. Girolami and S. Rogers. Variational bayesian multinomial probit regression with gaussian process priors. *Neural Computation*, 18(8):1790–1817, 2006.

- H. Grimmer, R. Paul, R. Triebel, and I. Posner. Knowing when we don't know: Introspective classification for mission-critical decision making. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.
- A. Johnson. *Spin-Images: A Representation for 3-D Surface Matching*. PhD thesis, Robotics Institute, Carnegie Mellon Univ., 1997.
- R. Kästner, N. Engelhard, R. Triebel, and R. Siegwart. A bayesian approach to learning 3d representations of dynamic environments. In *Proc. of The 12th International Symposium on Experimental Robotics (ISER)*, Berlin, 2010. Springer Press.
- H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *In Joint IEEE Intern. Workshop on Vis. Surv. and Perform. Eval. of Tracking and Surv.*, 2003.
- J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *Int. Conf. on Machine Learning (ICML)*, 2001.
- K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2011.
- N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, 15:609–616, 2002.
- N. D. Lawrence, J. C. Platt, and M. I. Jordan. Extensions of the informative vector machine. In *Proceedings of the First international conference on Deterministic and Statistical Methods in Machine Learning*, pages 56–87. Springer-Verlag, 2005.
- B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005. ISBN 0-7695-2372-2.
- D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)), 1989.
- D. G. Lowe. Object recognition from local scale-invariant features. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1150–1157, 1999.
- J. Maye, L. Spinello, R. Triebel, and R. Siegwart. Inferring the semantics of direction signs in public places. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2010.
- J. Maye, R. Triebel, L. Spinello, and R. Siegwart. Bayesian on-line learning of driving behaviors. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2011.
- T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001. AAI0803033.
- K. P. Murphy. *Machine Learning – A Probabilistic Perspective*. MIT Press, 2012.

- 
- C. Nieuwenhuis and D. Cremers. Spatially varying color distributions for interactive multi-label segmentation. *Trans. on Patt. Analysis and Machine Intell.*, 35(5):1234–1247, 2013.
- R. C. of Mines ParisTech. Traffic lights recognition (TLR) data set.
- R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Trans. on Graphics*, 21(4):807–832, 2002.
- G. Overett, L. Petersson, N. Brewer, L. Andersson, , and N. Pettersson. A new pedestrian dataset for supervised learning. In *IEEE Intelligent Vehicles Symposium*, 2008.
- R. Paul, R. Triebel, D. Rus, and P. Newman. Semantic categorization of outdoor scenes with uncertainty estimates using multi-class gaussian process classification. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, 2012.
- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, January 1998. URL <http://research.microsoft.com/apps/pubs/default.aspx?id=68391>.
- J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.
- R. B. Potts. Some generalized order-disorder transformations. *Cambridge Phil Soc.*, 48, 1952.
- F. Ramos, D. Fox, and H. Durrant-Whyte. CRF-matching: Conditional random fields for feature-based scan matching. In *Robotics: Science and Systems (RSS)*, 2007.
- C. E. Rasmussen and C. K. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, Massachusetts, 2006.
- A. Rosenberg and J. Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 410–420, 2007.
- A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online multi-class lpboost. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2010.
- J. Santner, T. Pock, and H. Bischof. Interactive multi-label segmentation. In *Asian Conf. on Computer Vision*, pages 397–410. Springer, 2011.
- R. E. Schapire and Y. Freund. *Boosting*. MIT Press, 2012.
- B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.
- B. Settles. *Active Learning*. Morgan & Claypool, 2012.
- J. Shin, R. Triebel, and R. Siegwart. Unsupervised discovery of repetitive objects. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2010.

- J. Shin, R. Triebel, and R. Siegwart. Unsupervised 3d object discovery and categorization for mobile robots. In *Proc. of the International Symposium on Robotics Research (ISRR)*, 2011.
- L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2008.
- L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2008.
- L. Spinello, A. Macho, R. Triebel, and R. Siegwart. Detecting pedestrians at very small scales. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2009.
- L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart. A layered approach to people detection in 3d range data. In *special track on Physically Grounded AI of AAAI*, 2010a.
- L. Spinello, R. Triebel, and R. Siegwart. Multiclass multimodal detection and tracking in urban environments. *International Journal of Robotics Research*, 29(12):1498 – 1515, 2010b.
- L. Spinello, R. Triebel, D. Vasquez, K. Arras, and R. Siegwart. Exploiting repetitive object patterns for model compression and completion. In *European Conference on Computer Vision*, 2010c.
- J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.
- C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012. ISSN 1935-8237.
- A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, May 2007a. ISSN 0162-8828.
- A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, May 2007b. ISSN 0162-8828.
- R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.
- R. Triebel, J. Shin, and R. Siegwart. Segmentation and unsupervised part-based discovery of repetitive objects. In *Robotics: Science and Systems (RSS)*, 2010.
- R. Triebel, R. Paul, D. Rus, and P. Newman. Parsing outdoor scenes from streamed 3d laser data using online clustering and incremental belief updates. In *Robotics Track of AAAI Conference on Artificial Intelligence*, 2012.
- R. Triebel, H. Grimmitt, R. Paul, and I. Posner. Driven learning for driving: How introspection improves semantic mapping. In *The International Symposium on Robotics Research (IJRR)*, 2013a.



- R. Triebel, H. Grimmer, and I. Posner. Confidence boosting: Improving the introspectiveness of a boosted classifier for efficient learning. In *Autonomous Learning Workshop at ICRA*, 2013b.
- R. Triebel, J. Stühmer, M. Souiai, and D. Cremers. Active online learning for interactive segmentation using sparse gaussian processes. In *German Conference on Pattern Recognition*, 2014.
- T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *Int. Journal of Computer Vision*, 88(2):284–302, 2009.
- P. Viola and M. Jones. Robust real-time object detection. *Int. Journ. of Comp. Vis.*, 2002.
- C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz. Geometrical diffusion measures for MRI from tensor basis analysis. In *ISMRM '97*, page 1742, Vancouver Canada, April 1997.



# **Publications**



---

---

---

## Index of Publications<sup>1</sup>

### Chapter 3

- L. Spinello, R. Triebel, R. Siegwart: "Multimodal People Detection and Tracking in Crowded Scenes", in: In Proc. of the Conference on Artificial Intelligence (AAAI) 2008 88
- L. Spinello, R. Triebel, R. Siegwart: "Multimodal Detection and Tracking of Pedestrians in Urban Environments with Explicit Ground Plane Extraction" in: Proc. of the International Conference on Intelligent Robots and Systems (IROS) 2008 94
- L. Spinello, A. Macho, R. Triebel, R. Siegwart: "Detecting Pedestrians at Very Small Scales" in: Proc. of the International Conference on Intelligent Robots and Systems (IROS) 2009 101
- L. Spinello, R. Triebel, R. Siegwart: "Multiclass Multimodal Detection and Tracking in Urban Environments " in: In Proc. of Field and Service Robotics (FSR) 2009 107
- L. Spinello, K. O. Arras, R. Triebel, R. Siegwart: "A Layered Approach to People Detection in 3D Range Data" in: In Proc. of the Conference on Artificial Intelligence (AAAI) 2010 118
- L. Spinello, R. Triebel, R. Siegwart: "Multiclass Multimodal Detection and Tracking in Urban Environments" in: The International Journal of Robotics Research (IJRR) 29 (12): 1498-1515, 2010 124

### Chapter 4

- R. Paul, R. Triebel, D. Rus, P. Newman: "Semantic Categorization of Outdoor Scenes with Uncertainty Estimates using Multi-Class Gaussian Process Classification" in: Proc. of the International Conference on Intelligent Robots and Systems (IROS) 2012 159
- H. Grimmett, R. Paul, R. Triebel, I. Posner: "Knowing When We Don't Know: Introspective Classification for Mission-Critical Decision Making" in: Proc. of the International Conference on Robotics and Automation (ICRA) 2013 166
- R. Triebel, H. Grimmett, I. Posner : "Confidence Boosting: Improving the Introspectiveness of a Boosted Classifier for Efficient Learning" in: Proc. of the International Conference on Robotics and Automation (ICRA) 2013 (Autonomous Learning Workshop) 174

### Chapter 5

- L. Spinello, R. Triebel, D. Vasquez, K. O. Arras, R. Siegwart: "Exploiting Repetitive Object Patterns for Model Compression and Completion" in: European Conference on Computer Vision (ECCV) 2010 178

---

<sup>1</sup>Publications are sorted thematically, not chronologically.

---

J. Maye, L. Spinello, R. Triebel, R. Siegwart: "Inferring the Semantics of Direction Signs in Public Places" in: Proc. of the International Conference on Robotics and Automation (ICRA) 2010	192
J. Shin, R. Triebel, and R. Siegwart: "Unsupervised Discovery of Repetitive Objects" in: Proc. of the International Conference on Robotics and Automation (ICRA) 2010	198
R. Triebel, J. Shin, R. Siegwart: "Segmentation and Unsupervised Part-based Discovery of Repetitive Objects" in: Robotics: Science and Systems (RSS) 2010	204
J. Shin, R. Triebel, R. Siegwart: "Unsupervised 3D Object Discovery and Categorization for Mobile Robots" in: Proc. of the International Symposium on Robotics Research (ISRR) 2011	212
<b>Chapter 6</b>	
R. Kästner, N. Engelhard, R. Triebel, and R. Siegwart "A Bayesian Approach to Learning 3D Representations of Dynamic Environments" in: Proc. of The 12th International Symposium on Experimental Robotics (ISER) 2010	228
J. Maye, R. Triebel, L. Spinello, and R. Siegwart: "Bayesian On-line Learning of Driving Behaviors" in: Proc. of the International Conference on Robotics and Automation (ICRA) 2011	243
R. Triebel, R. Paul, D. Rus, P. Newman: "Parsing Outdoor Scenes from Streamed 3D Laser Data Using Online Clustering and Incremental Belief Updates" in: Proc. of the Conference on Artificial Intelligence (AAAI) 2012	249
<b>Chapter 7</b>	
R. Triebel, H. Grimmer, R. Paul, I. Posner "Driven Learning for Driving: How Introspection Improves Semantic Mapping" in: Proc. of the International Symposium on Robotics Research (ISRR) 2013	257
R. Triebel, J. Stühmer, M. Souiai, D. Cremers: "Active Online Learning for Interactive Segmentation Using Sparse Gaussian Processes" in: German Conference on Pattern Recognition (GCPR) 2014	273
S. Debnath, S. Sankar Baishya, R. Triebel, V. Dutt, D. Cremers: "Environment-adaptive Learning: How Clustering Helps to Obtain Good Training Data" in: Advances in Artificial Intelligence (KI) 2014	284

## Multimodal People Detection and Tracking in Crowded Scenes

Luciano Spinello    Rudolph Triebel    Roland Siegwart

Autonomous Systems Lab, ETH Zurich

Tannenstrasse 3, CLA E, 8092 Zurich, Switzerland

email: {luciano.spinello, rudolph.triebel, roland.siegwart}@mavt.ethz.ch

### Abstract

This paper presents a novel people detection and tracking method based on a multi-modal sensor fusion approach that utilizes 2D laser range and camera data. The data points in the laser scans are clustered using a novel graph-based method and an SVM based version of the cascaded AdaBoost classifier is trained with a set of geometrical features of these clusters. In the detection phase, the classified laser data is projected into the camera image to define a region of interest for the vision-based people detector. This detector is a fast version of the Implicit Shape Model (ISM) that learns an appearance codebook of local SIFT descriptors from a set of hand-labeled images of pedestrians and uses them in a voting scheme to vote for centers of detected people. The extension consists in a fast and detailed analysis of the spatial distribution of voters per detected person. Each detected person is tracked using a greedy data association method and multiple Extended Kalman Filters that use different motion models. This way, the filter can cope with a variety of different motion patterns. The tracker is asynchronously updated by the detections from the laser and the camera data. Experiments conducted in real-world outdoor scenarios with crowds of pedestrians demonstrate the usefulness of our approach.

### Introduction

The ability to reliably detect people in real-world environments is crucial for a wide variety of applications including video surveillance and intelligent driver assistance systems. According to the National Highway Traffic Safety Administration report (NHTSA 2007) there were 4784 pedestrian fatalities in United States during the year 2006, which accounted for 11.6% of the total 42642 traffic related fatalities. In countries of Asia and Europe, the percentage of pedestrian accidents is even higher. The number of such accidents could be reduced if cars were equipped with systems that can automatically detect, track, and predict the motion of pedestrians. However, pedestrians are particularly difficult to detect because of their high variability in appearance due to clothing, illumination and the fact that the shape characteristics depend on the view point. In addition, occlusions caused by carried items such as backpacks, as well as clutter in crowded scenes can render this task even more complex, because they dramatically change the shape of a pedestrian.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Our goal is to detect pedestrians and localize them in 3D at any point in time. In particular, we want to provide a position and a motion estimate that can be used in a real-time application, e.g. online path planning in crowded environments. The real-time constraint makes this task particularly difficult and requires faster detection and tracking algorithms than the existing approaches. Our work makes a contribution into this direction. The approach we propose is multimodal in the sense that we use 2D laser range data and CCD camera images cooperatively. This has the advantage that both *geometrical structure* and *visual appearance* information are available for a more robust detection. In this paper, we exploit this information using supervised learning techniques based on a combination of AdaBoost with Support Vector Machines (SVMs) for the laser data and on an extension of the Implicit Shape Model (ISM) for the vision data. In the detection phase, both classifiers yield likelihoods of detecting people which are fused into an overall detection probability. Finally, each detected person is tracked using multiple Extended Kalman Filters (EKF) with three different motion models and a greedy data association. This way, the filter can cope with different motion patterns for several persons simultaneously. The tracker is asynchronously updated by the detections from the laser and the camera data. The major contributions of this work are:

- An improved version of the image-based people detector by Leibe *et al.* (2005). The improvement consists in two extensions to the ISM for a reduced computation time to make the approach better suited for real-time applications.
- A tracking algorithm based on EKF with multiple motion models. The filter is asynchronously updated with the detection results from the laser and the camera.
- The integration of our multimodal people detector and the tracker into a robotic system that is employed in a real outdoor environment.

This paper is organized as follows. The next section describes previous work that is relevant for our approach. Then, we give an overview of our overall people detection and tracking system. Section 4 presents our detection method based on the 2D laser range data. Then, we introduce the Implicit Shape Model (ISM) and our extensions to the ISM. Subsequently, we explain our EKF-based tracking algorithm with a focus on the multiple motion models we



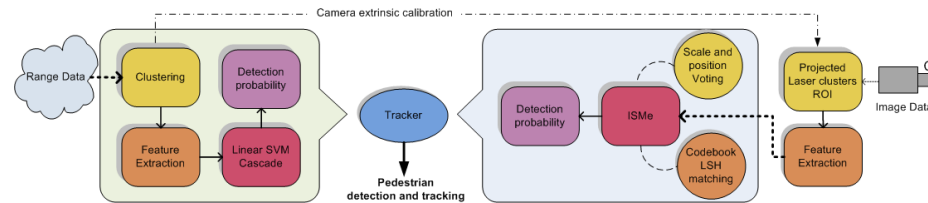


Figure 1: Overview of the individual steps of our system. See text for details.

use. Finally, we describe our experiments and conclusions.

### Previous Work

Several approaches can be found in the literature to identify a person in 2D laser data including analysis of local minima (Scheutz, Mcraven, & Cserey 2004; Schulz *et al.* 2003; Topp & Christensen 2005), geometric rules (Xavier *et al.* 2005), or a maximum-likelihood estimation to detect dynamic objects (Hähnel *et al.* 2003). Most similar to our work is the approach of Arras, Mozes, & Burgard (2007) which clusters the laser data and learns an AdaBoost classifier from a set of geometrical features extracted from the clusters. Recently, we extended this approach (Spinello & Siegwart 2008) by using multi-dimensional features and learning them using a cascade of Support Vector Machines (SVM) instead of the AdaBoost decision stumps. In this paper, we will make use of that work and combine it with an improved appearance-based people detection and an EKF-based tracking algorithm.

In the area of image-based people detection, there mainly exist two kinds of approaches (see Gavrilu (1999) for a survey). One uses the analysis of a *detection window* or *templates* (Gavrilu & Philomin 1999; Viola, Jones, & Snow 2003), the other performs a *parts-based* detection (Felzenszwalb & Huttenlocher 2000; Ioffe & Forsyth 2001). Leibe, Seemann, & Schiele (2005) presented an image-based people detector using *Implicit Shape Models* (ISM) with excellent detection results in crowded scenes.

Existing people detection methods based on camera and laser rangefinder data either use hard constrained approaches or hand tuned thresholding. Cui *et al.* (2005) use multiple laser scanners at foot height and a monocular camera to obtain people tracking by extracting feet and step candidates. Zivkovic & Kröse (2007) use a learned leg detector and boosted Haar features extracted from the camera images to merge this information into a parts-based method. However, both the proposed approach to cluster the laser data using Canny edge detection and the extraction of Haar features to detect body parts is hardly suited for outdoor scenarios due to the highly cluttered data and the larger variation of illumination encountered there. Therefore, we use an improved clustering method for the laser scans and SIFT features for the image-based detector. Schulz (2006) uses probabilistic exemplar models learned from training data of both sensors and applies a Rao-Blackwellized particle filter (RBPF) in order to track the person's appearance in the data. The RBPF tracks contours in the image based on Chamfer matching as

well as point clusters in the laser scan and computes the likelihood of different prototypical shapes in the data. However, in outdoor scenarios lighting conditions change frequently and occlusions are very likely, which is why contour matching is not appropriate. Moreover, the RBPF is computationally demanding, especially in crowded environments.

Several methods have been proposed to track moving objects in sequential data (see Cox (1993) for an overview). The most common ones include the joint likelihood filter (JLF), the joint probabilistic data association filter (JPDAF), and the multiple hypothesis filter (MHF). Unfortunately, the exponential complexity of these methods makes them inappropriate for real-time applications such as navigation and path planning. Cox & Miller (1995) approximate the MHF and JPDA methods by applying Murty's algorithm and demonstrate in simulations the resulting speedup for the MHF method. Rasmussen & Hager (2001) extend the JLM, JPDA, and MHF algorithms to track objects represented by complex feature combinations. Schumitsch *et al.* (2006) propose a method to reduce the complexity of MHT methods introducing the Identity Management Kalman Filter (IMKF) for entities with signature.

### Overview of the method

Our system is divided into three phases: training, detection and tracking (see Fig. 1). In the training phase, the system learns a structure-based classifier from a hand-labeled set of 2D laser range scans, and an appearance-based classifier from a set of labeled camera images. The first one uses a boosted cascade of linear SVMs, while the latter computes an ISM, in which a collected set of image descriptors from the training set vote for the occurrence of a person in the test set. In the detection phase, the laser-based classifier is applied to the clusters found in a new range scan and a probability is computed for each cluster to correspond to a person. The clusters are then projected into the camera image to define a region of interest, from which the appearance-based classifier extracts local image descriptors and computes a set of hypotheses of detected persons. Here, we apply a new technique to discard false positive detections. Finally in the tracking phase, the information from both classifiers is used to track the position of the people in the scan data. The tracker is updated whenever a new image or a laser measurement is received and processed. It applies several motion models per track to account for the high variety of possible motions a person can perform. In the following, we describe the particular steps of our system in detail.

### Structure Information from Laser Data Analysis

We assume that the robot is equipped with a laser range sensor that provides 2D scan points  $(\mathbf{x}_1, \dots, \mathbf{x}_N)$  in the laser plane. We detect a person in a range scan by first clustering the data and then applying a boosted classifier on the clusters, which we describe as follows.

#### Clustering

Jump distance clustering is a widely used method for 2D laser range data in mobile robotics (see Premebida & Nunes (2005) for an overview). It is fast and simple to implement: if the Euclidean distance between two adjacent data points exceeds a given threshold, a new cluster is generated. Although this approach performs well in indoor scenarios, it gives poor results for outdoor data, because the environment is geometrically more complex and bigger distances, reflections and direct sunlight effects usually occur. This often leads to over-segmented data with many small clusters. To address this problem, we use a simple and effective technique that extends the classic jump distance method. It consists in the following steps:

1. Perform jump distance clustering with threshold  $\vartheta$ . Each cluster  $\mathcal{S}_i$  is defined by its left border  $\mathbf{x}_i^l$ , its central point  $\mathbf{x}_i^c$ , and its right border  $\mathbf{x}_i^r$ :

$$\mathcal{S}_i = \{\mathbf{x}_i^l, \mathbf{x}_i^c, \mathbf{x}_i^r\} \quad (1)$$

2. Compute a Delaunay triangulation on the centers  $\mathbf{x}_i^c$ .
3. Annotate each edge  $\mathbf{e}_{ij} := (\mathbf{x}_i^c, \mathbf{x}_j^c)$  of the Delaunay graph with the Euclidean distance between  $\mathcal{S}_i$  and  $\mathcal{S}_j$ .
4. Remove edges with a distance greater than  $\vartheta$  and merge each remaining connected component into a new cluster.

Note that the same threshold  $\vartheta$  is used twice: first to define the minimum jump distance between the end points of adjacent clusters and then to define the Euclidean distance between clusters. Experimental results showed that this reduces the cluster quantity of 25% – 60%, significantly reducing overclustering. The additional computational cost due to the Delaunay triangulation and distance computation is lower compared to a full 2D agglomerative clustering approach.

#### Boosted Cascade of Support Vector Machines

We use AdaBoost (Freund & Schapire 1997) to classify the clustered laser data into the classes “person” and “no person”. AdaBoost creates a strong classifier from a set of weak classifiers. Viola & Jones (2002) further improved this approach by ordering the weak classifiers in a degenerate decision tree which they call an *attentional cascade*. This reduces the computation time significantly. We apply this method, but we use support vector machines (SVMs), in particular  $c$ -SVMs with linear kernel (Boser, Guyon, & Vapnik 1992), instead of the standard decision stumps based on thresholding. The main reason for this is to obtain a small number of classifiers in each stage and to guarantee an optimal separation of the two classes. Before applying the

SVMs, we normalize the input data in order to avoid numerical problems caused by large attribute values. The parameters  $c$  of the  $c$ -SVMs were obtained from a local search where the classification results were evaluated using 5-fold cross validation.

We denote the detection of a person using a binary random variable  $\pi$  that is true whenever a person is detected. Each of the  $L$  cascaded SVM-classifiers  $h_i$  yields either 1 or 0 for a given input feature vector  $\mathbf{f}$ . The overall detection probability can then be formulated as

$$p(\pi | \mathbf{f}) = \sum_{i=1}^L w_i h_i(\mathbf{f}) \quad (2)$$

In the learning phase, the weights  $w_i$  and the hyperplanes are computed for each SVM classifier  $h_i$ . The laser-based people detector then computes (2) for each feature vector  $\mathbf{f}$  in the test data set. In our implementation, we compute the features  $\mathbf{f}$  of a cluster  $\mathcal{S}$  as described in our previous work (Spinello & Siegwart 2008).

### Appearance Information from Image Data Analysis

Our image-based people detector is mostly inspired by the work of Leibe, Seemann, & Schiele (2005) on scale-invariant Implicit Shape Models (ISM). An ISM is a generative model for object detection and has been applied to a variety of object categories including cars, motorbikes, animals and pedestrians. In this paper, we extend this approach, but before we briefly explain the steps for learning an object model in the original ISM framework.

An Implicit Shape model consists of a *codebook*  $\mathcal{I}$  and a set of votes  $\mathcal{V}$ . The  $K$  elements of  $\mathcal{I}$  are local region descriptors  $\mathbf{d}_1^C, \dots, \mathbf{d}_K^C$  and  $\mathcal{V}$  contains for each  $\mathbf{d}_i^C$  a set of  $D_i$  local displacements  $\{(\Delta x_{ij}, \Delta y_{ij})\}$  and scale factors  $\{s_{ij}\}$  with  $j = 1, \dots, D_i$ . The interpretation of the votes is that each descriptor  $\mathbf{d}_i^C$  can be found at different positions inside an object and at different scales. To account for this, each local displacement points from  $\mathbf{d}_i^C$  to the center of the object as it was found in the labeled training data set. We can think of this as a sample-based representation of a spatial distribution  $p(\pi, \hat{\mathbf{x}} | \mathbf{d}_i^C, \mathbf{x}_i)$  for each  $\mathbf{d}_i^C$  at a given image location  $\mathbf{x}_i = (x_i, y_i)$  where  $\hat{\mathbf{x}} = (\hat{x}, \hat{y})$  denotes the center of the detected person. To obtain an ISM from a given training data set, two steps are performed:

1. **Clustering** All region descriptors are collected from the training data. The descriptors are then clustered using agglomerative clustering with average linkage. In the codebook, only the cluster centers are stored.
2. **Computing Votes** In a second run over the training data, the codebook descriptors  $\mathbf{d}_i^C$  are matched to the descriptors  $\mathbf{d}_j^I$  found in the images, and the scale and center displacement corresponding to  $\mathbf{d}_j^I$  is added as a vote for  $\mathbf{d}_i^C$ .

In the detection phase, we again compute interest points  $\mathbf{x}_i^I$  and corresponding region descriptors  $\mathbf{d}_j^I$  at various scales on a given test image  $I$ . The descriptors are matched to the

codebook and a matching probability  $p(\mathbf{d}_i^C | \mathbf{d}_j^I)$  is obtained for each codebook entry. To compute the likelihood to detect a person at location  $\bar{\mathbf{x}}$  we use the following marginalization:

$$p(\pi, \bar{\mathbf{x}} | \mathbf{x}_j^I, \mathbf{d}_j^I) = \sum_{i=1}^K p(\pi, \bar{\mathbf{x}} | \mathbf{d}_i^C, \mathbf{x}_j^I) p(\mathbf{d}_i^C | \mathbf{d}_j^I) \quad (3)$$

This defines the weight of the vote that is cast by each descriptor  $\mathbf{d}_j^I$  at location  $\mathbf{x}_j^I$  for a particular occurrence of a person at position  $\bar{\mathbf{x}}$ . The overall detection probability is then the sum over all votes:

$$p(\pi, \bar{\mathbf{x}} | \mathbf{g}^I) = \sum_{j=1}^M p(\pi, \bar{\mathbf{x}} | \mathbf{x}_j^I, \mathbf{d}_j^I) \quad (4)$$

where  $\mathbf{g}^I = (\mathbf{x}_1^I, \dots, \mathbf{x}_M^I, \mathbf{d}_1^I, \dots, \mathbf{d}_M^I)$ . With the sample-based representation, we can find the  $\bar{\mathbf{x}}$  that maximizes (4) by a maxima search using a variable-bandwidth mean shift balloon density estimator (Comaniciu, Ramesh, & Meer 2001) in the 3D voting space.

### First Extension to ISM: Strength of Hypotheses

In the definition of the ISM there is no assumption made on the particular shape of the objects to be detected. This has the big advantage that the learned objects are detected although they might be occluded by other objects in the scene. However, the drawback is that usually there is a large number of false positive detections in the image background. Leibe, Seemann, & Schiele (2005) address this problem using a minimum description length (MDL) optimization based on pixel probability values. However, this approach is rather time demanding and not suited for real-time applications. Therefore, we suggest a different approach.

First, we evaluate the quality of a hypothesis about a detected object center  $\mathbf{x}$  with respect to two aspects: the overall *strength* of all votes and the way in which the voters are *distributed*. Assume that ISM yields an estimate of a person at position  $\mathbf{x}$ . We can estimate the spatial distribution of voters  $\mathbf{x}_j^I$  that vote for  $\mathbf{x}$  using a 1D circular histogram that ranges from 0 to  $2\pi$ . When computing the weight of the vote according to (3) we also compute the angle  $\alpha$

$$\alpha(\mathbf{x}_j^I, \mathbf{x}) = \arctan 2(y_j^I - y, x_j^I - x) \quad (5)$$

and store the voting weight in the bin that corresponds to  $\alpha$ . This way we obtain a histogram  $\xi(\mathbf{x})$  with, say,  $B$  bins for each center hypothesis  $\mathbf{x}$ . Now we can define an ordering on the hypotheses based on the histogram difference

$$d(\mathbf{x}_1, \mathbf{x}_2) := \sum_{b=1}^B \xi_b(\mathbf{x}_1) - \xi_b(\mathbf{x}_2), \quad (6)$$

where  $\xi_b(\mathbf{x}_1)$  and  $\xi_b(\mathbf{x}_2)$  denote the contents of the bins with index  $b$  from the histograms of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  respectively. We say that hypothesis  $\mathbf{x}_1$  is *stronger* than  $\mathbf{x}_2$  if  $d(\mathbf{x}_1, \mathbf{x}_2) > 0$ .

The second idea is to reduce the search area in the voting space using the region of interest computed from segmented clusters in the laser data. This further reduces the search space and results in a faster and more robust detection due to the scale information.

### Second Extension to ISM:

#### High-dimensional Nearest Neighbor Search

Another problem of the ISM-based detector is the time required to compute the matching probability  $p(\mathbf{d}_i^C | \mathbf{d}_j^I)$ . Image descriptors such as SIFT, GLOH or PCA-SIFT are very effective (see Mikolajczyk & Schmid (2005) for a comparison), but they may have up to 256 dimensions. Considering that the size of the codebook can be as big as 25000, we can see that a linear nearest-neighbor (NN) search can not be used for real-time applications. A potential alternative would be the use of  $k$ D-trees, but these provide efficient NN search only for dimensions not more than around 20, because the number of neighboring cells inside a given hypersphere grows exponentially with the number of dimensions.

Therefore we apply *approximate* NN search, which is defined as follows. For a given set of  $d$ -dimensional points  $\mathcal{P} \subset \mathbb{R}^d$  and a given radius  $r$ , find all points  $\mathbf{p} \in \mathcal{P}$  for a query point  $\mathbf{q}$  so that  $\|\mathbf{p} - \mathbf{q}\|_2 \leq r$  with a probability of at least  $1 - \delta$ . This can be implemented efficiently using locality-sensitive hashing (LSH) as proposed by Andoni & Indyk (2006).

### Tracking Pedestrians

So far, we described how pedestrians can be detected in 2D laser range data and in camera images. The result of these detectors is an estimate of the position of a person at a given time frame. However, for many applications it is required to also have information about the *kinematics* of the person, e.g. provided by a motion vector. This can be achieved by tracking the position of the person and predicting the future motion based on the observations from the previous time frames. A key issue for a people tracking algorithm is the definition of the motion model. Pedestrians are not constrained to a particular kind of motion and they can abruptly change their motion direction at any time. To address this problem, we use the following motion models for each tracked person:

1. **Brownian motion:** This accounts for sudden motion changes.
2. **Constant speed:** The person does not change direction or speed.
3. **Smooth turning:** The forward speed is constant and we fit a second order polynomial into the last 10 positions of the pedestrian using least mean square fitting (LMS).

In each time step, the tracker needs to solve the data association problem that consist in finding a mapping between observations and tracked objects. In our system, we use a greedy approach to do the data association, which is performed in two steps: In the first step we choose the motion model whose prediction has the smallest distance to the closest observation. In the second step, we consider the person with the longest tracking history and assign to it the observation that is closest to it and still inside a  $3\sigma$  ellipse from the last position. For the distance computation we use the Mahalanobis metric. Then we assign the observation that is closest to the person with the second-longest history and so on. If this process ends up with unassociated observations, a

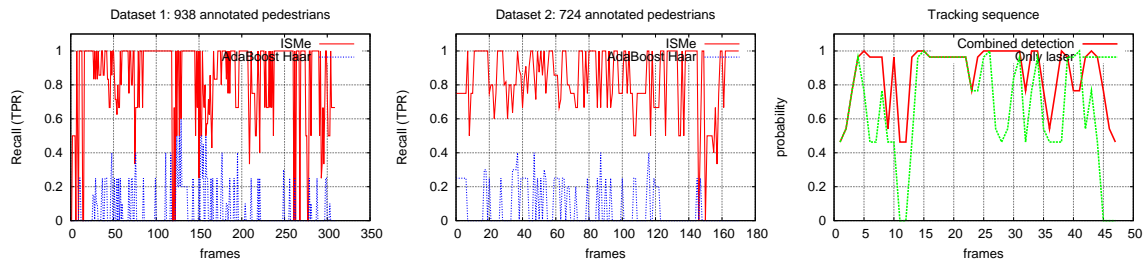


Figure 2: **Left and Center:** Image detection recall value and frames in dataset 1 and 2: ISMe vs Haar Adaboost cascade method. ISMe yields a higher detection rate than Haar/Adaboost mainly due to the distinctiveness of the features used, the detection based on a soft decision on multiple votes, and the robustness against occlusions. **Right:** Comparison of multimodal and laser-only based people detection on a tracking sequence. The tracker follows a pedestrian and a higher overall probability is obtained with the multimodal detection method compared to the laser-only detection. A part of the graph shows that the laser detection performs better in case of multiple continuous false negatives in the image detection, but then the algorithm quickly regains confidence.

new track is created. In the case that no assignment is found for a tracked person, the corresponding track is updated only using the motion prediction based on all three motion models. This is done until a new observation can be assigned, but at most for 0.5 seconds, afterwards the track is removed.

### Experimental Results

A car equipped with several active and passive sensors is used to acquire the datasets. In particular, we use a camera with a wide angle lens in combination with a 2D laser range finder in front of the car. An accurate camera-laser synchronization has been developed for this work.

#### Training datasets

**Image detection** We trained our image detection algorithm using a set of 400 images of persons with a height of 200 pixels at different positions and dressed with different clothing and accessories such as backpacks and hand bags in a typical urban environment. SIFT descriptors (Lowe 2003) computed at Hessian-Laplace interest points are collected for the codebook building. Binary segmentation masks are used to select only features that are inside the person's shape.

**Laser detection** We trained our laser-range detection algorithm computing several features on clustered points. Laser training datasets have been taken in different outdoor scenarios: a crowded parking lot and a university campus. The training data set is composed of 750 positive and 1675 negative samples. The resulting cascade consists of 4 stages with a total of 8 features.

#### Qualitative and quantitative results

We evaluated our extension of ISM (ISMe) on a challenging dataset. We collected two datasets in an urban environment and selected sequences in which pedestrians are walking, crossing, standing and where severe occlusions are present. Both sequences are manually annotated with bounding boxes of at least 80 pixel height and where at least half of a person's body is shown. The first test set consists of 311 images containing 938 annotated pedestrians, the second consists of 171 images and 724 annotated pedestrians.

In order to show a quantitative performance a comparison is performed between the classic Haar based AdaBoost pedestrian detection and our detector (see Figure 2 left and center). We can see that the AdaBoost based approach yields a very low hit rate (on average less than 50%) on both datasets due to the low robustness of Haar features and the concept of the cascade. If top level stages do not classify, the detector produces false negatives, which is often the case in a complex or occluded image frame. Conversely, ISMe has a quite high true-positive rate (TPR) during the entire frame sequence. Recall and precision rates have been computed in order to verify the role of false positives for ISMe. For both datasets the computed recall value is similar and comparably high (82% and 81%). Similar results are obtained for the precision ( $\approx 61\%$ ). ISMe has also been compared with an unconstrained implementation of ISM (maximum strength center selection and no image ROI constraint) and the resulting precision was half of the ISMe precision value.

In order to quantify the laser classification, a test data set in crowded scenes composed of 249 positive and 1799 negative samples data was prepared. We obtained a true positive rate (TPR) of 64.7% and a false positive rate (FPR) of 30.0% (FP:161 FN: 88 FP: 536 TN: 1273). To test the usefulness of using a multimodal detection algorithm a single person was tracked, and a comparison with a laser-only detection is shown in the right plot of Fig. 2. The overall detection probability for this track increases and a smoother and more confident tracking is achieved. It is important to remark that there is a part in which the multimodal detection performs slightly worse than plain laser detection. There, a continuous false negative detection occurred in the image detector, but this was quickly recovered as can be seen. We also note that many pedestrians were severely occluded, and that the detection task is so difficult that a performance of over 90% is far beyond the state of current computer vision systems.

Qualitative results from two frames are shown in Fig. 3. The box colors in the image correspond to different tracks, and the size of the filled circles is proportional to the pedestrian detection confidence. Another experiment has been performed to evaluate the time advantage of using an LSH approach during codebook matching with respect to linear

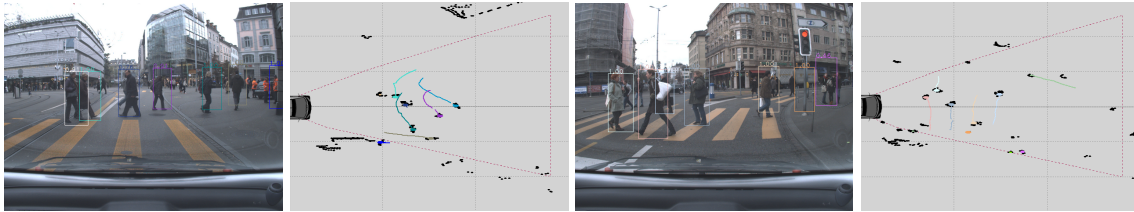


Figure 3: Qualitative results from dataset 1 and 2 showing pedestrian crossings. The colored boxes in the image describe different tracks and probability levels; the size of the filled circle in the tracking figure is proportional to the confidence of the pedestrian detection. It is important to notice that highly occluded pedestrians are also successfully detected and tracked.

neighbor search. A clustered codebook has been produced and tested by matching a random test image extracted from one of the two sequences with the codebook. LSH-based NN search resulted 12 times faster than the linear approach.

### Conclusions

In this paper, we presented a method to reliably detect and track people in crowded outdoor scenarios using 2D laser range data and camera images. We showed that the detection of a person is improved by cooperatively classifying the feature vectors computed from the input data, where we made use of supervised learning techniques to obtain the classifiers. Furthermore we presented an improved version of the ISM-based people detector and an EKF-based tracking algorithm to obtain the trajectories of the detected persons. Finally, we presented experimental results on real-world data that point out the usefulness of our approach.

### Acknowledgment

This work has been supported by the EC under contract number FP6-IST-027140-BACS.

### References

- Andoni, A., and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. of the Symp. on Found. of Comp. Sc.*
- Arras, K. O.; Mozoš, Ó. M.; and Burgard, W. 2007. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*.
- Boser, B. E.; Guyon, I.; and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, 144–152.
- Comaniciu, D.; Ramesh, V.; and Meer, P. 2001. The variable bandwidth mean shift and data-driven scale selection. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 438–445.
- Cox, I. J., and Miller, M. L. 1995. On finding ranked assignments with application to multitarget tracking and motion correspondence. *Trans. Aerospace and Electronic Systems* 31:486–489.
- Cox, I. J. 1993. A review of statistical data association for motion correspondence. *Int. Journ. of Computer Vision* 10(1):53–66.
- Cui, J.; Zha, H.; Zhao, H.; and Shibasaki. 2005. Tracking multiple people using laser and vision. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, 2116–2121.
- Felzenszwalb, P., and Huttenlocher, D. 2000. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 66–73.
- Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* 55(1):119–139.

- Gavrila, D., and Philomin, V. 1999. Real-time object detection for smart vehicles. In *IEEE Int. Conf. on Comp. Vision (ICCV)*.
- Gavrila, D. M. 1999. The visual analysis of human movement: A survey. *Computer Vision and Image Understanding (CVIU)* 73(1):82–98.
- Hähnel, D.; Triebel, R.; Burgard, W.; and Thrun, S. 2003. Map building with mobile robots in dynamic environments. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*.
- Ioffe, S., and Forsyth, D. A. 2001. Probabilistic methods for finding people. *Int. Journ. of Computer Vision* 43(1):45–68.
- Leibe, B.; Seemann, E.; and Schiele, B. 2005. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 878–885.
- Lowe, D. 2003. Distinctive image features from scale-invariant keypoints. *Int. Journ. of Computer Vision* 20:91–110.
- Mikolajczyk, K., and Schmid, C. 2005. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis & Machine Intelligence* 27(10):1615–1630.
- NHTSA. 2007. 2006 traffic safety annual assessment – a preview. Traffic Safety Facts, National Center for Statistics and Analysis. <http://www-nrd.nhtsa.dot.gov/Pubs/810791.PDF>.
- Premežida, C., and Nunes, U. 2005. Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications. In *Robotica 2005 - Scientific meeting of the 5th National Robotics Festival*.
- Rasmussen, C., and Hager, G. D. 2001. Probabilistic data association methods for tracking complex visual objects. *IEEE Trans. on Pattern Analysis & Machine Intelligence* 23(6):560–576.
- Scheutz, M.; Craven, and Cserey. 2004. Fast, reliable, adaptive, bimodal people tracking for indoor environments. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*.
- Schulz, D.; Burgard, W.; Fox, D.; and Cremers, A. 2003. People tracking with mobile robots using sample-based joint probabilistic data association filters. *Int. Journ. of Robotics Research (IJRR)* 22(2):99–116.
- Schulz, D. 2006. A probabilistic exemplar approach to combine laser and vision for person tracking. In *Robotics: Science and Systems (RSS)*.
- Schumitsch, B.; Thrun, S.; Guibas, L.; and Olukotun, K. 2006. The identity management Kalman filter (IMKF). In *Robotics: Science and Systems (RSS)*.
- Spinello, L., and Siegwart, R. 2008. Human detection using multimodal and multi-dimensional features. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*.
- Topp, E. A., and Christensen, H. I. 2005. Tracking for following and passing persons. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*.
- Viola, P., and Jones, M. 2002. Robust real-time object detection. *Int. Journ. of Computer Vision*.
- Viola, P.; Jones, M. J.; and Snow, D. 2003. Detecting pedestrians using patterns of motion and appearance. In *IEEE Int. Conf. on Comp. Vision (ICCV)*, 734. Washington, DC, USA: IEEE Computer Society.
- Xavier, J.; Pacheco, M.; Castro, D.; Ruano, A.; and Nunes, U. 2005. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 3930–3935.
- Zivkovic, Z., and Kröse, B. 2007. Part based people detection using 2d range data and images. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*.

# Multimodal Detection and Tracking of Pedestrians in Urban Environments with Explicit Ground Plane Extraction

Luciano Spinello, Rudolph Triebel and Roland Siegwart

Autonomous Systems Lab, ETH Zurich, Switzerland

email: {luciano.spinello, rudolph.triebel, roland.siegwart}@mavt.ethz.ch

**Abstract**—This paper presents a novel people detection and tracking method based on a combined multimodal sensor approach that utilizes 2D and 3D laser range and camera data. Laser data points are clustered and classified with a set of geometrical features using an SVM AdaBoost method. The clusters define a region of interest in the image that is adjusted using the ground plane information extracted from the 3D laser. In this areas a novel vision based people detector based on Implicit Shape Model (ISM) is applied. Each detected person is tracked using a greedy data association technique and multiple Extended Kalman Filters that use different motion models. This way, the filter can cope with a variety of different motion patterns. The tracker is asynchronously updated by the detections from the laser and the camera data. Experiments conducted in real-world outdoor scenarios with crowds of pedestrians demonstrate the usefulness of our approach.

## I. INTRODUCTION

The ability to reliably detect people in real-world environments is crucial for a wide variety of applications including video surveillance and intelligent driver assistance systems. The detection of pedestrians is the next logical step after the development of a successful navigation and obstacle avoidance algorithm for urban environments. However, pedestrians are particularly difficult to detect because of their high variability in appearance due to clothing, illumination and the fact that the shape characteristics depend on the view point. In addition, occlusions caused by carried items such as backpacks or briefcases, as well as clutter in crowded scenes can render this task even more complex, because they dramatically change the shape of a pedestrian.

Our goal is to detect pedestrians and localize them in 3D at any point in time. In particular, we want to provide a position and a motion estimate that can be used in a real-time application. The real-time constraint makes this task particularly difficult and requires faster detection and tracking algorithms than the existing approaches. Our work makes a contribution into this direction. The approach we propose is multimodal in the sense that we use laser range data and images from a camera cooperatively. This has the advantage that both *geometrical structure* and *visual appearance* information are available for a more robust detection. In this paper, we propose to exploit this information using supervised learning techniques that are based on a combination of AdaBoost with Support Vector Machines (SVMs) for the laser data and on an extension of the Implicit Shape Model (ISM) for the camera data. In the detection phase, both classifiers yield likelihoods

of detecting people which are fused into an overall detection probability. The information extracted from 3D and 2D data define the positioning of the hypotheses in the image. The image detection method is constrained in region of interest generated by the 2D laser and positioned in the image using a ground plane extraction method from 3D scans. Finally, each detected person is tracked using a greedy data association method and multiple Extended Kalman Filters that use different motion models. This way, the filter can cope with a variety of different motion patterns for several persons simultaneously. The tracker is asynchronously updated by the detections from the laser and the camera data. In particular, the major contributions of this work are:

- An improved version of the image-based people detector by Leibe *et al.* [12]. It consists in three extensions to the Implicit Shape Model (ISM), resulting in a reduced computation time and an improved feature selection.
- A method to discard false positive detections by computing regions of interest in the camera images.
- The use of a 3D scanning device, which facilitates a fast and robust detection of the ground plane and thus helps to disambiguate possible detections of pedestrians.

This paper is organized as follows. The next section describes previous work that is relevant for our approach. Then, we give a brief overview of our overall people detection and tracking system. The following section presents in detail our detection method based on the 2D laser range data and explains 3D plane extraction. Then, we introduce the implicit shape model (ISM), present our extensions to the ISM and expose the region of interest generation algorithm. Subsequently, we explain our EKF-based tracking algorithm focusing particularly on the multiple motion models we use. Finally, we present experiments and conclude the paper.

## II. PREVIOUS WORK

Several approaches can be found in the literature to identify a person in 2D laser data including analysis of local minima [17], [20], geometric rules [23], or a maximum-likelihood estimation to detect dynamic objects [10]. Most similar to our work is the approach of Arras *et al.* [2] which clusters the laser data and learns an AdaBoost classifier from a set of geometrical features extracted from the clusters. Recently, we extended this approach *et al.* [18] in such a way that multi-dimensional features are used and that they are learned using a cascade of Support Vector Machines (SVM)

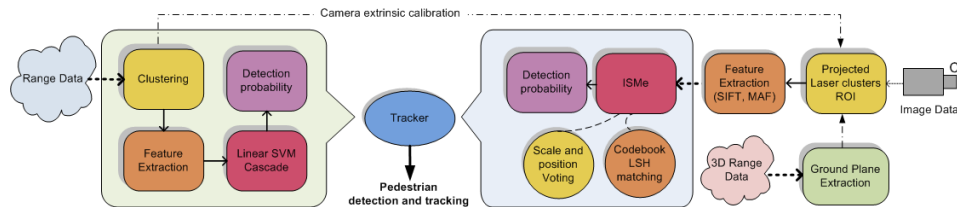


Fig. 1. Overview of the individual steps of our system. See text for details.

instead of the AdaBoost decision stumps. In the area of image-based people detection, there mainly exist two kinds of approaches (see [8] for a survey). One uses the analysis of a *detection window* or *templates* [7], [22], the other performs a *parts-based* detection [5], [11]. Leibe *et al.* [12] presented an image-based people detector using *Implicit Shape Models* (ISM) with excellent detection results in crowded scenes.

Existing people detection methods based on camera *and* laser rangefinder data either use hard constrained approaches or hand tuned thresholding. Zivkovic and Kröse [24] use a learned leg detector and boosted Haar features extracted from the camera images to merge this information into a parts-based method. However, both the proposed approach to cluster the laser data using Canny edge detection and the extraction of Haar features to detect body parts is hardly suited for outdoor scenarios due to the highly cluttered data and the larger variation of illumination encountered there. Therefore, we use an improved clustering method for the laser scans and SIFT features for the image-based detector. Schulz [16] uses probabilistic exemplar models learned from training data of both sensors and applies a Rao-Blackwellized particle filter (RBPF) in order to track the person's appearance in the data. However, in outdoor scenarios lighting conditions change frequently and occlusions are very likely, which is why contour matching is not appropriate. Moreover, the RBPF is computationally demanding, especially in crowded environments.

### III. OVERVIEW OF THE METHOD

Our system is divided into three phases: training, detection and tracking (see Figure 1). In the training phase, the system learns a structure-based classifier from a hand-labeled set of 2D laser range scans, and an appearance-based classifier from a set of labeled camera images. The first one uses a boosted cascade of linear SVMs, while the latter computes an implicit shape model (ISM), in which a collected set of image descriptors from the training set vote for the occurrence of a person in the test set. In the detection phase, the laser-based classifier is applied to the clusters found in a new range scan and a probability is computed for each cluster to correspond to a person. The clusters are then projected into the camera image to define a region of interest and positioned using the information of the ground plane extracted from the online retrieved 3D point cloud. Thus an appearance-based classifier extracts local image descriptors and uses them to obtain a set of hypotheses of detected persons. Here, we apply a new technique to discard false positive detections. Finally

in the tracking phase, the information from both classifiers is used to track the position of the people in the scan data. The tracker is updated whenever a new image or a laser measurement is received and processed. It applies several motion models per track to account for the high variety of possible motions a person can perform. For the scope of this paper, we omit the details of our tracking algorithm and refer instead to [19] for an extensive explanation. In the following, we describe the particular steps of our system in detail.

### IV. STRUCTURE INFORMATION: LASER DATA ANALYSIS

Our robotic system features a 2D and a 3D laser range scanner. The dense and frequent 2D range data is used to estimate possible locations of a person's legs, and the 3D point clouds are used to extract the ground plane to aid the appearance-based person detector (see section V).

#### A. Clustering and Classification of 2D range data

A graph based reasoning on the classic *jump distances* segmentation has been proposed in [18] in order to address the problem of clustering range data in outdoor scenario. Experimental results showed that this reduces the cluster quantity of 25%–60%, significantly reducing overclustering but maintaining clusters information.

We use an improved version of Adaboost [6] based on a cascade of support vector machines (SVMs) [18] to classify the clustered laser data into the classes “person” and “no person”. The main reason for this is to obtain a small number of classifiers in each stage and to guarantee an optimal separation of the two classes. We denote the detection of a person using a binary random variable  $\pi$  that is true whenever a person is detected. Each of the  $L$  cascaded SVM-classifiers  $h_i$  yields either 1 or 0 for a given input feature vector  $\mathbf{f}$ . The overall detection probability can then be formulated as

$$p(\pi | \mathbf{f}) = \sum_{i=1}^L w_i h_i(\mathbf{f}) \quad (1)$$

In the learning phase, the weights  $w_i$  and the hyperplanes are computed for each SVM classifier  $h_i$ . The laser-based people detector then computes (1) for each feature vector  $\mathbf{f}$  in the test data set.

#### B. Ground Plane Extraction from 3D Scans

As mentioned, a point cloud  $\mathcal{P}$  obtained with our 3D rotating scanner device reflects the full 360° environment of the vehicle. The idea is to use this information to extract the

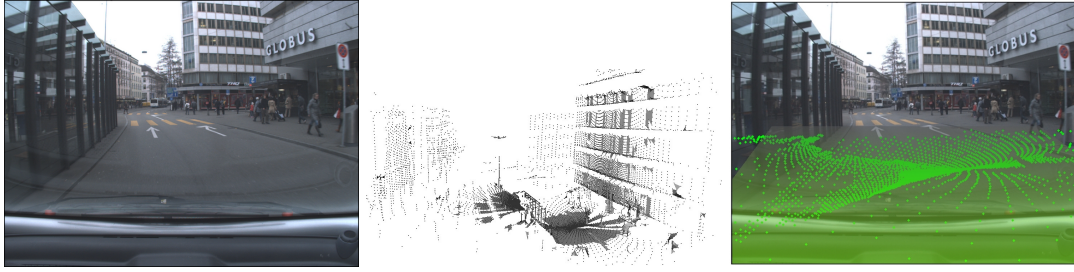


Fig. 2. 3D ground plane extraction. **Left:** Camera image as seen from the inside of the vehicle. **Middle:** Triangulated 3D point cloud of the same scene (seen from above). **Right:** Camera image with the points of the extracted ground plane overlaid.

position of the ground plane in the local environment of the vehicle to be able to further disambiguate detected persons from the camera images and to reduce false positives. In the literature, there exist many different approaches to detect planes in 3D range data [13], [21], [9]. For the application described here we want to detect and track persons if as fast as possible. Therefore, we decided to use a simple but time efficient region growing technique to detect the ground plane. The criterion for a scan point to belong to the ground plane is that its corresponding normal vector deviates only slightly (in our implementation by maximal  $25^\circ$ ) from the upright vector  $(0, 0, 1)^T$  and that it is not farther away from its closest neighbor than a given threshold (we use  $1\text{ m}$ ). The region growing is initiated always at the same fixed point right in front of the vehicle at the ground level. To efficiently compute the normal vectors, we exploit the fact that the point clouds are structured in *slices* – each scan line of the vertically mounted rotating laser scanner accounts for one slice. This facilitates a fast and simple mesh triangulation performed by connecting two consecutive points from one slice with one point of the consecutive slice. From this triangulation the normal vectors are easily computed from the normalized cross product of difference vectors. An example result of the ground plane extraction is shown in Figure 2. To clarify: rectangular bounding boxes are created in the image where laser clusters are found then the extracted ground plane is used to place those region of interest (ROI) at the correct height in the image. The resulting ROI placement helps the image detector in creating valid detection hypotheses.

## V. APPEARANCE INFORMATION: IMAGE DATA ANALYSIS

Our image-based people detector is mostly inspired by the work of [12] on scale-invariant Implicit Shape Models (ISM). An ISM is a generative model for object detection. In this paper we extend this approach, but before we briefly explain the steps for learning an object model in the original ISM framework.

An Implicit Shape model consists of a *codebook*  $\mathcal{I}$  and a set of votes  $\mathcal{V}$ . The  $K$  elements of  $\mathcal{I}$  are local region descriptors  $\mathbf{d}_1^C, \dots, \mathbf{d}_K^C$  and  $\mathcal{V}$  contains for each  $\mathbf{d}_i^C$  a set of  $D_i$  local displacements  $\{(\Delta x_{i,j}, \Delta y_{i,j})\}$  and scale factors  $\{s_{i,j}\}$  with  $j = 1, \dots, D_i$ . The interpretation of the votes is

that each descriptor  $\mathbf{d}_i^C$  can be found at different positions inside an object and at different scales. To account for this, each local displacement points from  $\mathbf{d}_i^C$  to the center of the object as it was found in the labeled training data set. To obtain an ISM from a given training data set, two steps are performed:

- 1) **Clustering** All region descriptors are collected from the training data. The descriptors are then clustered using agglomerative clustering with average linkage. In the codebook, only the cluster centers are stored.
- 2) **Computing Votes** In a second run over the training data, the codebook descriptors  $\mathbf{d}_i^C$  are matched to the descriptors  $\mathbf{d}_j^I$  found in the images, and the scale and center displacement corresponding to  $\mathbf{d}_j^I$  is added as a vote for  $\mathbf{d}_i^C$ .

In the detection phase, we again compute interest points  $\mathbf{x}_j^I$  and corresponding region descriptors  $\mathbf{d}_j^I$  at various scales on a given test image  $I$ . The descriptors are matched to the codebook and a matching probability  $p(\mathbf{d}_i^C | \mathbf{d}_j^I)$  is obtained for each codebook entry. With the sample-based representation, we can detect a person at location  $\bar{\mathbf{x}}$  by a maxima search using variable bandwidth mean shift balloon density estimation [4] in the 3D voting space.

### A. First Extension to ISM: Strength of Hypotheses

In the definition of the ISM there is no assumption made on the particular shape of the objects to be detected. This has the big advantage that the learned objects are detected although they might be occluded by other objects in the scene. However, the drawback is that usually there is a large number of false positive detections in the image background. [12] address this problem using a minimum description length (MDL) optimization based on pixel probability values. However, this approach is rather time demanding and not suited for real-time applications. Therefore, we suggest a different approach.

First we evaluate the quality of a hypothesis of a detected object center  $\mathbf{x}$  with respect to two aspects: the overall *strength* of all votes and the way in which the voters are *distributed*. Assume that ISM yields an estimate of a person at position  $\mathbf{x}$ . We can estimate the spatial distribution of voters  $\mathbf{x}_j^I$  that vote for  $\mathbf{x}$  using a 1D circular histogram that ranges from 0 to  $2\pi$ . When computing the weight of the vote



we also compute the angle  $\alpha$

$$\alpha(\mathbf{x}_j^I, \mathbf{x}) = \arctan2(y_j^I - y, x_j^I - x) \quad (2)$$

and store the voting weight in the bin that corresponds to  $\alpha$ . This way we obtain a histogram  $\xi(\mathbf{x})$  with, say,  $B$  bins for each center hypothesis  $\mathbf{x}$ . Now we can define an ordering on the hypotheses based on the histogram difference:

$$d(\mathbf{x}_1, \mathbf{x}_2) := \sum_{b=1}^B \xi_b(\mathbf{x}_1) - \xi_b(\mathbf{x}_2), \quad (3)$$

where  $\xi_b(\mathbf{x}_1)$  and  $\xi_b(\mathbf{x}_2)$  denote the contents of the bins with index  $b$  from the histograms of  $\mathbf{x}_1$  and  $\mathbf{x}_2$  respectively. We say that hypothesis  $\mathbf{x}_1$  is *stronger* than  $\mathbf{x}_2$  if  $d(\mathbf{x}_1, \mathbf{x}_2) > 0$ . The second idea is to reduce the search area in the voting space using the region of interest computed from segmented clusters in the laser data. This further reduces the search space and results in a faster and more robust detection due to the scale information.

### B. Second Extension to ISM: Features weight analysis

An important problem of classifying high dimensional feature vectors consist in the correct positioning of the separating hypersurfaces between negative and positive samples. The original ISM approach does not consider this problem and it just classifies the feature distribution of pedestrian feature descriptors  $\pi^+$  thus, during the detection step, it uses a distance threshold  $T$  in order to match features to the codebook. In this paper we enrich the pedestrian feature set

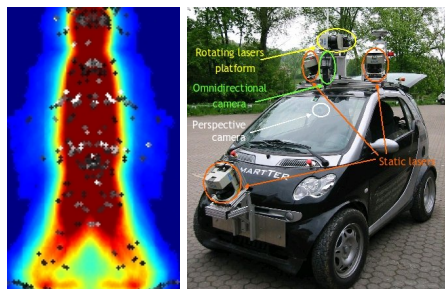


Fig. 3. **Left:** Features are weighted for their positional stability  $w_i^u$ . Features found in the trunk are more stable (white), features found in the legs are less stable due to the part motion. **Right:** Smarter vehicle platform.

using two different informative weights in order to select and treat differently each match in the detection phase. Features found in the pedestrian silhouette and features found in the background are now both collected in the training phase. Therefore, a neighborhood analysis of each positive feature descriptor is computed considering the quantity of negative samples in a radius of distance  $T$  (the same value used in the detection step). This value called  $w_i^f$  is then normalized with respect to the cardinality of the negative set  $\pi^-$ :

$$w_i^f = 1 - \frac{\text{card}(\text{neigh}_i^T(\pi^-))}{\text{card}(\pi^-)} \quad (4)$$

This weight gives an information about the distinctiveness of each feature, assigning very low values to positive samples

in loci where a high number of negative descriptors are found. In order to prune out *weak* feature vectors without impoverishing the learned pedestrian feature distribution, a low value of  $w_i^f$  have to be chosen. This elimination method decreases the amount of false positive matching and it can be seen as a compact way of expressing a  $k-nn$  classification.

Another proposed improvement in the classification method is to consider statistics in the position of the positive feature set as an informative cue of the pedestrian pose. Pedestrian features are analyzed for positional stability with respect to the object center: more the same feature is found in the same area more a high weight  $w_i^u$  is assigned. According to this weight, features found on the trunk of the pedestrian body will have high values due to its rigidness and features found on the limbs area will have a low value due to their flexibility and position change with respect to the object center. Rigid features will vote the center as a part of a rigid body keeping a fixed angle between the vector pointing to the object center and the vector parallel to the direction of its support (the direction in which the descriptor is computed to be rotation invariant). The rest of the features are classified also with their support angle and matched on the codebook during detection with a given variance in order to distinct that similar descriptors at totally different angles do not classify pedestrians (see Figure 3)

### C. Third Extension to ISM:

#### High-dimensional Nearest Neighbor Search

Another problem of the ISM-based detector is the time required to compute the matching probability  $p(\mathbf{d}_i^c | \mathbf{d}_j^f)$ . Image descriptors such as SIFT, GLOH or PCA-SIFT are very powerful (see [15] for a comparison), but they may have up to 256 dimensions. Considering that the size of the codebook can be as big as 25000, we can see that neither a linear nearest-neighbor (NN) search can be used for real-time applications or  $kD$ -trees that provide efficient NN search only for dimensions not more than 20, because the number of neighboring cells inside a given hypersphere grows exponentially with the number of dimensions.

Therefore we apply *approximate* NN search, which is defined as follows. For a given set of  $d$ -dimensional points  $\mathcal{P} \subset \mathbb{R}^d$  and a given radius  $r$ , find all points  $\mathbf{p} \in \mathcal{P}$  for a query point  $\mathbf{q}$  so that  $\|\mathbf{p} - \mathbf{q}\|_2 \leq r$  with a probability of at least  $1 - \delta$ . This can be implemented efficiently using locality-sensitive hashing (LSH) as proposed by [1].

#### D. Region of interest generation in urban environment

A common problem of ISM based methods is the tendency of generating a high quantity of false positives. In the voting stage an image feature can match several times a codebook entry and therefore it can vote for multiple object centers. Due to object symmetries, feature mismatches and scene configurations (i.e. vertical structures, complex buildings) strong false positive object hypotheses can occur in empty or unlikely areas on the image. In this paper we propose an effective and fast way to remove this kind of errors based on a distance transform computation. The idea here is that

large connected ridges in the distance transform image can be safely disregarded in the detection process because they do not contain any gradient information, which is a necessary condition for the detection of a pedestrian. Two additional parameters are required here: The minimal area  $I_q$  of a ridge that can be discarded, and a safety distance  $I_w$  between a pixel and the edge that is closest to it in the image. Both of these parameters are set so that no contour of a pedestrian is included in the discarded area (in our case we use  $I_q =$  and  $I_w =$ ). This method is particularly effective in urban environments where roads and sky are often visible and contain no or little information. It consists in the following four steps (see Figure 4):

- 1) Compute an edge map using Canny edge detector.
- 2) Compute an approximate distance transform [3].
- 3) Cluster connected components from all points that have a distance of at least  $I_w$  to the nearest edge.
- 4) Discard all regions with an area that is bigger than  $I_q$ . The remaining polygonal map constitutes the region of interest for the pedestrian detection.

The only assumption we make is that a sufficient contrast is present in the image, which is reasonable, because object detection is generally hard in low contrast images.

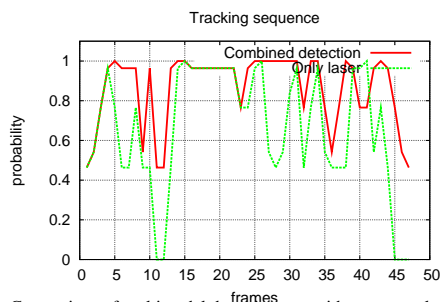


Fig. 6. Comparison of multimodal detection rate with respect to laser based people detection on a tracking sequence. The tracking follows a pedestrian and an overall higher probability is obtained with multimodal detection method than a laser detection method. A part of the graph shows that laser detection performs better in case of multiple continuous false negatives obtained from image detection but then it quickly regains confidence.

## VI. EXPERIMENTAL RESULTS

### A. Training datasets

Our mobile platform Smartert has been equipped with an IBEO ALASKA laser scanner ( $0.25deg$  resolution,  $180deg$  field of view, max range up to  $200m$ ), a rotating turntable with two SICK LMS 291-S05 lasers ( $3D$  laser) ( $1.0deg$  resolution,  $180deg$  field of view, max range up to  $200m$ ), and a camera behind the windscreen (see Fig. 3).

1) *Image detection*: We trained our image detection algorithm using a set of 400 images of persons with a height of 200 pixels at different positions and dressed with different clothing and accessories such as backpacks and hand bags in a typical urban environment. SIFT descriptors [14] computed at Hessian-Laplace interest points are collected for the codebook building. Binary segmentation masks are used to select only features that are inside the person's shape.

2) *Laser detection*: We trained our laser-range detection algorithm computing several features on clustered points. Laser training datasets have been taken in different outdoor scenarios: a crowded parking lot and a university campus. The training data set is composed of 750 positive and 1675 negative samples. The resulting cascade consists of 4 stages with a total of 8 features.

### B. Qualitative and quantitative results

We evaluated our extension of ISM (ISMe) on a challenging dataset. We collected two datasets in an urban environment and selected sequences in which pedestrians are walking, crossing, standing and where severe occlusions are present. Both sequences are manually annotated with bounding boxes of at least 80 pixel height and where at least half of a person body is shown. The first test set consists of 311 images containing 938 annotated pedestrians, the second consists of 171 images and 724 annotated pedestrians.

In order to show a quantitative performance several comparisons have been performed. A comparison between ISM, the proposed ISM extended (ISMe) and Haar based Adaboost (HAda) classifier is shown in the Precision-Recall graph of Fig. 5 (**top center**). Equal error rates (EER) are highlighted in each curve in order to show the performance gain. It is important to notice that at higher Recall values ISM (and HAda) shows a low precision (lots of false positives), while our method, thanks to generated ROIs and the proposed extension performs much better. HAda in general shows the limit of using boosted cascades and not robust Haar features for obtaining detection in complex backgrounds: if top level stages do not classify, the detector produces false negatives, which is often the case in a complex or occluded image frame. ISMe is significantly flatter than the other two methods and tends to the optimal upper right corner of the graph. Another comparison presented is the normalized difference in number of features processed between ISM and ISMe (Fig. 5(**top right**)). ISMe works with one type of descriptor and one type of interest point, ISM usually has two or three. In average the number of descriptors to be matched and processed by ISMe is less than half than ISM. Therefore, we considered a clustered codebook and a single ROI in the image with a fixed number of features (about 150) and we activated the approximate NN in the matching step to show a speed gain of about 5 times between the two methods. Moreover, we plotted Recall over frames to show a comparison for each sample between ISMe and HAda. We can see, as we expected, AdaBoost based approach yields a very low hit rate, conversely, ISMe has a quite high true-positive rate during the entire sequence frame. Another experiment shown in the section is a comparison between ISMe and ISM in the false positive rate. Here the difference is evident and it is interesting to see that the two graphs never intersects, depicting a clear advantage of using ISMe. To quantify: ISMe, ISM and HAda obtained respectively Recall 80%; 81%; 78% at Precision 63%; 22%; 0.01%.

We evaluated the laser classification on a data set in crowded scenes with 249 positive and 1799 negative samples.

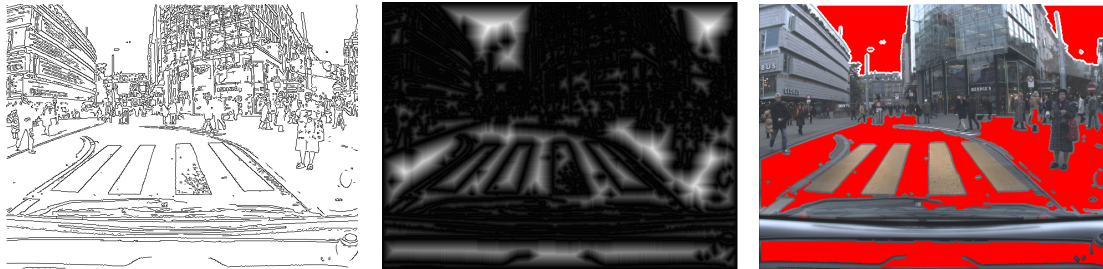


Fig. 4. Region of interest generation. Uninformative content is discarded from the image by reasoning on the distance transformed image. **Left:** Edge image (Canny). **Middle:** Approximate distance transform. **Right:** Result of the clustering in the distance transform image: areas in red are discarded

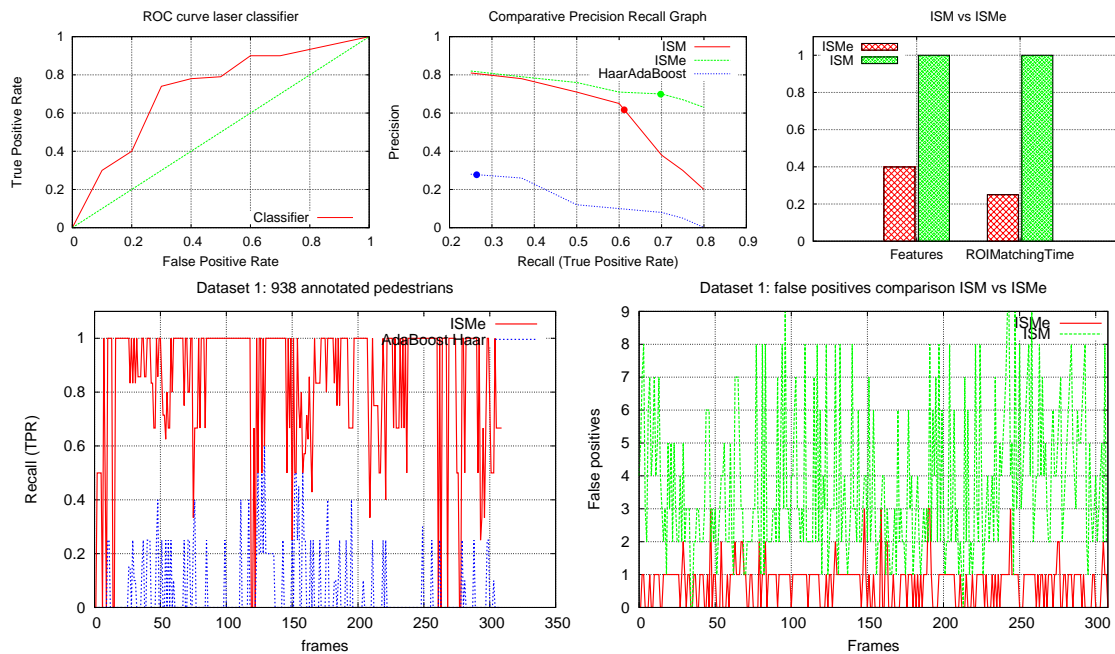


Fig. 5. **Top Left:** ROC curve for the laser classifier. AUC is  $\approx 0.8$ . **Top Center:** Precision Recall graph to compare ISM, ISMe and Haar based Adaboost, equal error rate (EER) is respectively at 26%; 61%; 26%; 69%. Notice the flatness of ISMe specially at high Recall values. **Top Right:** Average features quantity to match in ISM vs ISMe is shown in the left histogram; matching time is evaluated in the right histogram when approximate NN search is activated. **Lower Left:** Image detection recall value on frames in dataset 1: ISMe vs Haar Adaboost cascade method. ISMe obtains a higher detection rate than the other method mainly due the distinctiveness of the features used, the detection given by a soft decision on multiple votes and the robustness against occlusion. **Lower Right:** False positives in image classification evaluated for each frame of dataset 1 compared to standard ISM. Here it is clear the advantage of restricting the voting in ROIs with the other proposed improvements.

We obtained a true positive rate (TPR) of 74.7% and a false positive rate (FPR) of 30.0% (TP:184 FN: 65 FP: 536 TN: 1273), the ROC curve is shown in Fig. 5(top left).

We evaluated the usefulness of the multimodal detection computing statistics of pedestrian detection at maximum range of 15m. In order to quantify the performance of the system we considered the probability evolution of tracking a single person with both sensors and with just one 2D laser (see Fig. 6). The overall detection probability for this track increases and a smoother and more confident tracking is achieved. It is important to remark that there is a part in which the multimodal detection performs slightly worse than plain laser detection. There, a continuous false negative

detection occurred in the image detector but this was quickly recovered as can be seen. We also note that many annotated pedestrians are severely occluded, and the detection task is so difficult that a performance of over 90% is far beyond the state of current computer vision systems.

Qualitative results are shown in Fig. 7. The box colors in the image describe different tracks, the size of the filled circle is proportional to the pedestrian detection confidence.

## VII. CONCLUSIONS

In this paper, we presented a method to reliably detect and track people in crowded outdoor scenarios using 2D and 3D laser range data and camera images. We showed

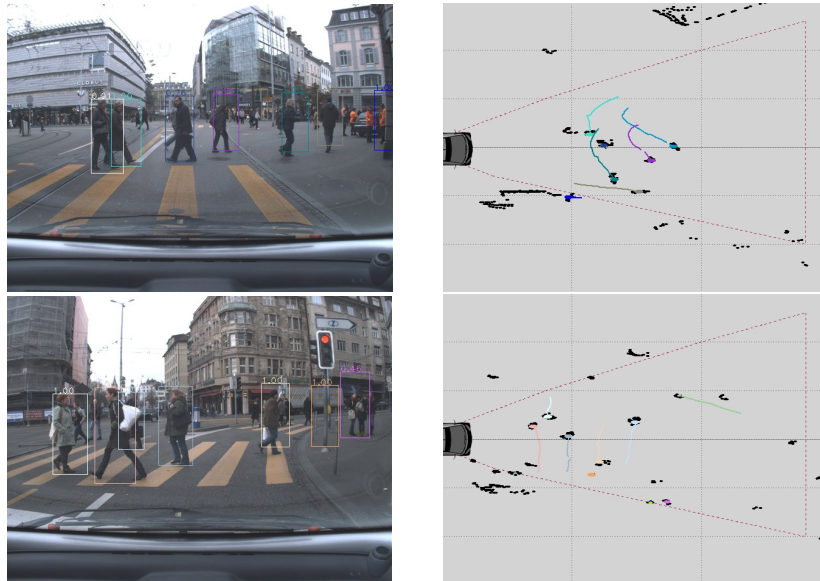


Fig. 7. Qualitative results from dataset 1 and 2 showing pedestrian crossings. The colored boxes in the image describe different tracks and probability levels; the size of the filled circle in the tracking figure is proportional to pedestrian detection confidence. It is important to notice that highly occluded pedestrians are also successfully detected and tracked.

that the detection of a person is improved by cooperatively classifying the feature vectors computed from the input data, where we made use of supervised learning techniques to obtain the classifiers. Furthermore we presented an improved version of the ISM based people detector and an EKF-based tracking algorithm to obtain the trajectories of the detected persons. Finally, we presented experimental results on real-world data that point out the usefulness of our approach.

#### VIII. ACKNOWLEDGMENTS

This work was conducted and funded within the EU Integrated Projects BACS - FP6-IST-027140

#### REFERENCES

- [1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proc. of the Symp. on Found. of Comp. Sc.*, 2006.
- [2] K. O. Arras, Ó. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2007.
- [3] G. Borgefors. Distance transformations in digital images. *Computer Vision, Graphics and Image Processing*, 34:344–371.
- [4] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 438–445, 2001.
- [5] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 66–73, 2000.
- [6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [7] D. Gavrila and V. Philomin. Real-time object detection for “smart” vehicles. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.
- [8] D. M. Gavrila. The visual analysis of human movement: A survey. *Comp. Vis. and Image Und. (CVIU)*, 73(1):82–98, 1999.
- [9] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44:15–27, 2003.
- [10] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2003.
- [11] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. Journ. of Comp. Vis.*, 43(1):45–68, 2001.
- [12] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 878–885, Washington, DC, USA, 2005. IEEE Computer Society.
- [13] Y. Liu, R. Emery, D. Chakrabarti, W. Burgard, and S. Thrun. Using EM to learn 3D models with mobile robots. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2001.
- [14] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journ. of Comp. Vis.*, 20:91–110, 2003.
- [15] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.
- [16] D. Schulz. A probabilistic exemplar approach to combine laser and vision for person tracking. In *Robotics: Science and Systems (RSS)*, Philadelphia, USA, August 2006.
- [17] D. Schulz, W. Burgard, D. Fox, and A. Cremers. People tracking with mobile robots using sample-based joint probabilistic data association filters. *Int. Journ. of Robotics Research (IJRR)*, 22(2):99–116, 2003.
- [18] L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2008.
- [19] L. Spinello, R. Triebel, and R. Siegwart. Multimodal people detection and tracking in crowded scenes. In *Proc. of the Twenty-Third Conference on Artificial Intelligence (AAAI)*, 2008.
- [20] E. A. Topp and H. I. Christensen. Tracking for following and passing persons. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, 2005.
- [21] R. Triebel, W. Burgard, and F. Dellaert. Using hierarchical EM to extract planes from 3d range scans. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2005.
- [22] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE Int. Conf. on Computer Vision (ICCV)*, page 734, Washington, DC, USA, 2003. IEEE Computer Society.
- [23] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, pages 3930–3935, 2005.
- [24] Z. Zivkovic and B. Kröse. Part based people detection using 2d range data and images. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, San Diego, USA, November 2007.

## Detecting Pedestrians at Very Small Scales

Luciano Spinello, Albert Macho, Rudolph Triebel and Roland Siegwart

Autonomous Systems Lab, ETH Zurich, Switzerland

email: {luciano.spinello, rudolph.triebel, roland.siegwart}@mavt.ethz.ch

**Abstract**—This paper presents a novel image based detection method for pedestrians at very small scales (between  $16 \times 20$  and  $32 \times 40$ ). We propose a set of new distinctive image features based on collections of local image gradients grouped by a *superpixel* segmentation. Features are collected and classified using AdaBoost. The positive classified features then vote for potential hypotheses that are collected using a mean shift mode estimation approach. The presented method overcomes the common limitations of a sliding window approach as well as those of standard voting approaches based on interest points. Extensive tests have been produced on a dataset with more than 20000 images showing the potential of this approach.

### I. INTRODUCTION

From the different participants in typical urban traffic scenarios, pedestrians are the most vulnerable ones as they are not protected by any kind of equipment as they exist for motorists and cyclists. This fact is lamentably reflected in the annual traffic accident statistics, as they are published, e.g. by the Touring Club Switzerland (TCS) [1]. Here, two major trends can be observed: first the steady decrease in the total number of dead and seriously injured persons over the last 30 years, and second the increase in the percentage of dead and injured pedestrians. The former is mostly due to the growing number of safety systems available for modern vehicles, while the latter originates from the fact that primarily motorists and cyclists benefit from such safety systems, but not pedestrians. One way to address this problem is to build more intelligent driver assistant systems that aim at protecting the driver *and* the pedestrian and avoid a potential collision. A major requirement for this is, of course, the reliable detection of pedestrians in urban traffic environments. However, this task is rendered particularly difficult by at least the following two facts:

- Pedestrians show a very high variability in shape and color due to physical size, clothing, carried items, etc.
- In urban environments, especially in city centers, pedestrians most often appear in large numbers, e.g. when crossing at a traffic light. This results in many occlusions where the pedestrians are only partly visible.

Despite these difficulties, there are already some encouraging approaches to detect pedestrians, majorly based on camera data (e.g.[15]), but also using 2D laser range scanners [2] or both [23]. However, these systems require a certain minimal size at which the pedestrians are visible in the data, which has the drawback that pedestrians that are far away, as well as children, can not be detected reliably. According



Fig. 1. Detection of very small scale pedestrians in a urban walkway.

to the rule of thumb from theoretical traffic lessons, a car that moves with  $50\text{km}$  per hour needs  $40\text{m}$  to come to a full stop. This is still far from the maximal distance at which pedestrians can be detected with current approaches, using a lens that provides still an acceptable opening angle (above  $90$  degrees). In this paper, we present an approach to detect pedestrians that are up to  $50\text{m}$  away while the lens still provides a wide field of view. The size in which the pedestrians appear in the image is as low as  $16$  by  $20$  pixels. Our proposed technique uses a supervised learning algorithm consisting of the following two major steps:

- **Training** Based on a *superpixel* segmentation proposed by Felzenszwalb and Huttenlocher [9] and a computation of the image gradient, segments of strong edge pixels are extracted. From these edge segments  $s$ , we extract feature vectors based on a combination of histograms of gradient orientations and the angles that each line segment from a polyline approximation of  $s$  forms with the horizontal axis. These features are used to train an AdaBoost classifier [11]. In addition, we store the positive training examples in a *codebook* together with their displacement vectors with respect to the object centers. This is inspired by the voting scheme of the Implicit Shape Model (ISM) approach (see [15]).
- **Classification** Again, we compute edge segments and feature vectors. Then we run the classifier and collect all votes for object centers that are cast from edge segments

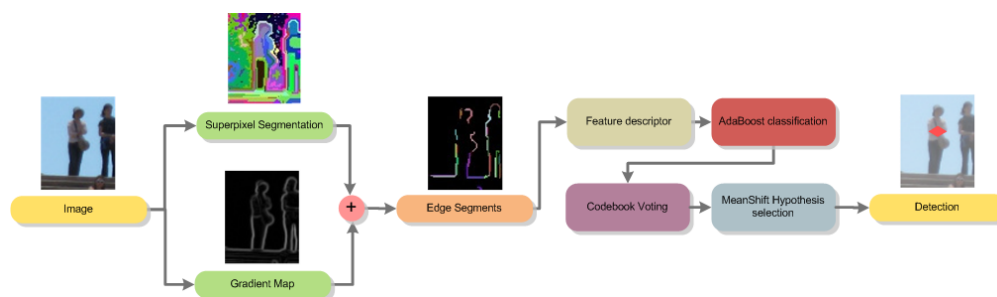


Fig. 2. Flowchart of our detection algorithm.

classified as positive. Using mean shift mode estimation [4], we obtain object positions for which many edge segments vote and thus are strong hypotheses for the position of an object, i.e. a pedestrian (see also Fig. 2).

Our approach avoids both the necessity of a sliding window, as e.g. in [24], [5], and the requirement of a minimal number of detected interest points (e.g. Hessian or Harris corners [17]) to obtain robust hypotheses for small objects such as in [15], [22]. We present the following novelties:

- the segmentation of edges from the gradient image using a superpixel segmentation. This divides the edges into chunks of homogeneous gradient variability and provides highly informative local edge features. This overcomes the usage of an overlapping tessellation to learn object features and uses a more semantical subdivision: at these image sizes, superpixels tend to segment persons into more meaningful parts like torso, head, limbs. The reason for that is that, due to the smaller resolution, the gradient variability is usually lower than at higher scales.
- a novel image descriptor particularly suited for the detection of objects at small scales,
- a classifier based on a combination of AdaBoost and the voting scheme known from the ISM approach.

The paper is organized as follows. In the next section we discuss approaches from the literature that are most closely related to our method. In Sec. III, we describe the feature extraction and the details of our edge descriptor. Then, in Sec. IV we present our classification technique and the hypothesis generation. Sec. V shows the experimental results and in Sec. VI we draw our conclusions.

## II. RELATED WORK

In the area of image-based people detection, there mainly exist two kinds of approaches (see [18] for a survey). One uses the analysis of a *detection window* [5] or *templates* [12], [24], the other performs a *parts-based* detection [8], [13]. Leibe *et al.* [15] presented an image-based people detector using *Implicit Shape Models* (ISM) with excellent detection results in crowded scenes. An extension of this method that proposes a feature selection enhancement and a nearest neighbor search optimization has been already shown

in [22][23]. In the specific area of small scales pedestrian detection very few works are present. Viola *et al.* [24] detect small pedestrians (bigger than the ones detected in this paper) including a time integration. Efros *et al.* [6] uses optical flow reasoning to detect humans and understand actions. Ferrari *et al.* [10] classify contours for detecting simple objects (coffee mugs, animals) in clutter by using an iterative path search among linked segments. The superpixel method has been introduced by Ren and Malik [20] using a Normalized Cut criterion [21] to recursively partition an image using contour and texture cues. Other methods have been proposed to obtain quality superpixel segmentations [9], [7].

## III. FEATURE EXTRACTION

In the literature, many different approaches are presented to compute local interest point detectors and appropriate region descriptors (for a comparison see [17]). However, for our particular problem of object detection at very small scales, none of these approaches is well suited for the following reasons:

- 1) In areas of many small objects, usually – if at all – only few interest points such as Harris corners or Hessian blobs can be detected. Thus, the number of possible voters is very low compared to the number of objects to be detected. This results in detection results with low confidence. We therefore decided to use edges instead of interest points, as described below.
- 2) Standard descriptors such as SIFT [16], Histogram of Oriented Gradients (HOG) [5], and shape context [3] represent the local information in a high dimensional feature space. One could think of applying such descriptors to all (or some) points of an edge chain, but this would result in a large number of feature dimensions. Given that the size of the objects to be detected usually ranges only about 300 pixels, this seems inappropriate.

As a conclusion, we aim at finding a simple but informative descriptor that is defined on chains of edge pixels and can be computed efficiently. The decision to use chains of edge pixels or, as we will denote them, *edge segments*, is somehow inspired by the use of *edgelets* for detecting pedestrians (see [25]). In the following, we present the details of our method to compute edge segments and local descriptors.

### A. Superpixel Segmentation

The aim of this first step of our detection algorithm is to preprocess a given input image and to obtain a more semantic representation that is independent on the pixel resolution of the image. One common way to achieve that is by grouping image pixels into regions in such a way that all pixels in a region are similar with respect to some kind of similarity measure. In the literature, this is also known as *image segmentation*, and it is crucial for a large number of algorithms and applications in computer vision. Many different algorithms have been suggested for this problem and we refer to the related work section in [9] for a good overview. Two of the more recent and mostly used approaches, namely [20] and [9], define a graph where the nodes are the image pixels and the graph edges are defined by a neighbor relationship between pixels. Of these two, the approach by Felzenszwalb and Huttenlocher [9] is more tuned for computational efficiency and the one by Ren and Malik [20] is more robust and yields more informative regions. For our application of detecting small scale objects, the use of complex similarity measures such as the peaks in contour orientation energy as in [20] is not required. Therefore, we use the former approach in our framework. This algorithm groups the pixels into segments so that the minimal dissimilarity *across* two segments is still higher than the maximal dissimilarity *within* both segments. The number of produced segments – usually named *superpixels* – can be adjusted by a parameter  $k$ . An important characteristic of this method is its ability to preserve details in low-variability image regions while ignoring details in high-variability regions. Therefore, it is especially suited for our application, because pedestrians at small scales are usually represented by only very few pixels in which the color variability is comparably low due to the lower sampling resolution. This means that one superpixel often represents an entire body part like a leg, a torso, or a head.

### B. Edge Segments and the Edge Descriptor

As mentioned above, we need to find a descriptor that is not only assigned to single interest points, as those occur less frequently in areas of small scale objects. Using superpixels as regions of interest are a much better choice here, as they are always found and they represent a higher vicinity. However, defining a region descriptor for superpixels would result in very complex computations. For our purpose, this is not appropriate, as we only want to represent the information contained in small image regions. As a tradeoff between single pixels and regions, we use *edge segments*, which are defined as chains of edge pixels that lie inside a superpixel. For the computation of the edge pixels, we apply the Sobel operator to the grayscale image and remove edges with a gradient magnitude that is below a threshold  $\tau$ . From that, we compute the edge segments by simply applying the superpixel segmentation described above to the edge image.

Adapted to our choice of edge segments we define a descriptor that reflects the local information of each edge segment. This information is later used for our object detection

algorithm. In accordance to the notion of a region descriptor, we refer to this as an *edge descriptor*. In our experiments, we tested the following two kinds of edge descriptors:

- **Histogram of orientations:** The local gradient orientations along an edge segment are collected in a histogram with  $n$  bins: each bin  $B_i$  counts the number  $e_i$  of edge points  $\mathbf{p}$  at which the gradient  $\gamma(\mathbf{p})$  has a certain orientation (see Fig. 4, left). For the descriptor, we use  $2n$  values, where the first  $n$  are the values  $e_i$ , normalized by the sum  $m := \sum_{i=1}^n e_i$ , and the second  $n$  values are the sums  $\sum_{\mathbf{p}_j \in B_i} |\gamma(\mathbf{p}_j)|$  for each bin  $B_i$ , again normalized by  $m$ . We name this descriptor HIST.
- **Vector of directions:** First we compute for each edge segment a polyline approximation consisting of  $l$  line segments. We do this using a variant of split-and-merge. Then, we collect all angles between the line segments and the horizontal axis in a vector of length  $l$  (see Fig. 4, right). We name this descriptor VECT.

## IV. FEATURE CLASSIFICATION

Based on the feature extraction described in the previous section, our goal is to formulate an algorithm that classifies these feature vectors into one of the two classes 'pedestrian' or 'background'. For this task, we employ a supervised learning technique that uses a hand-labeled training data set with positive examples of small scale pedestrians. Many techniques have been proposed to achieve this task. Two very successful approaches are the face detection algorithm of Viola and Jones [24] and the voting technique named Implicit Shape Model (ISM) by Leibe *et al.* [15]. The advantage of the first method is the strength of AdaBoost [11], i.e. a classifier that is arbitrarily accurate on the training data and at the same time yields a rating of the most relevant feature dimensions for classification. The downside is that the image has to be searched with a sliding window approach and at different scales. In contrast, the voting scheme of ISM relies on scale invariant features that are stored in a codebook along with the relative position of the object center. No feature search is needed here in the image, but the algorithm does

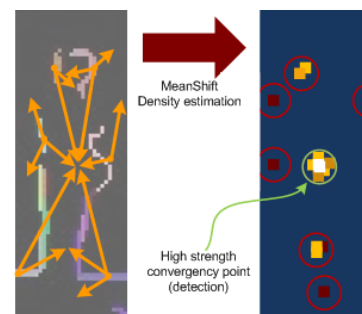


Fig. 3. Visual explanation of codebook voting. 1) Matched descriptors vote for different center positions 2) Mean Shift mode estimator is run in order to converge in local high density areas in the voting space 3) High strength hypotheses are selected as detections.

not rank certain feature dimensions over others when finding matches in the codebook. Thus, extracted feature vectors may vote for a potential object center, even though they reveal a low evidence for the occurrence of the object.

In this paper, we suggest to combine both ideas to a method that pre-classifies a given feature vector using AdaBoost and then, in the case a positive classification, searches for a vote of the object center in the codebook. The details of this are described in the following.

#### A. AdaBoost Classification

Boosting is a method to combine a set of weak classifiers into a strong classifier. The only requirement for a weak binary classifier is that its classification error on any given training data set is bigger than 0.5, i.e. it must be better than random guessing. Strong classifiers, however, can reach arbitrary low training error rates. AdaBoost [11] achieves this by adaptively assigning weights to the training examples and iteratively learning  $M$  weak classifiers  $h_i$  and corresponding weights  $\alpha_i$ . After learning, the sum

$$g(\mathbf{z}) := \sum_{i=1}^M \alpha_i h_i(\mathbf{z}) \quad (1)$$

is used to decide whether a given test feature  $\mathbf{z}$  is classified as positive or negative by simply taking the sign of the result of  $g$ . A broadly used type of weak classifiers are *decision stumps*, and we also use them in our framework. A decision stump finds a hyperplane  $\eta$  in feature space that is perpendicular to one feature dimension. It is uniquely defined by the index of the feature dimension, the orientation of the normal vector of  $\eta$ , and the distance of  $\eta$  to the origin.

The features extracted in the previous step are expressed as a  $2n$  dimensional point for the first case and as a  $l$  dimensional point in the second case. Features quality are evaluated by learning a classifier for each kind of descriptor. Moreover, we measured the quality of the combination of descriptor VECT with HIST concatenating their values in a single feature of dimension  $2n + l$ . We call this descriptor MIX.

#### B. Descriptor Codebook

The main idea of voting based classification techniques, such as the one described by Leibe *et al.* [15], is to collect a set of image descriptors together with displacement vectors,



Fig. 4. The two types of edge descriptors used in our detection algorithm. **Left:** Histogram of orientations: for each edge point of a segment we use the orientations of the corresponding gradient, here shown in yellow on four sample edge points, and compute a histogram over them. **Right:** Vector of line segment orientations: From a polyline approximation to the edge segment, here shown as yellow arrows, we store the orientation angles of each line segment in a vector.

usually named *votes*, and to store them into a *codebook*. The justification of this is that each descriptor can be found at different positions inside an object. Thus, a vote points from the position of the descriptor to the center of the object as it was found in the training data. To obtain a codebook from labeled training data, all descriptors are clustered, usually using agglomerative clustering, and the cluster centers are stored, along with all votes corresponding to a particular cluster. For the detection, new descriptors are computed on a test image and matched against the descriptors in the codebook. The votes that are cast by each matched descriptor are collected in a *voting space*, and a mean-shift maximum density estimator is used to find the most likely position of an object (see Fig. 3).

#### C. Detecting Pedestrians

Once the AdaBoost classifier is trained and a codebook is created from the training data, our detection algorithm proceeds as follows. For a given input image, the gradient map and the superpixel segmentation is computed. Using the latter ones, we obtain the edge segments of the test image. Then, we compute the descriptors as described above and apply AdaBoost using equation (1). All descriptors that are classified positive, are matched to the entries in the codebook. Here, we do a range search to find all descriptors  $\mathbf{d}$  that are within a given Euclidean distance  $r$  from the query descriptor  $\mathbf{d}_q$ . Then, all the votes cast from these descriptors are collected in the voting space by adding their displacements to the centroid of the edge segment for which  $\mathbf{d}_q$  was computed. In the last step, we apply mean shift mode estimation [4] in the voting space to find the most likely object center for the given votes. Here, we set the kernel radius to half of the width of the training images. To initialize the mean shift estimation, we first collect all votes in a 2D histogram with  $0.5w \times 0.5h$  bins where  $w$  and  $h$  are the width and height of the test image, and then start mean shift at the position of the biggest bins. After convergence of mean shift, we obtain all object hypotheses. From these, we retain those that have a minimum number of votes  $\tau_v$ .

## V. EXPERIMENTS

To evaluate our detection algorithm quantitatively, we applied it on a large set of test images with labeled positive and negative examples. We trained our classifier with images from pedestrians in two sizes, namely  $16 \times 20$  and  $32 \times 40$  pixels. This corresponds in our case to an approximate distance of  $56m$  and  $28m$ , respectively (the focal length of our lens is  $4.2mm$ ).

#### A. Setting the Parameters

As mentioned before, our algorithm depends on several parameters: the superpixel coarseness  $k$ , the gradient strength threshold  $\tau$ , the length of the descriptor vectors  $m$  and  $n$ , and the distance parameter  $r$  for codebook clustering. To determine these parameters, we created a validation dataset of 2000 random images and evaluated 25 combinations of these parameters on these images. To limit the parameter



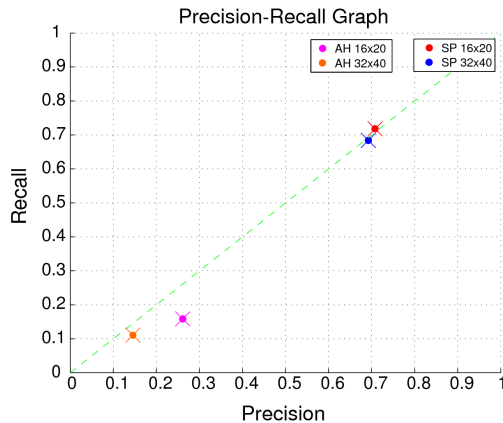


Fig. 5. Precision-Recall graph for our detection algorithm (SP) and for AdaBoost with Haar features as in [14] (AH). Due to the special design of our descriptor for small scales, the detection on the smaller images yields better results. The green line depicts the Equal Error Rate (EER)

search space, we chose  $m$  and  $n$  from the interval  $[6, 8]$  as described in [5], and  $\tau$  from  $[0, 40]$  to ensure that at most 15% of the gradient information is lost (considering that the maximal possible value is 255). The parameter combination with the maximal sum of true positive and true negative detections was used in the later experiments. We obtained  $k = 25$ ,  $m = 8$ ,  $n = 8$ ,  $r = 18$  and  $\tau = 25$ .

### B. Training

For training, we used an internationally standard dataset: the NICTA large pedestrian image dataset [19]. It contains pictures with pedestrians taken in typical urban environment. They appear either alone or in crowded scenes with partial occlusions, in different poses (walking, standing, sitting) and in a broad range of lighting variations. Negative examples are represented by random crops of images from indoor and outdoor environments.

We randomly selected 10000 positive images and 50000 negative images for each scale. In total we trained our algorithm with 120000 image samples. In each image we encountered between 10 and 20 edge segments, i.e. several million descriptors were used for training. We used 5 times more negative training examples to provide a large variety of background. To assess the quality of the AdaBoost training we used a *leave-one-out* cross validation, in which data is partitioned into subsets such that the analysis is initially performed on a single subset, while the other subsets are retained for confirming and validating the initial results. The training was performed on a quad core Intel Xeon CPU with 4GB of RAM in several hours of processing time.

### C. Quantitative Results

The test set is composed of 24000 images with 4000 and 20000 negative examples. We evaluated our algorithm for three different kinds of edge descriptors (see Sec. III-B): Histogram of orientations (HIST), Vector of directions

(VEC), and both (MIX). The evaluations of the three types of classifiers (HIST, VECT, and MIX) are shown in tables I–III for image size  $16 \times 20$  and in tables IV–VI for  $32 \times 40$ . The precision-recall values are depicted in Fig. 5, along with the result from the full-body detector for  $14 \times 28$  images by Kruppa *et al.* [14]. This method, outperformed by our technique, uses AdaBoost with Haar features and is very similar to the one that is described by Munder and Gavrila [18] as close to the best.

The VEC descriptor yields a much lower True Positive Rate (TPR) than the HIST descriptor, which is most probably due to the information loss caused by the polyline approximation of an edge segment. Note that the False Positive Rate (FPR) of both descriptors are similar. The best results are obtained using the combination (MIX) of both descriptors that improves each statistics. It is important to remark that the results for images of size  $16 \times 20$  are generally better than for those of size  $32 \times 40$ . The reason for this the specific design of our feature descriptors: a bigger image scale tends to exhibit a higher level of detail, therefore the superpixel segmentation yields edge segments that are less distinctive compared to those from the low scale images. In the latter ones, superpixels represent body parts at a higher semantic level (legs, heads, arms), whereas at larger scales, the superpixels are less informative. Moreover, due to the fact that low scale images have a lower resolution, mainly strong edge pixels prevail. This means that thresholding the gradient map at the same value  $\tau$  results in a lower loss of information. Nevertheless, the proposed technique performs comparably well for images at higher scale: the TPR is only about 3% and the TNR is only about 5% lower.

As a qualitative result, we show in Fig 6 the detection result from a fraction of the test data set at image size  $16 \times 20$ . All images are arranged in a grid and the estimated object centers are depicted with yellow dots.

## VI. CONCLUSIONS

We presented a novel image based detection method for pedestrians at very small scales. For this particular problem with sparse visual information we propose a new feature descriptor inspired by edgelets in combination with superpixel segmentation. Our technique overcomes common drawbacks of the standard interest point voting approach and of the scrolling window approaches using a descriptor codebook and a robust AdaBoost classification technique. We have evaluated parameters and show quantitative results on a large dataset, showing the effectiveness of our method. In future works we want to investigate how an intelligent tracking can improve the results and how to improve the feature robustness with respect to the scale magnification.

## VII. ACKNOWLEDGMENTS

This work was funded within the EU Projects BACS-FP6-IST-027140 and EUROPA-FP7-231888.

TABLE I  
CONFUSION MATRIX FOR SIZE 16x20 -  
HIST DESCRIPTOR

Ground truth	Prediction	
	P	N
P	70.5%	29.5%
N	18.0%	82.0%

TABLE IV  
CONFUSION MATRIX FOR SIZE 32x40 -  
HIST DESCRIPTOR

Ground truth	Prediction	
	P	N
P	66.7%	33.3%
N	23.8%	76.2%

TABLE II  
CONFUSION MATRIX FOR SIZE 16x20 -  
VECT DESCRIPTOR

Ground truth	Prediction	
	P	N
P	56.6%	43.4%
N	18.8%	81.2%

TABLE V  
CONFUSION MATRIX FOR SIZE 32x40 -  
VECT DESCRIPTOR

Ground truth	Prediction	
	P	N
P	53.0%	47.0%
N	23.9%	76.1%

TABLE III  
CONFUSION MATRIX FOR SIZE 16x20 -  
MIX DESCRIPTOR

Ground truth	Prediction	
	P	N
P	71.8%	28.2%
N	17.1%	82.9%

TABLE VI  
CONFUSION MATRIX FOR SIZE 32x40 -  
MIX DESCRIPTOR

Ground truth	Prediction	
	P	N
P	68.4%	31.6%
N	22.1%	77.9%



Fig. 6. Qualitative result of our detection algorithm. 600 full size images of correct detections from the test dataset are shown here in matrix form. The yellow dots are the estimated object centers. To keep the presentation uncluttered, the detected bounding box for each image is not displayed.

## REFERENCES

- [1] <http://www.tcs.ch/main/de/home/sicherheit/infrastrukturen/statistik.unfalle.html>. (in german).
- [2] K. O. Arras, Ó. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2007.
- [3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 24:509–522, 2002.
- [4] D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619, 2002.
- [5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.
- [6] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, pages 726–733, Nice, France, 2003.
- [7] D. Engel, L. Spinello, R. Triebel, R. Siegwart, H. Bülthoff, and C. Curio. Medial features for superpixel segmentation. In *Proc. of Machine Vision and Applications*, 2009.
- [8] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 66–73, 2000.
- [9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. Journ. of Comp. Vis.*, 59(2):167–181, 2004.
- [10] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *European Conf. on Computer Vision (ECCV)*, pages III: 14–28, 2006.
- [11] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [12] D. Gavrila and V. Philomin. Real-time object detection for “smart” vehicles. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.
- [13] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. Journ. of Comp. Vis.*, 43(1):45–68, 2001.
- [14] H. Kruppa, M. Castrillon-Santana, and B. Schiele. Fast and robust face finding via local context. In *Joint IEEE Intern. Workshop on Vis. Surv. and Perform. Eval. of Tracking and Surv.*, 2003.
- [15] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 878–885, Washington, DC, USA, 2005. IEEE Computer Society.
- [16] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. Journ. of Comp. Vis.*, 20:91–110, 2003.
- [17] K. Mikołajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 27(10):1615–1630, 2005.
- [18] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 28(11):1863–1868, 2006.
- [19] G. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Petterson. A new pedestrian dataset for supervised learning. In *IEEE Intelligent Vehicles Symposium*, 2008.
- [20] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proc. of Conf. on Computer Vision*, 2003.
- [21] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis & Machine Intell.*, 22:888–905, 2000.
- [22] L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *IEEE Int. Conf. on Intell. Rob. and Sys. (IROS)*, 2008.
- [23] L. Spinello, R. Triebel, and R. Siegwart. Multimodal people detection and tracking in crowded scenes. In *Proc. of The AAI Conference on Artificial Intelligence*, July 2008.
- [24] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *Int. Journ. of Comp. Vis.*, 63(2):153–161, 2005.
- [25] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 2005.

## Multiclass Multimodal Detection and Tracking in Urban Environments

Luciano Spinello, Rudolph Triebel, and Roland Siegwart

**Abstract** This paper presents a novel approach to detect and track pedestrians and cars based on the combined information retrieved from a camera and a laser range scanner. Laser data points are classified using boosted Conditional Random Fields (CRF), while the image based detector uses an extension of the Implicit Shape Model (ISM), which learns a codebook of local descriptors from a set of hand-labeled images and uses them to vote for centers of detected objects. Our extensions to ISM include the learning of object sub-parts and template masks to obtain more distinctive votes for the particular object classes. The detections from both sensors are then fused and the objects are tracked using an Extended Kalman Filter with multiple motion models. Experiments conducted in real-world urban scenarios demonstrate the usefulness of our approach.

### 1 Introduction

One research area that has turned more and more into the focus of interest during the last years is the development of driver assistant systems and (semi-)autonomous cars. In particular, such systems are designed for operation in highly unstructured and dynamic environments. Especially in city centers, where many different kinds of transportation systems are encountered (walking, cycling, driving, etc.), the requirements for an autonomous system are very high. One key prerequisite for such systems is a reliable detection and distinction of dynamic objects, as well as an accurate estimation of their motion direction and speed. In this paper, we address this problem focusing on the detection and tracking of pedestrians and cars. Our system is a robotic car equipped with cameras and a 2D laser range scanner. As we will show, the use of different sensor modalities helps to improve the detection results.

---

Autonomous Systems Lab, ETH Zurich, Switzerland,  
e-mail: {luciano.spinello, rudolph.triebel}@mavt.ethz.ch, rsiegwart@ethz.ch  
This work was funded within the EU Projects BACS-FP6-IST-027140 and EUROPA-FP7-231888

The system we present here employs a variety of different methods from machine learning and computer vision, which have been shown to provide good detection rates. We extend these methods obtaining substantial improvements and combine them into a complete system of detection, sensor fusion and object tracking. We use supervised-learning techniques for both kinds of sensor modalities, which extract relevant information from large hand-labeled training data sets. In particular, the major contributions of this work are:

- Several extensions to the vision based object detector by Leibe *et al.* [13] using a feature based voting scheme denoted as Implicit Shape Models (ISM). Our major improvements to ISM are the subdivision of objects into sub-parts to obtain a more differentiated voting, the use of *template masks* to discard unlikely votes, and the definition of *superfeatures* that exhibit a higher evidence of an object's occurrence and are more likely to be found.
- The application and combination of boosted Conditional Random Fields (CRF) for classifying laser scans with the ISM based detector using vision. We use an Extended Kalman Filter (EKF) with multiple motion models to fuse the sensor information and to track the objects in the scene.

This paper is organized as follows. The next section describes work that is related to ours. Sec. 3 gives a brief overview of our overall object detection and tracking system. In Sec. 4, we introduce the implicit shape model (ISM) and present our extensions. Sec. 5 describes our classification method of 2D laser range scans based on boosted Conditional Random Fields. Then, in Sec. 6 we explain our EKF-based object tracker. Finally, we present experiments in Sec. 7 and conclude the paper.

## 2 Related Work

Several approaches can be found in the literature to identify a person in 2D laser data including analysis of local minima [19, 23], geometric rules [24], using maximum-likelihood estimation to detect dynamic objects [10], using AdaBoost on a set of geometrical features extracted from segments [1], or from Delaunay neighborhoods [20]. Most similar to our work is that of Douillard *et al.* [5] who use Conditional Random Fields to classify objects from a collection of laser scans. In the area of vision-based people detection, there mainly exist two kinds of approaches (see [9] for a survey). One uses the analysis of a *detection window* or *templates* [8, 4], the other performs a *parts-based* detection [6, 11]. Leibe *et al.* [13] present a people detector using *Implicit Shape Models* (ISM) with excellent detection results in crowded scenes. In earlier works, we showed already extensions of this method with a better feature selection and an improved nearest neighbor search [21, 22].

Existing people detection methods based on camera *and* laser data either use hard constrained approaches or hand tuned thresholding. Zivkovic and Kröse [25] use a learned leg detector and boosted Haar features from the camera images and employ a parts-based method. However, both their approach to cluster the laser data using

Canny edge detection and the use of Haar features to detect body parts is hardly suited for outdoor scenarios due to the highly cluttered data and the larger variation of illumination. Schulz [18] uses probabilistic exemplar models learned from training data of both sensors and applies a Rao-Blackwellized particle filter (RBPF) to track a person's appearance in the data. However, in outdoor scenarios illumination changes often and occlusions are very likely, which is why contour matching is not appropriate. Also, the RBPF is computationally demanding, especially in crowded environments. Douillard *et al.* [5] also use image features to enhance the object detection but they do not consider occlusions and multiple image detection hypotheses.

### 3 Overview of Our Method

Our system consists of three main components: an appearance based detector that uses the information from camera images, a 2D-laser based detector providing structural information, and a tracking module that uses the combined information from both sensor modalities and provides an estimate of the motion vector for each tracked object. The laser based detection applies a Conditional Random Field (CRF) on a boosted set of geometrical and statistical features of 2D scan points. The image based detector extends the multiclass version of the Implicit Shape Model (ISM)[13]. It only operates on a region of interest obtained from projecting the laser detection into the image to constrain the position and scale of the detected objects. Then, the tracking module applies an Extended Kalman Filter (EKF) with two different motion models, fusing the information from camera and laser. In the following, we describe the particular components in detail.

### 4 Appearance Based Detection

Our vision-based people detector is mostly inspired by the work of Leibe *et al.* [13] on scale-invariant Implicit Shape Models (ISM). In summary, an ISM consists in a set of local region descriptors, called the *codebook*, and a set of displacements and scale factors, usually named *votes*, for each descriptor. The idea is that each descriptor can be found at different positions inside an object and at different scales. Thus, a vote points from the position of the descriptor to the center of the object as it was found in the training data. To obtain an ISM from labeled training data, all descriptors are clustered, usually using agglomerative clustering, and the votes are computed by adding the scale and the displacement of the objects' center to the descriptors in the codebook. For the detection, new descriptors are computed on a test image and matched against the descriptors in the codebook. The votes that are cast by each matched descriptor are collected in a 3D *voting space*, and a maximum density estimator is used to find the most likely position and scale of an object.

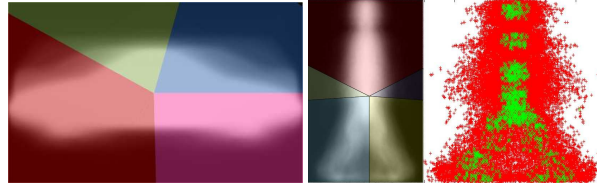
In the past, we presented already several improvements of the standard ISM approach (see [21, 22]). Here, we show some more extensions of ISM to further improve the classification results. These extensions concern both the learning and the detection phase and are described in the following.

#### 4.1 ISM Extensions in the Learning Phase

**Sub-Parts:** The aim of this procedure is to enrich the information from the voters by distinguishing between different object subparts from which the vote was cast. We achieve this by learning a circular histogram of interest points from the training data set for each object class. The number of bins of this histogram is determined automatically by using  $k$ -means clustering. The final number of clusters, here denoted as  $q$ , is obtained using the Bayesian Information Criterion (BIC). Note that this subpart extraction does not guarantee a semantical subdivision of the object (i.e.: legs, arms, etc. for pedestrians) but it is interesting to see that it nevertheless resembles this automatically without manual interaction by the user (see Fig. 1, left and center).

**Template Masks:** In the training data, labeled objects are represented using a binary image named *segmentation mask*. This mask has the size of the object's bounding box and is 1 inside the shape of the object and 0 elsewhere. By overlaying all these masks for a given object class so that their centers coincide and then averaging over them, we obtain a *template mask* of each object class (see Fig. 1, left and center). This method is more robust against noise than, e.g., Chamfer matching [3], and does not depend on an accurate detection of the object contours. We use the template mask later to discard outlier votes cast from unlikely areas.

**Superfeatures:** The original ISM maintains all features from the training data in the codebook as potential voters and does not distinguish between stronger and weaker votes. This has the disadvantage that often too many votes are cast, even if an occurrence of the object is not likely given the training data, and leads to many false positive detections. To overcome this, we propose to extract *superfeatures* from the training data, i.e. descriptor vectors that cast a stronger vote than standard features. We keep these superfeatures in a separate codebook to avoid clutter in the implementation. A superfeature is defined by a local density maximum in descriptor space, where only feature vectors are considered that correspond to interest points from a dense area in the image space (in  $x$ ,  $y$ , and scale). This definition ensures that for superfeatures a high evidence of the occurrence of the object is combined with a high probability to encounter an interest point. We compute superfeatures by first employing mean shift estimation on all interest points found in the training data set for each class, and then clustering the feature vectors in descriptor space that correspond to the interest points from the found areas of high density. This clustering is done agglomeratively. In the end, we select the 50% of the cluster centers that correspond to the biggest clusters. The right part of Fig. 1 shows an example. Note that the superfeatures inherently reflect the skeleton of the object.



**Fig. 1 Left and Center:** Sub-parts, depicted in colored slices, and template masks, in white. They are computed from the training set. Note that even though the subparts are computed unsupervised, they exhibit some semantic interpretation. **Right:** Superfeatures are stable features in image and descriptor space. This figure shows Shape Context descriptors at Hessian interest points (in red) for the class 'pedestrian'. The position of the superfeatures are depicted in green.

## 4.2 ISM Extensions in the Inference Phase

**Sub-Parts and Template Masks:** After collecting all the votes for a given set of extracted input features from a test image, we first discard the ones that are implausible by placing the template mask at the potential object centers and removing the votes that are cast from outside the mask. For the remaining ones we find the maximum density point  $\mathbf{m}$  using mean shift and insert all votes for  $\mathbf{m}$  into a circular histogram with  $q$  bins: one per sub-part of the object. We denote each such histogram as a *hypothesis*  $\mathbf{h} = (h_1, \dots, h_q)$  of an object's position. The *strength*  $\sigma$  of a hypothesis is defined as the sum of all bins, i.e. the number of all voters for the object center. To find the best hypothesis we define a partial order  $\prec$  based on a function  $\Delta_h$ :

$$\mathbf{h}_i \prec \mathbf{h}_j \Leftrightarrow \Delta_h(\mathbf{h}_i, \mathbf{h}_j) < 0 \quad \text{where} \quad \Delta_h(\mathbf{h}_i, \mathbf{h}_j) := \sum_{k=1}^q \text{sign}(h_k^i - h_k^j). \quad (1)$$

Using this, we select the hypothesis with the highest order (in case of ambiguity we use the one with the highest strength) for each class. Then, we find the best hypothesis *across* all classes as described below, remove all its voters and recompute the ordering. This is done until a minimum hypothesis strength  $\sigma_{min}$  is reached. Thus, the parameter  $\sigma_{min}$  influences the number of false positive detections.

**Superfeatures:** Superfeatures and standard features vote for object centers in the same voting space, but the votes from superfeatures are weighted higher (in our case by a factor of 2). Thus, the score of a hypothesis is higher if the fraction of superfeatures voting for it is higher. In some cases where an object's shape visibility is low only superfeatures might be used to obtain a very fast detection.

**Best Inter-Class Hypothesis:** As mentioned above, we need to rate the best object hypotheses from all classes. To be independent on an over- or under-representation of a class in the codebooks, we do this by comparing the relative areas covered by the voters from all class hypotheses. More precisely, we define a square area  $\gamma$  around each voter that depends on the relative scale of the descriptor, i.e. the ratio of the test descriptor's scale and that of the found descriptor in the codebook. The fraction of the area covered by all voters of a hypothesis and the total area of the

object (computed from the template mask) is then used to quantify the hypothesis. Care has to be taken in the case of overlapping class hypotheses. Here, we compute the set intersection of the interest points in the overlapping area and assign their corresponding  $\gamma$  values alternately to one and the other hypothesis.

## 5 Structure Based Detection

For the detection of objects in 2D laser range scans, several approaches have been presented in the past (see for example [1, 16]). Most of them have the disadvantage that they disregard the conditional dependence between data points in a close neighborhood. In particular, they can not model the fact that the label  $l_i$  of a given scan point  $\mathbf{z}_i$  is more likely to be  $l_j$  if we know that  $l_j$  is the label of  $\mathbf{z}_j$  and  $\mathbf{z}_j$  and  $\mathbf{z}_i$  are neighbors. One way to model this conditional dependence is to use Conditional Random Fields (CRFs) [12], as shown by Douillard *et al.* [5]. CRFs represent the conditional probability  $p(\mathbf{y} | \mathbf{z})$  using an undirected cyclic graph, in which each node is associated with a hidden random variable  $l_i$  and an observation  $\mathbf{z}_i$ . In our case, the  $l_i$  is a discrete label that ranges over 3 different classes (pedestrian, car and background) and the observations  $\mathbf{z}_i$  are 2D points in the laser scan. At this point we omit the mathematical details about CRFs and refer to the literature (e.g. [5, 17]). We only note that for training the CRF we use the L-BFGS gradient descent method [14] and for the inference we use max-product loopy belief propagation.

We use a set of statistical and geometrical features  $\mathbf{f}_n$  for the nodes of the CRF, e.g. height, width, circularity, standard deviation, kurtosis, etc. (for a full list see [20]). We compute these features in a local neighborhood around each point, which we determine by jump distance clustering. However, we don't use these features directly in the CRF, because, as stated in [17] and also from our own observation, the CRF is not able to handle non-linear relations between the observations and the labels. Instead, we apply AdaBoost [7] to the node features and use the outcome as features for the CRF. For our particular classification problem with multiple classes, we train one binary AdaBoost classifier for each class against the others. As a result, we obtain for each class  $k$  a set of  $M$  weak classifiers  $u_i$  (decision stumps) and corresponding weight coefficients  $\alpha_i$  so that the sum

$$g_k(\mathbf{z}) := \sum_{i=1}^M \alpha_i u_i(\mathbf{f}(\mathbf{z})) \quad (2)$$

is positive for observations assigned with the class label  $k$  and negative otherwise. We apply the inverse logit function  $a(x) = (1 + e^{-x})^{-1}$  to  $g_k$  to obtain a classification likelihood. Thus, the node features for a scan point  $\mathbf{z}_i$  and a label  $l_i$  are computed as  $\mathbf{f}_n(\mathbf{z}_i, l_i) = a(g_{l_i}(\mathbf{z}_i))$ . For the edge features  $\mathbf{f}_e$  we compute two values, namely the Euclidean distance  $d$  between the points  $\mathbf{z}_i$  and  $\mathbf{z}_j$  and a value  $g_{ij}$  defined as

$$g_{ij}(\mathbf{z}_i, \mathbf{z}_j) = \text{sign}(g_i(\mathbf{z}_i)g_j(\mathbf{z}_j))(|g_i(\mathbf{z}_i)| + |g_j(\mathbf{z}_j)|). \quad (3)$$



This feature has a high value if both  $\mathbf{z}_i$  and  $\mathbf{z}_j$  are equally classified (its sign is positive) and low otherwise. Its absolute value is the sum of distances from the decision boundary of AdaBoost where  $g(\mathbf{z}) = 0$ . Thus, we define the edge features as

$$\mathbf{f}_e(\mathbf{z}_i, \mathbf{z}_j, l_i, l_j) = \begin{cases} (a(d(\mathbf{z}_i, \mathbf{z}_j)) & a(g_{i,j}(\mathbf{z}_i, \mathbf{z}_j)))^T \\ (0 & 0)^T \end{cases} \text{ if } l_i = l_j \quad (4)$$

The intuition behind Eq. (4) is that edges that connect points with equal labels have a non-zero feature value and thus yield a higher potential.

## 6 Object Tracking and Sensor Fusion

To fuse the information from camera and laser and for object tracking we use an Extended Kalman Filter (EKF) as presented in [21]. In our implementation, we use two different motion models – Brownian motion and linear velocity – in order to cope with pedestrian and car movements. The data association is performed in the camera frame: we project the detected objects from the laser scan into the camera image. Assuming a fixed minimal object height, we obtain a rectangular search region, in which we consider all hypotheses from the vision based detector for the particular object class. Using a previously calibrated distance  $r_0$  of an object at scale 1.0 (using the normalized training height), we can estimate the distance  $r_{est}$  of a detected object in the camera image by multiplying  $r_0$  with the scale of the object. Then,  $r_{est}$  is compared to the measured distance  $r_{meas}$  from the laser and both detections are assigned to each other if  $|r_{meas} - r_{est}|$  is smaller than a threshold  $\tau_d$  (in our case  $2m$ ).

We track cluster centers of gravity in the 2D laser frame using two system states:

$$\mathbf{x}_{m1} = \langle (x^{cog}, y^{cog}), (v_x^{cog}, v_y^{cog}), (c_1, \dots, c_n) \rangle \text{ and } \mathbf{x}_{m2} = \langle (x^{cog}, y^{cog}), (c_1, \dots, c_n) \rangle,$$

one for each motion model. Here,  $(v_x^{cog}, v_y^{cog})$  is the velocity of the cluster centroid  $(x_x^{cog}, y_y^{cog})$  and  $c_1, \dots, c_n$  are the probabilities of all  $n$  classes. We use a static state model where the observation vector  $\mathbf{w}$  consists of the position of the cluster and the class probabilities for each sensor modality:

$$\mathbf{w} = \langle \hat{x}^{cog}, \hat{y}^{cog}, (c_1, \dots, c_n)^1, \dots, (c_1, \dots, c_n)^s \rangle. \quad (5)$$

Here,  $(\hat{x}^{cog}, \hat{y}^{cog})$  is a new observation of a cluster center and  $s$  denotes the number of sensors. The matrix  $H$  models the mapping from states to the predicted observation and is defined as  $H = (P^T S_1^T \dots S_s^T)^T$ , where  $P$  maps to pose observations and the  $S_i$  map to class probabilities per sensor. For example, for one laser, one camera and constant velocity we have

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad S_1 = S_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6)$$

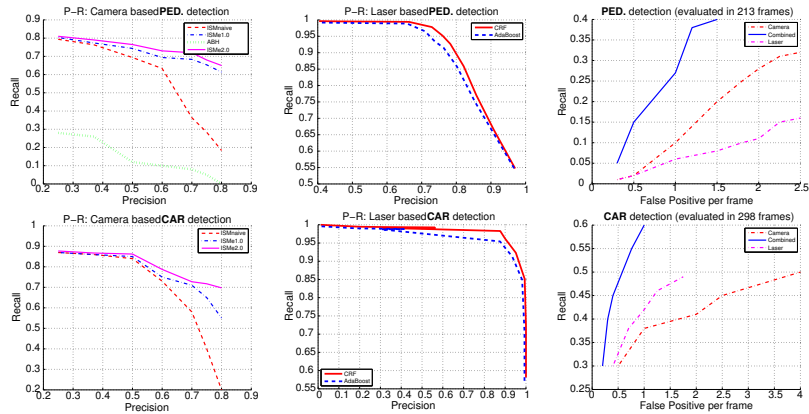


**Fig. 2 Left:** For car classification, we use codebooks from 7 different views. For training, mirrored images are included for each view to obtain a wider coverage. **Center:** For pedestrians we use 2 codebooks of side views with mirroring. Lateral views have sufficient information to generalize frontal/back views. **Right:** Setup used for the city data set. Only a small overlap of the cameras' field of view is used to cover a larger part of the laser scans. No stereo vision is used in this work.

## 7 Experimental Results

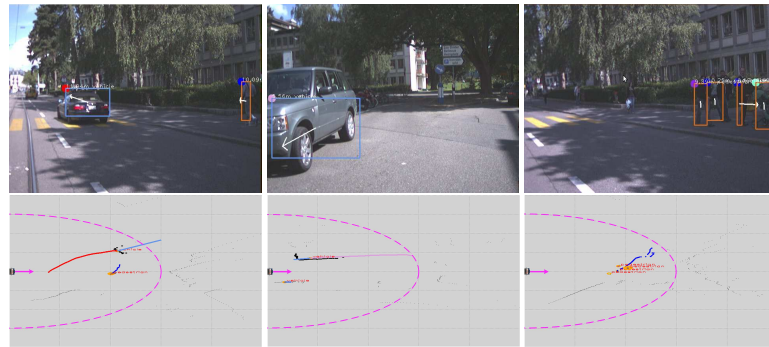
To acquire the data, we used a car equipped with two CCD cameras and a 2D laser range finder mounted in front (see Fig. 2, right). The 3D transform between the laser and the camera coordinate frame was calibrated beforehand. We acquired training data sets for both sensor modalities. For the camera, we collected images of pedestrians and cars that we labeled by hand. The pedestrian data set consists of 400 images of persons with a height of 200 pixels in different poses and with different clothing and accessories such as backpacks and hand bags in a typical urban environment. The class 'car' was learned from 7 different viewpoints as in [13] (see also Fig. 2, left). Each car data set consists of 100 pictures from urban scenes with occlusions. Car codebooks are learned using Shape Context (SC) descriptors [2] at Hessian-Laplace interest points [15]. The pedestrian codebook uses lateral views and SC descriptors at Hessian-Laplace and Harris-Laplace interest points for more robustness. Experience shows [13] that lateral views of pedestrians also generalize well to front/back views. Our laser training data consists of 800 annotated scans with pedestrians, cars and background. There is no distinction of car views in the laser data as the variation in shape is low. The range data consists in 4 layers where each has an angular resolution of  $0.25^\circ$  and a maximum range of  $15m$ .

To quantify the performance of our detector we acquired two datasets containing cars and pedestrians. The results of our detection algorithm are shown in Fig. 3. Our vision based detection named ISMe2.0 is compared to the standard ISM, our previous extension ISMe1.0, and for the pedestrian class, with AdaBoost trained on Haar features (ABH). For the class 'car', we averaged the results over all different views. We can see that our method yields the best results with an Equal Error Rate (EER) of 72.3% for pedestrians and 74% for cars. The improvements are mainly due to a decreased rate of false positive detections. The results of our laser based detection are shown in the middle column of Fig. 3. We can see that our approach using boosted CRFs performs better than standard AdaBoost. The right column of Fig. 3 depicts the results for the combined detection using laser and vision. These graphs clearly show that using both sensors the number of false positive detections decreases and the hit rate increases. Some qualitative results are shown in Fig. 4 where a passing car and a crossing pedestrian are correctly detected and tracked.



**Fig. 3** Quantitative evaluation. **Upper row:** pedestrian detection, **Lower row:** car detection. From left to right we show the results only using camera, only using laser, and both. As we can see, our approach outperforms the other methods for both sensor modalities. The image based detection is compared with standard ISM, our first extension of ISM (ISMe1.0) and AdaBoost with Haar features. Our CRF-based laser detector is compared with AdaBoost. We can also see that the combination of both sensors improves the detection result of both single sensors.

In addition, we evaluated our algorithm on a third, more challenging dataset acquired in the city of Zurich. It consists of 4000 images and laser scans. The equal error rates of this experiment resulted in 64.1% (laser-only), 64.1% (vision-only) and 68% (combined) for pedestrians, and in (72.2%, 73.5%, 75.7%) for cars. As a comparison, we evaluated the state-of-the-art pedestrian detector based on Histogram of Oriented Gradients [4] and ABH obtained an EER of 36.4 and 8.9.



**Fig. 4** Cars and pedestrian detected and tracked under occlusion, clutter and partial views. In the camera images, upper row, blue boxes indicate car detections, orange boxes pedestrian detections. The colored circle on the upper left corner of each box is the track identifier. Tracks are shown in color in the second row and plotted with respect to the robot reference frame.

## 8 Conclusions

We presented a method to reliably detect and track multiple object classes in outdoor scenarios using vision and 2D laser range data. We showed that the overall performance of the system is improved using a multiple-sensor system. We presented several extensions to the ISM based image detection to cope with multiple classes. We showed that laser detection based on CRFs performs better than a simpler AdaBoost classifier and presented tracking results on combined data. Finally, we showed the usefulness of our approach through experimental results on real-world data.

## References

1. K. O. Arras, Ó. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2007.
2. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 24(4):509–522, 2002.
3. G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 10(6):849–865, 1988.
4. N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.
5. B. Douillard, D. Fox, and F. Ramos. Laser and vision based outdoor object mapping. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.
6. P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, pages 66–73, 2000.
7. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
8. D. Gavrila and V. Philomin. Real-time object detection for “smart” vehicles. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.
9. D. M. Gavrila. The visual analysis of human movement: A survey. *Comp. Vis. and Image Und. (CVIU)*, 73(1):82–98, 1999.
10. D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2003.
11. S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. Journ. of Comp. Vis.*, 43(1):45–68, 2001.
12. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *Int. Conf. on Machine Learning (ICML)*, 2001.
13. B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2007.
14. D. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)), 1989.
15. K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.
16. C. Premebida, G. Monteiro, U. Nunes, and P. Peixoto. A lidar and vision-based approach for pedestrian and vehicle detection and tracking. In *ITSC*, 2007.
17. F. Ramos, D. Fox, and H. Durrant-Whyte. CRF-matching: Conditional random fields for feature-based scan matching. In *Robotics: Science and Systems (RSS)*, 2007.
18. D. Schulz. A probabilistic exemplar approach to combine laser and vision for person tracking. In *Robotics: Science and Systems (RSS)*, 2006.
19. D. Schulz, W. Burgard, D. Fox, and A. Cremers. People tracking with mobile robots using sample-based joint probabilistic data ass. filters. *Int. Journ. of Rob. Res. (IJRR)*, 22(2), 2003.

20. L. Spinello and R. Siegwart. Human detection using multimodal and multidimensional features. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2008.
21. L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2008.
22. L. Spinello, R. Triebel, and R. Siegwart. Multimodal people detection and tracking in crowded scenes. In *Proc. of the AAAI Conf. on Artificial Intelligence*, July 2008.
23. E. A. Topp and H. I. Christensen. Tracking for following and passing persons. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2005.
24. J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2005.
25. Z. Zivkovic and B. Kröse. Part based people detection using 2d range data and images. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, San Diego, USA, November 2007.

## A Layered Approach to People Detection in 3D Range Data

Luciano Spinello<sup>a,b</sup>

Kai O. Arras<sup>a</sup>

Rudolph Triebel<sup>b</sup>

Roland Siegwart<sup>b</sup>

<sup>a</sup>Social Robotics Lab, University of Freiburg, Germany

<sup>b</sup>Autonomous Systems Lab, ETH Zurich, Switzerland

### Abstract

People tracking is a key technology for autonomous systems, intelligent cars and social robots operating in populated environments. What makes the task difficult is that the appearance of humans in range data can change drastically as a function of body pose, distance to the sensor, self-occlusion and occlusion by other objects. In this paper we propose a novel approach to pedestrian detection in 3D range data based on supervised learning techniques to create a bank of classifiers for different height levels of the human body. In particular, our approach applies AdaBoost to train a strong classifier from geometrical and statistical features of groups of neighboring points at the same height. In a second step, the AdaBoost classifiers mutually enforce their evidence across different heights by voting into a continuous space. Pedestrians are finally found efficiently by mean-shift search for local maxima in the voting space. Experimental results carried out with 3D laser range data illustrate the robustness and efficiency of our approach even in cluttered urban environments. The learned people detector reaches a classification rate up to 96% from a single 3D scan.

### 1. Introduction

Robustly detecting pedestrians is a key problem for mobile robots and intelligent cars. Laser range sensors are particularly interesting for this task as, in contrast to vision, they are highly robust against illumination changes and typically provide a larger field of view.

In this paper we address the problem of detecting pedestrians in 3D range data. The approach presented here uses techniques from people detection in 2D range data for which a large amount of related work exists (Kluge, Köhler, and Prassler, 2001; Fod, Howard, and Mataric, 2002; Schulz et al., 2003; Cui et al., 2005; Arras, Martínez Mozos, and Burgard, 2007). In early works, people are detected using ad-hoc classifiers, looking for moving local minima in the scan. Learning has been applied for this task by Arras, Martínez Mozos, and Burgard (2007) where a classifier for 2D point clouds has been learned by boosting a set of geometrical features. As there is a natural performance limit for people detection in a *single* 2D layer of range data, several authors started looking into the use of multiple co-planar 2D

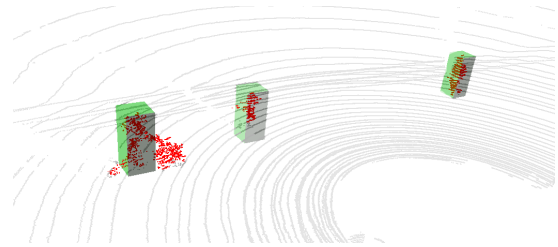


Figure 1: 3D pedestrian detection. A person pushing a buggy, a child and a walking pedestrian are correctly identified in the point cloud.

laser scanners Gidel et al. (2008); Carballo, Ohya, and Yuta (2008). Close to our context is the work of Mozos, Kurazume, and Hasegawa (2010), in which the authors apply boosting on each of three horizontal layers and use a probabilistic rule set to combine the three classifiers assuming a known ground plane.

There is little related work on pedestrian detection in 3D data. Navarro-Serment, Mertz, and Hebert (2009) collapse the 3D scan into a virtual 2D slice to find salient vertical objects above ground. For these objects, they align a window to the principal data direction, compute a set of features, and classify pedestrians using a set of SVMs. Bajracharya et al. (2009) detect people in point clouds from stereo vision by processing vertical objects and considering a set of geometrical and statistical features of the cloud based on a fixed pedestrian model. From the comprehensive body of literature on people detection in images, we mention the most related ones, namely the HOG detector by Dalal and Triggs (2005) and the ISM approach by Leibe, Seemann, and Schiele (2005). People detection from multimodal data using laser and vision has been presented by Spinello, Triebel, and Siegwart (2008).

To our knowledge, this work presents the first principled learning approach to people detection in 3D range data. The idea is to subdivide a pedestrian into parts defined by different height levels, and learn a highly specialized classifier for each part. We exploit the fact that most of the commercial 3D laser devices retrieve the environment as a set of individual *scan lines* which are not necessarily co-planar.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

The building blocks used for classification are *segments*, i.e. groups of consecutive points in each scan line, on which a set of geometrical and statistical features are computed. We do not define a 3D pedestrian shape model beforehand, but instead learn it from labeled data by storing the displacements between the segment centers and the person's center. This allows for a general and robust description for articulated and complex 3D objects. Then, each segment is classified based on the likelihood of belonging to each part. We relate the output of each classifier geometrically by employing a 3.5D voting approach where each segment votes for the center of a person. Areas of high density in the continuous voting space define hypotheses for the occurrence of a person. This allows for robustness against occlusions as not all parts are needed for a detection. Moreover, our approach does not rely on any ground plane extraction heuristics and does not require any motion cues. No tracking is done in this work.

The paper is structured as follows: Sec. 2 explains the preprocessing steps we apply to the 3D data. Sec. 3 describes how we subdivide and learn a 3D person model from data. In Sec. 4 the detection step is presented. Sec. 5 contains the experimental results and Sec. 6 concludes the paper.

## 2. Preprocessing 3D Range Data

Different systems exist to acquire 3D range data from the environment. Many of them rely on a scanning device that sends out laser rays and measures the distance to the closest object. To acquire a 3D scan, such devices are usually rotated about one of the main axes of the sensor-based coordinate frame. Examples include 2D range finders such as the SICK LMS laser scanner, mounted on a turntable that rotates about its vertical or horizontal axis (Lamon, Kolski, and Siegart, 2006). Other devices, such as the Velodyne HDL-64E, also rotate about the  $z$ -axis sending out 64 independent laser beams that are not coplanar. The Alasca XT rangefinder uses a beam deflected by a rotating mirror and 4 receivers. Such sensors return point clouds that consist of individual *scan lines*, i.e. sequences of points that have been measured with the same beam. With some abstraction, we can thus think of such a 3D point cloud as a collection of 2D laser points arranged in *slices* or *layers*. This definition holds also for a wide set of non-laser sensors: range camera data (e.g. Swissranger) or point cloud data from stereo cameras can also be transformed into sets of scan lines by horizontally sampling image pixels.

Formally, we consider a point cloud  $\mathcal{X}$  as consisting of layers  $\mathcal{L}_i = \{\mathbf{x}_{ij}\}$ , where  $\mathbf{x}_{ij} = (x_{ij}, y_{ij}, z_{ij})$ . In this paper, we demonstrate that by treating a 3D scan as a collection of 2D scans at different levels, known and proven techniques for detecting people in 2D range data can be easily extended to the 3D case, yielding a fast and robust people detector for 3D range data.

### 2.1 Point Cloud Segmentation per Layer

As a first step of our detection algorithm, we divide each scan line into *segments* using Jump Distance Clustering (JDC). JDC initializes a new segment each time the distance

Nr	Feature Name	Nr	Feature Name
$f_1$	Width	$f_2$	Number of points
$f_3$	Circularity	$f_4$	Linearity
$f_5$	Boundary length	$f_6$	Boundary regularity
$f_7$	Mean angular difference	$f_8$	Mean curvature
$f_9$	Quadratic spline fitting	$f_{10}$	Cubic spline fitting
$f_{11}$	Standard dev. w.r.t. centroid	$f_{12}$	Mean avg. dev. from median
$f_{13}$	Kurtosis w.r.t. centroid	$f_{14}$	Radius
$f_{15}$	PCA ratio	$f_{16}$	Bounding box area
$f_{17}$	Convex hull area		

Table 1: Features used to describe the shape and statistical properties of a segment.

between two consecutive points exceeds a threshold  $\theta_d$ . As a result, the data is reduced to a smaller number of segments with a higher amount of information than that of the raw data points. We denote each segment as a set  $\mathcal{S}_j, j = 1, \dots, N_i$  of consecutive points where  $N_i$  is the number of segments in scan line  $i$ . Our algorithm assumes that the 3D scanner rotates about the vertical  $z$ -axis, which means that the points in a segment are sorted by ascending azimuth angles. The segments constitute the primal element to extract local information.

### 2.2 Segment shape characterization

In the next step, we compute several *descriptors* for each extracted segment. A descriptor is defined as a function  $f_k: \mathcal{S}_j \rightarrow \mathbb{R}$  that takes the  $M$  points contained in a segment  $\mathcal{S}_j = \{(x_1, y_1, z_1) \dots (x_M, y_M, z_M)\}$  as an input argument and returns a real value. Most of the features we use ( $f_1 \dots f_8$ ) have been presented by Arras, Martínez Mozos, and Burgard (2007) and Spinello, Triebel, and Siegart (2008), the following ones ( $f_9 \dots f_{17}$ ) are added for this particular task:

- *Quadratic spline fitting*: this feature measures the residual sum of squares of a quadratic B-Spline regression  $s_2$  (a piecewise polynomial approximation introduced by De Boor (1978)) of the points in  $\mathcal{S}_j$ :  $f_9 = \sum_i (s_2(x_i, y_i) - y_i)^2$
- *Cubic spline fitting*: this feature measures the residual sum of squares of a cubic B-Spline regression  $s_3$  of the points in  $\mathcal{S}_j$ , i.e.  $f_{10} = \sum_i (s_3(x_i, y_i) - y_i)^2$
- *Kurtosis with respect to centroid*: the kurtosis is defined as the fourth standardized moment of the cluster  $\mathcal{S}_j$ , i.e.  $f_{12} = \frac{\sum_i (\mathbf{x}_i - \hat{\mathbf{x}}_j)^4}{M \cdot f_{11}^2}$ , where  $f_{11}$  represents the standard deviation with respect to the centroid, and  $\hat{\mathbf{x}}_j$  the center of gravity of  $\mathcal{S}_j$ .
- *PCA ratio*: this feature is the ratio between the second biggest eigenvalue  $\lambda_2$  and the biggest eigenvalue  $\lambda_1$  of the scatter matrix associated with  $\mathcal{S}_j$ . It measures the aspect ratio of the oriented bounding box, i.e.  $f_{13} = \frac{\lambda_2}{\lambda_1 + 1}$
- *Bounding box area*: this feature represents the area of the axis-aligned bounding box of  $\mathcal{S}_j$ .
- *Convex hull area*: this feature represents the area computed from the convex hull polygon extracted from  $\mathcal{S}_j$ .

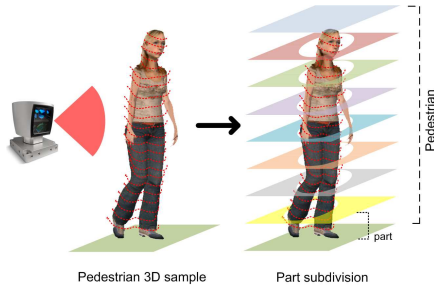


Figure 2: Learning a 3D person model. Objects are vertically divided into  $K$  parts. For each part an independent AdaBoost classifier is learned: all the segmented points contained on each scan line are considered as positive samples for the  $K$  AdaBoost classifiers.

Table 1 lists all 17 used features. The set of feature values of each segment  $S_i$  then forms a vector  $\mathbf{f}_i = (f_1, \dots, f_{17})$ .

### 3. Learning a 3D Model of People

The appearance of people is highly variable. Humans have different sizes and body shapes, wear clothes, carry bags, backpacks, or umbrellas, pull suitcases, or push buggies. This makes it hard to predefine models for their appearance and motivates a learning approach based on a model that is created from acquired data.

#### 3.1 Definition of Parts

We tackle the problem of the shape complexity of humans by a subdivision into different height layers or *parts* (see Fig. 2). The subdivision is defined beforehand and does not follow an anatomical semantics like legs, trunk, head. Results from computer vision literature (Dalal and Triggs, 2005; Zhu et al., 2006; Viola and Jones, 2002) and also our own experience show that descriptors computed in geometrically overlapping tessellations are powerful tools for learning an object model. Therefore, for learning a 3D person model we create  $K$  different and independent classifiers, each corresponding to a height-divided part of a human.

For training, all the scan lines that fall within a part are considered. The model is learned from subjects with similar heights (within  $\pm 15\text{cm}$  from the mean). As part classifier we use AdaBoost (Freund and Schapire, 1997), a well known machine learning algorithm, that has been proven successful for people detection in 2D range data (Arras, Martínez Mozos, and Burgard, 2007).

#### 3.2 Learning the Part Detectors

AdaBoost is a general method for creating an accurate strong classifier by combining a set of weighted weak classifiers, in this case decision stumps. A decision stump  $h_t$  defines a single axis-parallel partition of the feature space. The final strong classifier  $H(\mathbf{f})$  computed for the feature vector  $\mathbf{f}$  is a

weighted sum of the  $T$  best weak classifiers:

$$H(\mathbf{f}) = \text{sign} \left( \sum_{t=1}^T \alpha_t h_t(\mathbf{f}) \right), \quad (1)$$

where  $\alpha_t$  are the weights learned by AdaBoost.

As people are usually represented only by a few number of data points in a 3D scan, there are many more background segments than segments on people. This makes the training set unbalanced. Now, instead of down-sampling the negative set, which could lead to an under-representation of the feature distribution, we use an adaptive initial weight vector  $\mathbf{w}_0 \in \mathbb{R}^N$  where  $N$  is the total number of training segments. Usually  $\mathbf{w}_0$  is set to a uniform distribution, i.e.  $1/N \cdot \mathbf{1}^N$ , where  $\mathbf{1}^N$  is the vector of dimension  $N$  with all entries equal to 1. Instead we use

$$\mathbf{w}^p := \frac{1}{2N_{pos}} \mathbf{1}^{N_{pos}}, \quad \mathbf{w}^n := \frac{1}{2N_{neg}} \mathbf{1}^{N_{neg}}, \quad \mathbf{w}_0 = (\mathbf{w}^p, \mathbf{w}^n), \quad (2)$$

where  $N_{pos}, N_{neg}$  are the numbers of positive and negative training samples. Thus, the bigger training set – in our case the negative set – obtains a smaller weight.

To avoid early hard decisions in the classification of segments, we apply a sigmoid to the classification result in Eq. (1). This can be interpreted as a measure of likelihood  $p(\pi^k | \mathbf{f}_i)$  of a segment  $i$ , represented by its feature vector  $\mathbf{f}_i$ , of corresponding to a part  $\pi^k$  of a pedestrian:

$$g_k(\mathbf{f}_i) = \frac{\sum_{t=1}^T \alpha_t^k h_t^k(\mathbf{f}_i)}{\sum_{t=1}^T \alpha_t^k}, \quad p(\pi^k | \mathbf{f}_i) = \left( 1 + e^{2-13g_k(\mathbf{f}_i)} \right)^{-1}, \quad (3)$$

where  $g_k$  is the normalized sum of weak classification results of the  $k$ -th classifier and  $\pi^k$  is a binary label that is true if segment  $i$  corresponds to part  $k$ .

In our case we need to classify one part against all others and the background. We therefore face a multi-class classification problem for which we follow a *one-vs-all* strategy: when training a part, all the features of the segments contained in that part are considered positive samples, the features of the background and of the other parts are tagged as negative samples.

#### 3.3 Learning Geometric Relations

So far we described a way to classify parts of a person, now we combine the individual classifications into a full person detector. In computer vision, this problem is addressed using *part constellations* (Fergus, Perona, and Zisserman, 2003), Conditional Random Fields (CRF) (Felzenszwalb and Huttenlocher, 2005), or implicit shape models (ISM) (Leibe, Seemann, and Schiele, 2005). Loosely inspired from the latter, we propose the following voting model.

First, we use the 3D displacement information of segments to define the geometric relations that constitute a 3D person model. Each part and each segment found in a part are considered independently. For a segment  $S_i$  found in part  $\pi^k$  in the training set, we store the 3D displacement vector  $\mathbf{v}_i^k$ , also called ‘vote’, i.e. the difference between the center of the person and the center of  $S_i$ . Then all votes for part  $\pi^k$  are collected in a set  $\mathcal{V}^k$ . This information implicitly resembles different body poses of humans. For instance,



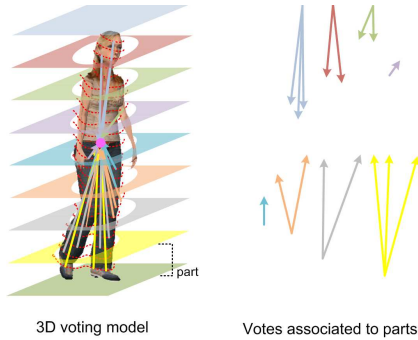


Figure 3: Learning a 3D voting model for training the detector. The displacement, or vote, between a segment and the person center of gravity in 3D associated to each subpart is stored as  $\mathcal{V}^k$ . Agglomerative clustering is carried out to obtain a more compact representation of  $\mathcal{V}^k$ .

if the training data is acquired during a walking sequence, the segments contained in the lowest body part (associated to feet/ankles) typically have a larger variation of displacements with respect to the center of gravity, i.e. the votes are spread out wider. After all person samples have been processed, a large amount of votes for each part is stored. The final step is then to compress  $\mathcal{V}^k$  for each part into a new set  $\hat{\mathcal{V}}^k$ . This is achieved using agglomerative clustering with average linkage and a distance threshold  $\theta_v$  (see Figure 3). Then, a weight  $\hat{w}_i^k = |\hat{\mathcal{V}}^k|^{-1}$  is assigned to each clustered vote  $\hat{v}_i^k$  of  $\hat{\mathcal{V}}^k$ , where  $|\hat{\mathcal{V}}^k|$  denotes the number of vote clusters for part  $\pi^k$ . The intuition here is that parts with a higher displacement variability imply a lower voting confidence. Finally, to obtain a practical geometrical interpretation of people in 3D, we compute the average bounding box from all person samples.

#### 4. Detecting People in 3D

Our detection method processes a single point cloud as input and retrieves as output a set of detected people. After acquiring a new 3D scan, it is processed by the JDC segmentation step for each scan layer. Then, the feature vector  $\mathbf{f}_i$  is computed for each segment found in each part. The likelihood of a segment to belong to a part is obtained by classifying the features with the corresponding AdaBoost model. Thus, we obtain a vector

$$\mathbf{c}_i = (p(\pi^1 | \mathbf{f}_i), \dots, p(\pi^K | \mathbf{f}_i)). \quad (4)$$

This defines a multiple weighted part hypothesis, therefore we need to find a way of properly treating this information. Here, we use the voting model learned in the training phase (see Section 3) to generate hypotheses of person centers in 3D. It is important to note that no assumptions about the position of the ground plane are done in this work.

We formulate a 3.5D continuous voting space procedure in which each segment  $\mathcal{S}_i$  casts a set of non-negative weighed votes  $\mathcal{V}_1, \dots, \mathcal{V}_K$  in 3D, where  $K$  is the number of

pedestrian subparts. Each vote set  $\mathcal{V}_m$  is weighted by the subpart classification likelihood of equation (4):

$$\rho(k) = \frac{\mathbf{c}_i(k)}{K} \quad (5)$$

where  $\mathbf{c}_i(k)$  is the value of the  $m$ -element of the vector (4) and  $K$  the number of subparts.

All generated votes are collected in a continuous voting space  $\mathcal{W}$ . High density loci represent hypotheses of pedestrians centers in 3D. Therefore, we estimate the modes of the voting space distribution. This is achieved using Mean Shift estimation (Comaniciu and Meer, 2002) with a spherical uniform kernel. Mean shift locates stationary points of a density function given discrete data sampled from that function. As soon as a mode is found, its score is computed:

$$score(\mathbf{x}_k | \mathcal{W}) = \left[ \sum_j^{\mathcal{N}(\mathbf{x}_k)} \hat{v}_j^{\epsilon(v_j)} \rho(\epsilon(v_j)) \right] \frac{\zeta(\mathcal{N}(\mathbf{x}_k))}{K}, \quad (6)$$

where  $\mathbf{x}_k$  is a converged mode and  $\mathcal{N}(\mathbf{x}_k)$  contains all the indices of the votes contributing to the basin of attraction of  $\mathbf{x}_k$ ,  $\epsilon(v_j)$  is a function that returns the part index of the vote  $v_j$ ,  $\hat{v}_j$  the weight value of vote  $v_j$ .  $\zeta(\cdot)$  is a function that returns the number of parts from which the votes are originated. Thus,  $\zeta(\cdot)$  is a modifier that favors people that are explained by more parts than others and it is very useful to decrease strong false positives that receive vote from clutter at the same height. The higher the hypothesis score of equation 6, the higher the likelihood of detecting people in  $\mathbf{x}_k$ . It is important to notice that this approach implements a form of simultaneous detection and segmentation: votes contributing to a hypothesis identify segments that belong to a person.

## 5. Experiments

We evaluate our algorithm on two outdoor data sets collected with a Velodyne HDL 64E S2 laser scanner. The first data set, named *Polyterrasse*, has been collected in a large area in the front of the ETH Zurich main building, accessible only to people and bicycles. The second data set, named *Tannenstrasse*, has been collected on a busy street crossing in downtown Zurich with trams, cars, pedestrians, or bicycles.

We collected 900 full-view point clouds for the first set and 500 for the second set. The sensor rotates with a frequency of 5Hz at a maximum range limited to 20m. This produces around 120,000 points per 3D scan. In each frame, people are manually annotated by a bounding box if they are represented by at least 200 points and exceed 1.20m in height. A second type of annotations are made for people represented by at least 100 points and 1m height.

### 5.1 Training

We train with 203 persons, standing still and walking. We define a subdivision of 9 parts at different heights given in Table 2. The vote clustering threshold is set to  $\theta_v = 25cm$ , the JDC threshold is set to  $\theta_d = 40cm$ . Training has been done on 2592 background segments and 7075 people segments from the *Polyterrasse* data set.

Part number	Part height	# of votes	# pos. training segments
1	[0m, 0.2m]	7	500
2	[0.2m, 0.4m]	9	814
3	[0.4m, 0.6m]	4	751
4	[0.6m, 0.8m]	3	811
5	[0.8m, 1.0m]	6	866
6	[1.0m, 1.2m]	3	881
7	[1.2m, 1.4m]	3	903
8	[1.4m, 1.6m]	3	917
9	[1.6m, 2.5m]	2	632

Table 2: Person parts subdivision and vote set.

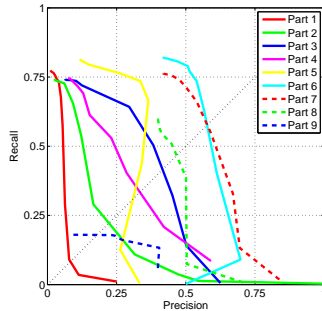


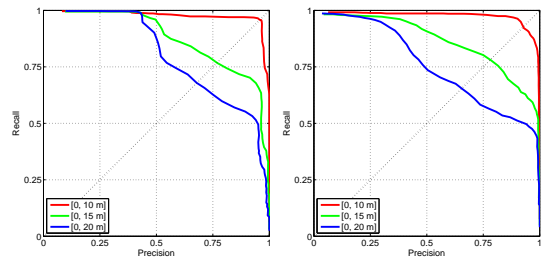
Figure 4: Precision-recall curve of the individual part detectors under the assumption of a known ground plane. Upper body parts (nr. 6-8) show better performance partly due to a better separation from the background, a higher point density, and a smoother shape.

The right-most column in Table 2 gives the resulting number of training samples for each part. The third column in Table 2 contains the number of votes for each part after the vote clustering step. Note that lower parts, related to legs, return more votes due to the varying displacements of the segments during walking. Only 20 decision stumps have been learned to avoid overfitting.

## 5.2 Quantitative Performance Evaluation

We evaluated each part detector on 440 frames not in the training set. To analyze the performance of the individual classifiers, we assumed to know the ground plane and selected the correct detector for each scan line. Figure 4 shows the precision-recall curve for each part classifier. It is interesting to see that the individual detection performances are rather poor. The detectors show a high recall behavior, high detection rates cause big quantities of false positives.

Figure 5 shows the overall precision-recall graph of the proposed method applied to both data sets. Detections are counted as true positives if the bounding box overlaps with a manually labeled person by more than 60% to account for metric inaccuracies in the annotation and the detection. Adopting the no-reward-no-penalization policy from Enzweiler and Gavrila (2009), when a detection matches an annotation of the second type, no true positives or no false positives are counted.

Figure 5: Evaluation for 3D people detection. Each figure depicts precision-recall graphs at different ranges: from 0 to 10m, 15m, and 20m. **Left:** Detection performance for the *Polyterrasse* data set, the Equal Error Rates (EER) are 96%, 71%, 65%. **Right:** Precision-recall graph for the *Tannenstrasse* data set, EER values are 95%, 76%, 63%.

The performance increase over the individual part detectors is significant. The false positive rate is greatly decreased while the true positive rate is increased. This means that the part classifiers are diverse and complementary: if some parts do not return positive classifications, others do. This property is likely to explain the result shown in Fig. 1 where the detector correctly finds the child although no child was in the training set.

The figure also shows how the detection performance decreases with distance from the sensor. For the *Polyterrasse* data set, the Equal Error Rate (EER) for ranges between 0 to 10m is 96%, to 15m it is 71% and to 20m it is 65%. For the *Tannenstrasse* data set, the respective numbers are 95%, 76%, and 63%. The decay is mainly caused by point sparsity that leads to oversegmentation and less distinctive descriptors. This loss of detail renders the distinction of people from vertical structures of similar size such as traffic signs or pillars more difficult. The overall performance is comparable in both data sets although the *Polyterrasse* environment is well structured while the crossing of the *Tannenstrasse* is a rather busy place with clutter and all kinds of dynamic objects. Note that the person model was learned only with data from the *Polyterrasse* environment.

In our evaluation set, people are described by 1062 points on average when they are in a range of 0 to 10m, by 557 points in a 15m range and by 290 points in a 20m range. Therefore, a 73% decrease in the number of points, from 10m to 20m range, causes only a 23% performance loss. Fig. 6 shows the detection results of two example frames.

A C++ implementation of the proposed method, not optimized for speed, obtains  $\sim 1\text{Hz}$  detection frequency in an average 3D scan, limited to 10m maximum range, of around 75,000 points.

## 6. Conclusions

In this paper we presented a principled learning approach to people detection in 3D range data. Our approach subdivides a person into parts at different height levels. For each part a specialized AdaBoost classifier is created from a set of geometrical and statistical features computed on segments. The

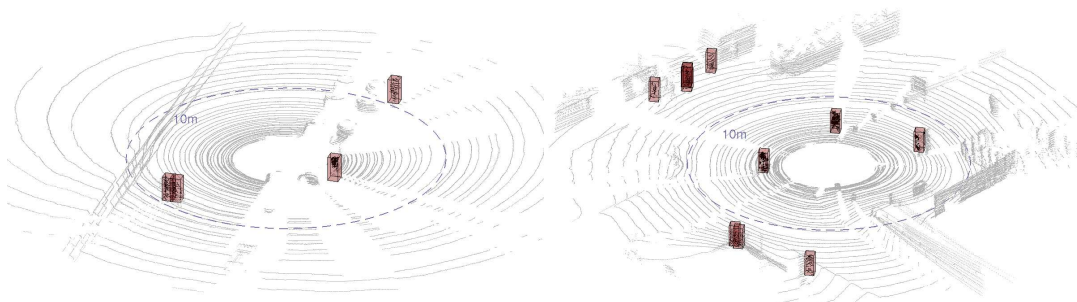


Figure 6: Two example frames showing detection results as red boxes. **Left:** A frame of the *Polyterrasse* data set. Closely walking people and a partly visible individual standing close to the sensor are correctly found. **Right:** A frame of the *Tannenstrasse* data set showing a cluttered urban street crossing with people, pillars, street signs, and a tram just entering the scene. People are correctly detected while crossing the street and walking at a large distance to the sensor. There are two false positives in the lower left part of the picture caused by a glass window with vertical steel poles.

classifiers mutually enforce their evidence across different heights by voting into a continuous space. This approach allows for the detection of people from partial views and does not require knowledge of the ground plane. In experiments with two different data sets in cluttered urban environments, a classification rate up to 96% has been achieved which is a high value given that we detect people from a single 3D scan. In future work, we plan to combine this method with tracking to integrate detection results over time.

#### Acknowledgements

This work has been supported by the German Research Foundation (DFG) under contract number SFB/TR-8 and EU Project EUROPA-FP7-231888.

#### References

- Arras, K. O.; Martínez Mozos, O.; and Burgard, W. 2007. Using boosted features for the detection of people in 2d range data. In *Int. Conf. on Rob. & Autom. (ICRA)*.
- Bajracharya, M.; Moghaddam, B.; Howard, A.; Brennan, S.; and Matthies, L. 2009. Results from a real-time stereo-based pedestrian detection system on a moving vehicle. In *Workshop on People Detection and Tracking, IEEE ICRA*.
- Carballo, A.; Ohya, A.; and Yuta, S. 2008. Fusion of double layered multiple laser range finders for people detection from a mobile robot. In *IEEE Int. Conf. on Multisensor Fusion and Integration for Intelligent Systems (MFI)*.
- Comaniciu, D., and Meer, P. 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. on Pattern Analysis & Machine Intell.* 24:603–619.
- Cui, J.; Zha, H.; Zhao, H.; and Shibasaki, R. 2005. Tracking multiple people using laser and vision. In *Int. Conf. on Intel. Rob. and Sys. (IROS)*.
- Dalal, N., and Triggs, B. 2005. Histograms of oriented gradients for human detection. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*.
- De Boor, C. 1978. *A Practical Guide to Splines*. Springer-Verlag.
- Enzweiler, M., and Gavrila, D. 2009. Monocular pedestrian detection: Survey and experiments. *IEEE Trans. on Pattern Analysis & Machine Intell.* 31(12):2179–2195.
- Felzenszwalb, P., and Huttenlocher, D. 2005. Pictorial structures for object recognition. *Int. Journ. of Comp. Vis.* 2:66–73.
- Fergus, R.; Perona, P.; and Zisserman, A. 2003. Object class recognition by unsupervised scale-invariant learning. *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*.
- Fod, A.; Howard, A.; and Mataric, M. 2002. Laser-based people tracking. In *Int. Conf. on Rob. & Autom. (ICRA)*.
- Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Jour. of Comp. and System Sciences* 55(1).
- Gidel, S.; Checchin, P.; Blanc, C.; Chateau, T.; and Trassoudaine, L. 2008. Pedestrian detection method using a multilayer laser-scanner: Application in urban environment. In *Int. Conf. on Intel. Rob. and Sys. (IROS)*.
- Kluge, B.; Köhler, C.; and Prassler, E. 2001. Fast and robust tracking of multiple moving objects with a laser range finder. In *Int. Conf. on Rob. & Autom. (ICRA)*.
- Lamon, P.; Kolski, S.; and Siegwart, R. 2006. The smarter - a vehicle for fully autonomous navigation and mapping in outdoor environments. In *CLAWAR*.
- Leibe, B.; Seemann, E.; and Schiele, B. 2005. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*.
- Mozos, O. M.; Kurazume, R.; and Hasegawa, T. 2010. Multi-part people detection using 2d range data. *International Journal of Social Robotics* 2(1).
- Navarro-Serment, L.; Mertz, C.; and Hebert, M. 2009. Pedestrian detection and tracking using three-dimensional lidar data. In *Int. Conf. on Field and Service Robotics*.
- Schulz, D.; Burgard, W.; Fox, D.; and Cremers, A. 2003. People tracking with a mobile robot using sample-based joint probabilistic data association filters. *Int. Journ. of Rob. Research* 22(2):99–116.
- Spinello, L.; Triebel, R.; and Siegwart, R. 2008. Multimodal people detection and tracking in crowded scenes. In *AAAI Conf. on Artif. Intell. (AAAI)*.
- Viola, P., and Jones, M. 2002. Robust real-time object detection. *Int. Journ. of Comp. Vis.*
- Zhu, Q.; Yeh, M. C.; Cheng, K. T.; and Avidan, S. 2006. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*.

# Multiclass Multimodal Detection and Tracking in Urban Environments

Luciano Spinello<sup>†, §</sup>      Rudolph Triebel<sup>§</sup>      Roland Siegwart<sup>§</sup>

<sup>†</sup>Social Robotics Lab, University of Freiburg, Germany

<sup>§</sup>Autonomous Systems Lab, ETH Zurich, Switzerland

## Abstract

This paper presents a novel approach to detect and track people and cars based on the combined information retrieved from a camera and a laser range scanner. Laser data points are classified by using boosted Conditional Random Fields (CRF), while the image based detector uses an extension of the Implicit Shape Model (ISM), which learns a codebook of local descriptors from a set of hand-labeled images and uses them to vote for centers of detected objects. Our extensions to ISM include the learning of object parts and template masks to obtain more distinctive votes for the particular object classes. The detections from both sensors are then fused and the objects are tracked using a Kalman Filters with multiple motion models. Experiments conducted in real-world urban scenarios demonstrate the effectiveness of our approach.

## 1 Introduction

One research area that has turned more and more into the focus of interest during the last years is the development of driver assistant systems and (semi-)autonomous cars. In particular, such systems are designed for operation in highly unstructured and dynamic environments. Especially in city centers, where many different kinds of transportation systems are encountered (walking, cycling, driving, etc.), the requirements for an autonomous system are very high. One key prerequisite is a reliable detection and distinction of dynamic objects, as well as an accurate estimation of their motion direction and speed. In this paper, we address this problem by focusing on the detection and tracking of people and cars. Our system is a robotic car

equipped with cameras and a 2D laser range scanner. As we will show, the use of different sensor modalities helps to improve the detection results.

The system we present employs a variety of different methods from machine learning and computer vision, which have been shown to provide robust performances. We extend these methods obtaining substantial improvements and combine them into a complete system of detection, sensor fusion and object tracking. We use supervised-learning techniques for both kinds of sensor modalities, which extract relevant information from large hand-labeled training data sets. In particular, the major contributions of this work are:

- Several extensions to the vision based object detector of [Leibe *et al.*, 2005], that uses a feature based voting scheme denoted as Implicit Shape Models (ISM). Our major improvements to ISM are the subdivision of objects into parts to obtain a more differentiated voting, the use of *template masks* to discard unlikely votes, and the definition of *super-features* that exhibit a higher evidence of an object's occurrence and are more likely to be found.
- The application and combination of boosted Conditional Random Fields (CRF) for classifying laser scans with the ISM based detector using vision. We use a Kalman Filter (KF) with multiple motion models to fuse the sensor information and to track the objects in the scene.

This paper is organized as follows. The next section describes work that is related to ours. Sec. 3 gives a brief overview of our overall object detection and tracking system. In Sec. 4, we introduce the implicit shape model (ISM) and present our extensions. Sec. 5 describes our classification method of 2D laser range scans based on boosted Conditional Random Fields. Then, in Sec. 6 we explain our sensor fusion techniques and our KF-based object tracker. Finally, we present experiments in Sec. 7 and conclude the paper.

## 2 Related Work

This section presents the scientific literature related to people and vehicle detection. It is organized in three parts: the first discusses range based approaches, the second image based methods, and the last one presents the related work in the area of laser-camera multimodal detection.

### 2.1 Range-based methods

Several approaches can be found in the literature to identify a person in 2D laser data. A popular approach is to extract legs by detecting moving blobs that appear as local minima in the range data [Fod *et al.*, 2002; Scheutz *et al.*, 2004; Schulz *et al.*, 2003]. They characterize people by computing geometrical and motion features. When motion features are used, people that do not move can not be detected. The work of [Topp and Christensen, 2005] overcomes

this problem, obtaining good results in an cluttered indoor environment. [Hähnel *et al.*, 2003] consider the problem of classifying beams in range scans that are reflected by dynamic objects. An expectation maximization (EM) estimation is run in order to determine which beam has been reflected by a dynamic object as a person. The work of [Xavier *et al.*, 2005] is also based on the identification of people by geometrical features on the range scan. They segment data into clusters and they apply a set of heuristics in order to distinguish between lines, circles and legs. The first work that formulates the problem of detecting people as a learning problem in a principled manner has been developed by [Arras *et al.*, 2007]. They use geometrical and statistical features computed in clusters extracted from the scan in order to learn a AdaBoost classifier. Excellent results have been presented for indoor environments. [Luber *et al.*, 2008] also make use of learning techniques for detecting and tracking several classes of objects using unsupervised creation of exemplar models. [Arras *et al.*, 2008] use a multi-hypotheses tracker to adaptively address the problem of occlusions and self-occlusions when tracking multiple pedestrians in range data. [Lau *et al.*, 2009] track groups of people by using a distance clustering in a multi-hypotheses tracking system.

Detection of people in 3D range data is recently gaining attention in the robotics community. [Navarro-Serment *et al.*, 2009] use a ground detector, PCA analysis and geometrical descriptors classified by Support Vector Machines for detecting people from 3D data retrieved from several nodding laser rangefinders. [Spinello *et al.*, 2010] detect people in 3D point cloud data by using a smart part-based voting approach based on banks of learned AdaBoost classifiers. This method is appealing for its generality (it does not need any ground detector) and for the accuracy of the results.

A successful work in the field of vehicles detection using range data is the one of [Petrovskaya and Thrun, 2008] that focuses on tracking and detection of multiple vehicles via a model based approach. It encompasses both geometric and dynamic properties of the tracked vehicle in a single Bayes filter. Other approaches based on segmentation and classification are the one of [Zhao and Thorpe, 1998] and [Streller *et al.*, 2002]. The first enforces a rectangular model of a car in range data by using heuristics on extracted lines and uses an Extended Interactive Motion Model for tracking. In the latter, several motion models are used and applied to simple geometrical models of vehicles.

## 2.2 Camera-based methods

In the area of image-based object detection, and people detection in particular, there mainly exist two kinds of approaches (see [Enzweiler and Gavrila, 2009] for a survey). One uses the analysis of a *detection window* or *templates* [Gavrila and Philomin, 1999; Viola *et al.*, 2003], the other performs a *parts-based* detection [Felzenszwalb and Huttenlocher, 2000; Ioffe and Forsyth, 2001]. The detection window approach uses a scalable window that is scrolled through the image. For each step, a classification of the image area under the detection window is obtained. A template-based detection technique is similar to the previously described approach, but in this case a simple distance measure is computed between the edges in

the image under the template silhouette and the silhouette itself. Part-based detection methods aim at independently detecting parts to obtain location hypotheses for entire objects. There exist plenty of computer vision based people detection systems described in the literature. Here, we refer to the most successful ones. [Leibe *et al.*, 2005] presented an image-based people detector using *Implicit Shape Models* (ISM) with excellent detection results in crowded scenes. This method is based on a database or *bag of words*, called codebook, extracted from standard descriptors, that vote for object centers. A mean shift mode estimation is used to define object hypotheses in the continuous space and a minimum description length method to select the winning ones. In earlier works, we showed already extensions of this method with a better feature selection and an improved nearest neighbor search [Spinello *et al.*, 2008b; Spinello *et al.*, 2008a]. Another image-based person detection algorithm that obtained remarkable detection results has been presented by [Dalal and Triggs, 2005]. This method is based on the classification of special image descriptors called Histogram of Oriented Gradients (HOG), computed over blocks of different sizes and scales in a fixed size detection window. The HOG descriptor is based on a collection of normalized image gradients on each cell. The resulting high dimensional vector is then classified with a linear support vector machine (SVM). [Zhu *et al.*, 2006] then refined this detector by using a fast rejector-based SVM cascade to discard the presence of a person in the detection window.

Unlike human bodies, cars have relatively peculiar characteristics in structure such as four wheels, a certain number of pillars, two bumpers, etc. The appearance of these parts changes due to different car models, view points and lighting conditions. The methods already discussed for people detection are also used for detecting cars. [Leibe *et al.*, 2007] detect and track people and cars by using stereo system and an ISM approach where detection hypotheses are selected via an optimization that takes in account overlaps between detections and between object categories. [Zheng and Liang, 2009] compute 'strip features' to describe image locations with arcs, edge-like and ridge-like patterns that are frequently found on vehicles. They learn a complexity-aware RealBoost to produce a fast and accurate classification method. [Papageorgiou and Poggio, 2000] detect cars and people by using an overcomplete set of Haar features classified with a support vector machine method.

### 2.3 Multimodal approaches

Most existing people detection methods based on camera *and* laser range data depend on hard constrains or hand tuned thresholding. [Cui *et al.*, 2005] use multiple laser scanners at foot height and a monocular camera to obtain people tracking by extracting feet and step candidates. [Zivkovic and Kröse, 2007] employ a range-based leg detector and boosted Haar features from camera images to detect people by using a probabilistic ruleset. Both methods cluster laser data points using a Canny edge detector and they extract unrobust image features to detect body parts. These approaches, based on simplistic processing of data, are hardly suited for outdoor scenarios due to presence of clutter in image and range data. Moreover, in such environments a large illumination variability could affect the descriptiveness of fea-

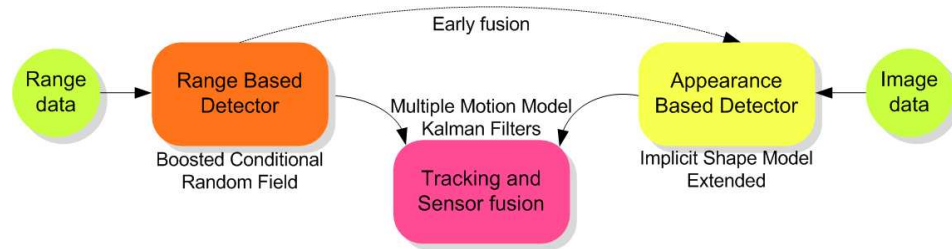


Figure 1: Overview of the method.

tures based on simple intensity-based descriptors (Haar features). The work of [Schulz, 2006] uses probabilistic exemplar models learned from camera and laser data and it applies a Rao-Blackwellized particle filter (RBPF) to track a person's appearance in the data. The RBPF tracks contours in the image based on Chamfer matching as well as point clusters in the laser scan and computes the likelihood of different prototypical shapes in the data. However, in outdoor urban scenarios occlusions are very likely, thus a contour matching approach does not represent an appropriate choice for dealing with partial object visibility. Nevertheless, RBPF is a computationally demanding technique, specially when tracking multiple objects in a scene. [Douillard *et al.*, 2008] employ a Conditional Random Field (CRF) learned on 2D laser data and robust image features to detect multiple classes of objects (i.e. cars, people, vegetation). Promising results are obtained, but occlusions and overlapping object detection hypothesis, critical for yielding good results in any frame, are not handled by the algorithm. The work of [Premebida *et al.*, 2009] does not implement tracking of objects but it evaluates several centralized and decentralized fusion rules with standard vision and laser detectors. [Wender and Dietmayer, 2008] employ a camera and a laser scanner to detect cars in front of a robotic platform. They use simplistic heuristic rules on range data for estimating the viewpoint of the vehicle (front, side etc). Thus, they apply an AdaBoost-based image detector trained with Haar features on different car viewpoints.

### 3 Overview of Our Method

Our system consists of three main components (see scheme in Fig. 1):

- an appearance-based detector that processes data from a camera image
- a range based detector that processes data from a laser rangefinder
- a tracking module that fuses the information from both sensor modalities and provides an estimate of the motion vector for each tracked object.

The laser-based detector is based on a Conditional Random Field (CRF), formulated with a boosted set of geometrical and statistical features of 2D laser range data. The image based



detector extends the multiclass version of the Implicit Shape Model (ISM)[Leibe *et al.*, 2007]. The vision-based detector operates only on regions of interest obtained from projecting range data into the image to constrain the position and scale of the detectable objects (the “early fusion” step). The tracking module applies a Kalman Filter with two different motion models, fusing the information from camera and laser. In the following, we describe the particular components in detail.

**Mathematical notations** Throughout this paper we use the following mathematical notations:

- a *vector* is denoted with a bold letter, e.g. **a**.
- a *matrix* is denoted with a bold capital letter, for example **B**.
- *sets* is denoted with calligraphic capital letters. For example,  $C$ . The cardinality of a set  $C$  is expressed by the notation  $|C|$ .
- numerical constants are denoted with capital letters.

## 4 Appearance Based Detection

Our vision-based people detector is mostly inspired by the work of [Leibe *et al.*, 2005] on scale-invariant Implicit Shape Models (ISM). In summary, an ISM consists in a set  $\mathcal{I}$  of local region descriptors called *codebook*, and a set  $\mathcal{V}$  of displacements and scale factors, usually named *votes*, for each descriptor. The idea is that each descriptor can be found at different positions inside an object and at different scales. Thus, a vote points from the position of the descriptor to the center of the object as it was found in the training data. To obtain an ISM from labeled training data, the descriptors are computed at interest point positions and then clustered, usually using agglomerative clustering with a maximal distance threshold  $\vartheta_d$ . Then, the votes are obtained by computing the scale and the displacement of the objects’ center to the descriptors. A training dataset consists in a collection of images and binary image masks defining the area and the position of the objects in each image. For the detection, new descriptors are computed on a test image and matched against the descriptors in the codebook. The votes that are cast by each matched descriptor are collected in a 3D *voting space*, and a maximum density estimator is used to find the most likely position and scale of an object.

In previous works, we presented already several improvements of the standard ISM approach [Spinello *et al.*, 2008a; Spinello *et al.*, 2008b]. Here, we show some more extensions of ISM to further improve the classification results. These extensions concern both the learning and the detection phase and are described in the following.

## 4.1 ISM Extension: Generating a Superfeature Codebook

In the standard ISM formulation, the process of generating a codebook does not include any feature selection. This has two potential disadvantages: first, a codebook for a given object category may contain many entries, and second, each entry may cast a big quantity of votes. One possibility to reduce the number of codebook entries is to increase the distance threshold  $\vartheta_d$  when creating the codebook. However, in this case each entry in the codebook represents a larger variability of descriptors which leads to more votes per entry. When matching a codebook to new descriptors found in a test image, usually the same distance threshold  $\vartheta_d$  is used as when generating the codebook. Therefore, if  $\vartheta_d$  is large, more matches are found for a given new descriptor. Both effects result in a larger number of votes, which increases the number of false positive detections.

The goal of a *superfeature* codebook is to overcome these disadvantages by collecting more informed descriptors that cast stronger votes. We define superfeatures as features that are stable in image space and in descriptor space. This means that a superfeature is frequently found in the training set, at approximately the same image position with respect to the object center, and its variability in descriptor space is low. This definition ensures that for superfeatures a high evidence of the occurrence of the object is combined with a high probability to encounter an interest point. Let  $\mathcal{O}^+$  be defined as the set of all interest points found inside the segmentation masks in the training data. Each element of  $\mathcal{O}^+$  is a three-dimensional vector, where the dimensions are the relative displacement  $(\Delta x, \Delta y)$  between the location of the interest point and the object center, and the scale  $s$  at which the interest point has been detected. Let furthermore  $\kappa$  be a function that maps from  $\mathcal{O}^+$  to the  $D$ -dimensional descriptor space  $\mathbb{R}^D$ . In the training phase,  $\kappa$  is computed for all interest points in the labeled images. To compute superfeatures, we perform four steps. First, we determine points that lie in very dense areas of  $\mathcal{O}^+$  by applying mean-shift mode estimation [Comaniciu *et al.*, 2001]. This way, we obtain a reduced set  $\mathcal{O}^*$  of interest points, i.e.:

$$\mathcal{O}^* = \text{ms}(\rho_x, \rho_y, \rho_s, \mathcal{O}^+), \quad (1)$$

where  $\text{ms}(\cdot)$  indicates the mean shift estimator with uniform ellipsoidal kernel  $\mathcal{K}$  of semiaxes  $\rho_x, \rho_y$  and  $\rho_s$ . We set  $\rho_x = \rho_y$  in order to give equal importance to interest points found in both directions. In our implementation we use  $\rho_x = \rho_y = 5$  and  $\rho_s = 0.2$ . Thus,  $\mathcal{O}^*$  consists of the  $M$  modes  $\mathbf{o}_1^*, \dots, \mathbf{o}_M^*$  of the interest point distribution in  $\mathcal{O}^+$  as found by the mean-shift estimator.

In the second step, we determine for each mode  $\mathbf{o}_i^*$  the set  $\mathcal{J}_i$  of image descriptors that have been computed at interest points inside the kernel around  $\mathbf{o}_i^*$ , i.e.

$$\mathcal{J}_i = \{\kappa(\mathbf{p}) \mid \mathbf{p} \in \mathcal{O}^+ \cap \mathcal{K}(\mathbf{o}_i^*)\}, \quad (2)$$

where  $\mathcal{K}(\mathbf{o}_i^*)$  denotes the ellipsoidal kernel centered at the mode  $\mathbf{o}_i^*$ . Then, we apply agglomerative clustering with average linkage to the descriptors in  $\mathcal{J}_i$ , i.e.

$$\{C_1, C_2, \dots\} = \text{ac}(\vartheta_d, \mathcal{J}_i), \quad (3)$$

where  $\text{ac}(\cdot)$  represents a function that computes agglomerative clustering with distance threshold  $\vartheta_d$ , and  $C_1, C_2, \dots$  are the resulting clusters in descriptor space.

In the last step, we remove all clusters with cardinality smaller than a threshold  $\vartheta_c$  and store the centroids of those clusters that are bigger than the median of the cardinality of the remaining clusters into the descriptor set  $\mathcal{I}_i^*$ , or formally

$$\mathcal{I}_i^* := \{\text{cn}(C) \mid C \in \mathcal{C} \wedge \|C\| \geq \text{md}(\mathcal{C})\}. \quad (4)$$

Here,  $\mathcal{C}$  denotes the set of all clusters that are bigger than  $\vartheta_c$ ,  $\text{cn}(\cdot)$  computes the centroid of a cluster, and  $\text{md}(\cdot)$  returns the median cluster cardinality. The resulting superfeature codebook  $\mathcal{I}^*$  is defined as

$$\mathcal{I}^* := \bigcup_{i=1}^M \mathcal{I}_i^*. \quad (5)$$

The computation of the set of votes  $\mathcal{V}^*$  for  $\mathcal{I}^*$  follows the same procedure as in standard ISM.

The resulting superfeature codebook  $\mathcal{I}^*$  has less elements than the standard ISM codebook and each entry is associated to less votes. Figure 2 shows a visual explanation of the superfeature codebook generation. It is interesting to see that the superfeatures inherently reflect the skeleton of the object. In case of a pedestrian, superfeatures are mostly taken in the  $\Lambda$ -shaped area between the legs, and nearby the shoulders. Even though this result is strictly related to the kind of interest point detector (e.g. Harris and Hessian interest points are located either on corners or on blobs), it intuitively reflects distinctive local areas for detecting pedestrians. This result is in agreement with other local weighting methods found in the area of image-based people detection (see e.g. the discussion of [Dalal and Triggs, 2005] on the high classification weight that such areas receive).

## 4.2 ISM Extension: Learning Object Parts

The aim of this procedure is to further enrich the information retrieved in the voting process by distinguishing between different object parts from which the vote has been cast. The segmentation into parts is computed offline during the training process for each object category. Here, an object part is defined as a circular sector, where the circle center is aligned with the center of the bounding box that encompasses all training instances of an object class. This definition of an object part is motivated by the fact that the displacement vectors stored in an ISM vote for object centers. Hence, a natural way to distinguish the voters in the training data is with respect to the orientation of their displacement vectors.

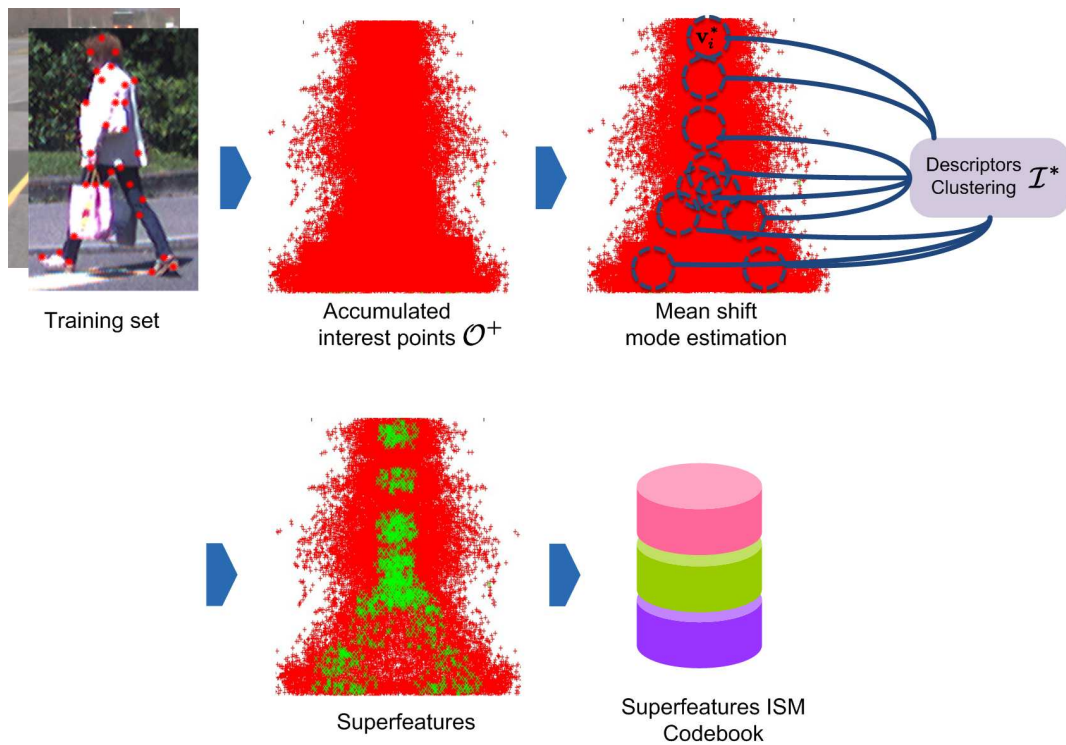


Figure 2: Generation of a superfeature codebook. Superfeatures are stable features in image and descriptor space. First, all interest points from the training data are accumulated in a continuous space. Then, high density areas are found using mean shift mode estimation. In the next step, we consider the descriptors associated to the clustered points and segment them using agglomerative clustering. From the resulting clusters, we select those that are larger than the median and store them together with the votes from the associated interest points in the superfeature codebook. In this example, we used Shape Context descriptors computed at Hessian interest points (in red) for the class 'pedestrian'. The position of the superfeatures are depicted in green.

To distinguish appropriate object parts, we perform three steps. We start again with the accumulated set  $O^+$  of interest points from the training data set. Then, we compute the orientation angle of each displacement vector with respect to the horizontal line through the center of the bounding box that encompasses all object instances (see Fig. 3). All orientation angles are collected in a set  $\mathcal{A}$ . Finally, we apply  $k$ -means clustering [Lloyd, 1982] to the elements of  $\mathcal{A}$ . The problem here is that the number  $K$  of clusters is not given beforehand. We solve this by re-running the clustering algorithm with increasing values of  $K$  and evaluating the resulting clusters with the Bayesian Information Criterion (BIC) [Schwarz, 1978]. The BIC can be used

---

**Algorithm 1:** K-means clustering with estimation of the number of clusters.

---

**Input:** Set of orientation angles  $\mathcal{A}$  from voters**Output:** Optimal set of clusters  $\mathfrak{A}^*$  $K \leftarrow 1$  $b_{old} \leftarrow -\infty$  $b_{new} \leftarrow -\infty$  $\mathfrak{A} \leftarrow \emptyset$ **while**  $b_{new} \geq b_{old}$  **do**     $\mathfrak{A}^* \leftarrow \mathfrak{A}$      $\mathfrak{A} \leftarrow \text{kMeans}(\mathcal{A}, K)$      $b_{old} \leftarrow b_{new}$      $b_{new} \leftarrow -2 \ln\left(\frac{\text{RSS}(\mathfrak{A})}{\|\mathcal{A}\|}\right) + K \ln(\|\mathcal{A}\|)$       Compute BIC using residual sum of squares (RSS)     $K \leftarrow K + 1$ **end****return**  $\mathfrak{A}^*$ 


---

for model selection from a class of parametric models with different numbers of parameters. It represents a balanced score based on the likelihood of the model and its complexity. Our overall clustering method is summarized in Algorithm 1. We note that the Residual Sum of Squares (RSS) of clusters obtained with the  $k$ -means algorithm decreases monotonically with growing  $K$ . The RSS is exactly 0 when  $K = \|\mathcal{A}\|$ , i.e. when each data point defines its own cluster. The BIC is used to trade off a low residual error with a low model cost. Once the BIC does not increase any longer, the maximum is found and the process stops. To perform  $k$ -means clustering on  $\mathcal{A}$ , we need to take care of the fact that the orientation angles are periodic, i.e. 0 needs to be identified with  $2\pi$ . Fortunately, in  $k$ -means clustering only relative distances between points and clusters are required. Thus, we can replace each element in  $\mathcal{A}$  by a corresponding point on the unit circle and use the arc length between two such points as the distance metric for clustering. When clustering is completed,  $\mathcal{A}$  is represented by a collection of angle intervals:  $\mathcal{A} = (a_1, \dots, a_K)$ , where  $a_i = [\alpha_{i-1}, \alpha_i]$  is an angle interval that defines an object part.

An example of the outcome of our clustering algorithm is shown in Fig. 4. Note that although our algorithm does not explicitly search for a semantical subdivision of the object (e.g.: legs, arms, etc. in case of the pedestrian object category), it nevertheless resembles this automatically without human interaction. In Sec. 4.4 we describe how we use this extended shape information for hypothesis selection.

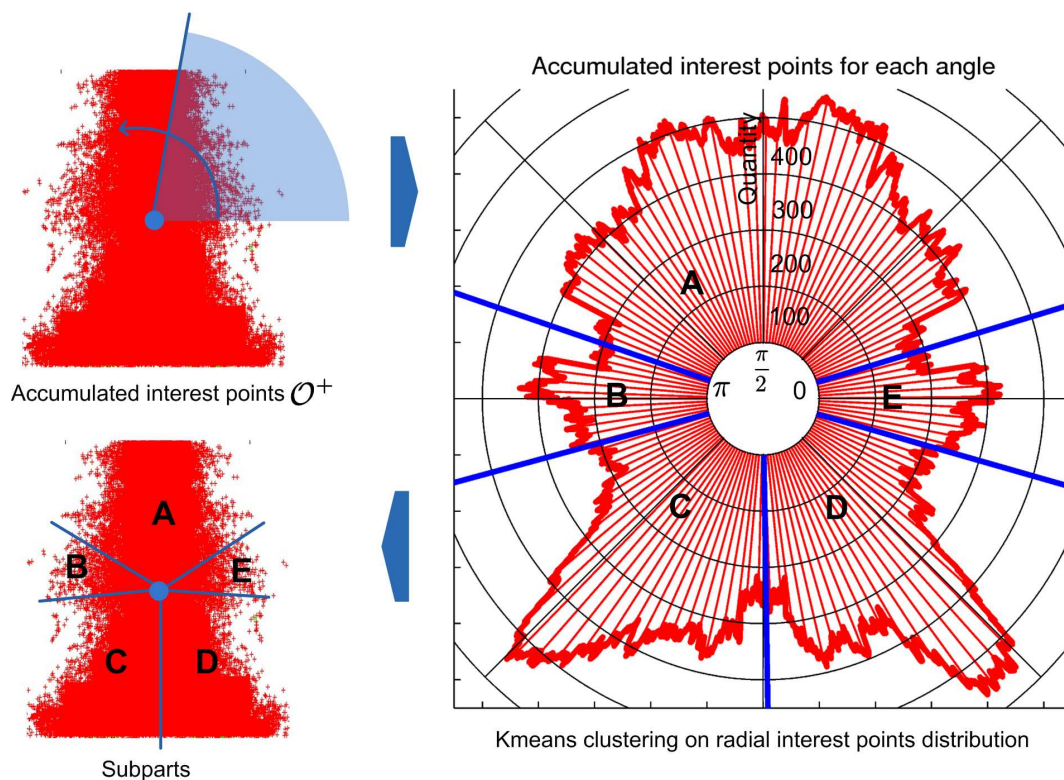


Figure 3: Subparts are computed by accumulating interest points, iteratively running  $k$ -means clustering, and using the BIC to score the cluster result.

### 4.3 ISM Extension: Learning Shape Templates

Based on a similar reasoning as described in the previous section, we propose another extension to the standard ISM approach to distinguish the votes with respect to their quality. The aim of this is to discard *outlier votes*, i.e. those that are cast from interest points located in unlikely areas for a given object class. Outliers are caused by training examples with an unusual shape where some interest points lie outside the most likely shape of the object. For example, there might be training examples of the class “pedestrian”, where a person extremely extends the arms. Then, if there are interest points detected on an arm, the resulting displacement vector stored in  $\mathcal{V}$  will be very rare and thus correspond to an unlikely vote. Later, in the detection phase, this causes problems, because such an unlikely vote is treated in the same way as likely ones, causing many false positive detections.

A first attempt to detect and remove outlier votes has already been made by [Leibe *et al.*, 2005]. There the authors compute a combined optimization between expected segmentation

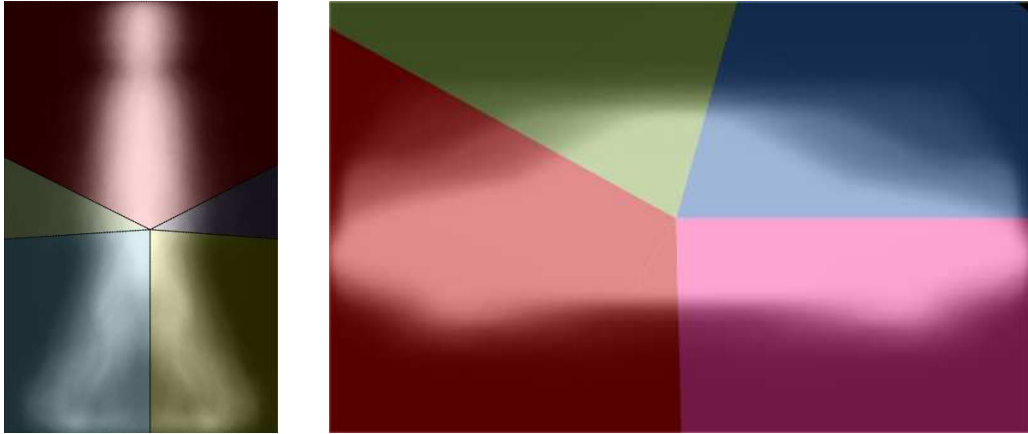


Figure 4: Clustered object parts (colored sectors) and template masks, overlaid as brightness values, for the classes *pedestrian* and *car-sideview*. Both are computed from the training set. Note that even though the object parts are computed unsupervised, they exhibit some semantic interpretation.

and silhouette matching via Chamfer matching [Borgefors, 1988]. This approach is computationally expensive and it is influenced by noise due to the nature of contour matching. In contrast, we propose a probabilistic approach. Instead of relying on the object’s silhouette to determine outliers, we use the entire binary shape masks from the training data. By aligning all shape masks for a given object class so that their center points coincide and by computing the average mask, we obtain a gray value mask  $T_c$  with pixel values between 0 and 1. This procedure is similar to the one used to produce eigenfaces [Sirovich and Kirby, 1987]. These pixel values can be interpreted as prior probabilities for the location of interest points in the given object class. We denote  $T_c$  as the *template mask* of the object class  $c$ . Naturally, all training images used to create the template mask are given in scale 1, but we can obtain template masks at different scales by scaling  $T_c$  using bilinear interpolation.

An example of the template masks which we obtained for the classes “pedestrian” and “car-sideview” is shown in Fig. 4. Here, the template masks are visualized as brightness values together with the part clustering method presented in the previous section. As can be seen, the average shapes of both object classes are clearly visible.

#### 4.4 ISM Extension: Multiclass Hypothesis selection

After learning standard codebooks  $\mathcal{I}_c$ , superfeature codebooks  $\mathcal{I}_c^*$ , segmented object parts  $\mathcal{A}_c$ , and template masks  $T_c$  for each object class  $c = 1, \dots, C$  from the training data as described before, we incorporate these information into the detection step. Here, we have to perform some further adaptation to the standard ISM approach, as we assume a multi-class problem.

Before however, we formulate the detection step mathematically.

After computing interest points and shape descriptors for a given test image, the latter are matched with all codebooks  $\mathcal{I}_c$  and  $\mathcal{I}_c^*$ , and the modes of the voting space are computed using mean-shift, as described above. Let  $\mathbf{h}_c = (\bar{x}_c, \bar{y}_c, s_c)$  be a resulting mode, i.e. a possible center location  $(\bar{x}_c, \bar{y}_c)$  of an object of class  $c$  and its scale  $s_c$ . We will refer to  $\mathbf{h}_c$  as a *hypothesis* of class  $c$ . Furthermore, let  $\mathcal{X}_c$  be the interest point locations  $\mathbf{x}_i$  of all voters that were responsible to create hypothesis  $\mathbf{h}_c$ . As in standard ISM, each vote has an assigned voting strength  $w_i$ . In the following, we will include the voting strength as an additional dimension to the point location vector, i.e.  $\mathbf{x}_i = (x_i, y_i, s_i, w_i)$ . Using this, we define a *voting score* as

$$\text{vs}(\mathbf{h}_c) = \sum_{\mathbf{x}_i \in \mathcal{X}_c} 2^{b_i} w_i T_c(x_i, y_i, \mathbf{h}_c), \quad \text{where } b_i = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ results from } \mathcal{I}_c^* \\ 0 & \text{if } \mathbf{x}_i \text{ results from } \mathcal{I}_c \end{cases} \quad (6)$$

and  $T_c(x_i, y_i, \mathbf{h}_c)$  is the evaluation of the template mask at position  $(x_i, y_i)$  after placing its center at  $(\bar{x}_c, \bar{y}_c)$  and rescaling it with  $s_c$  (see above). This means that the quality of a hypothesis is influenced by four values, namely the number of votes, their strength  $w_i$ , whether they arise from a superfeature match, and the prior quality of the voters obtained from the shape template  $T_c$ . Unlikely votes with respect to the shape template receive a very low weight and their contribution to the hypothesis score is strongly reduced.

Furthermore, for each object class  $c$  we make use of the information of the learned subparts  $\mathcal{A}_c$ . The idea is to obtain an information about the amount of parts that have been detected. Intuitively, a foreground object is expected to have most of the parts well detected, instead, an occluded object appears with less parts. To account for the different object parts from which votes may be cast, we first formulate the voting score  $\text{vs}_k$ , which is *restricted* to an interval  $\mathbf{a}_k = (\alpha_{k-1}, \alpha_k)$  of orientations of vote vectors, where  $k = 1, \dots, K$  is the index of the corresponding object part, i.e.

$$\text{vs}_k(\mathbf{h}_c) = \sum_{\mathbf{x}_i \in \mathcal{X}_c, \alpha_{k-1} \leq \alpha(\mathbf{x}_i) < \alpha_k} w_i T_c(x_i, y_i, \mathbf{h}_c) \quad \text{and} \quad \alpha(\mathbf{x}_i) = \arctan\left(\frac{y_i - \bar{y}_c}{x_i - \bar{x}_c}\right) \quad (7)$$

All part-based scores are then collected in a  $K$ -dimensional vector  $\xi$  defined as

$$\xi(\mathbf{h}_c) = (\text{vs}_1(\mathbf{h}_c), \dots, \text{vs}_K(\mathbf{h}_c)). \quad (8)$$

Intuitively, this is a weighted histogram of votes where each bin corresponds to a learned object part, or equally a sector of vote orientations.

To find the best hypothesis we define a partial order  $<$  on all hypotheses based on a function  $\Delta_r$  as

$$\mathbf{h}_i < \mathbf{h}_j \Leftrightarrow \Delta_r(\xi(\mathbf{h}_i), \xi(\mathbf{h}_j)) < 0 \quad \text{where} \quad \Delta_r(\xi(\mathbf{h}_i), \xi(\mathbf{h}_j)) := \sum_{k=1}^K \text{sign}(\xi_k(\mathbf{h}_i) - \xi_k(\mathbf{h}_j)) \quad (9)$$



**Algorithm 2:** Multiclass detection with ISMe**Input:**

- Interest points  $\mathbf{x}_i$  and corresponding shape descriptors  $\mathbf{d}_i$  from a new test image
- codebooks  $\mathcal{I}_1, \dots, \mathcal{I}_C$ , superfeature codebooks  $\mathcal{I}_1^*, \dots, \mathcal{I}_C^*$  and votes  $\mathcal{V}_1, \dots, \mathcal{V}_C$  for all  $C$  object classes
- minimal hypothesis score  $\sigma_{min}$

**Output:** Set of optimal object hypotheses  $\mathcal{H}^*$  $\mathcal{H}^* \leftarrow \emptyset$  $h_{win} \leftarrow \infty$ **while**  $h_{win} > \sigma_{min}$  **do**  **for**  $c = 1$  **to**  $C$  **do**     $\mathcal{D}_c \leftarrow \text{FindMatches}(\mathcal{I}_c, \{\mathbf{d}_i\})$      $\mathcal{D}_c^* \leftarrow \text{FindMatches}(\mathcal{I}_c^*, \{\mathbf{d}_i\})$      $\mathcal{Y}_c \leftarrow \text{CollectVotes}(\mathcal{D}_c, \mathcal{D}_c^*, \mathcal{V}_c)$      $\mathcal{H}_c \leftarrow \text{ms}(\rho_x, \rho_y, \rho_s, \mathcal{Y}_c)$       Mean-shift operation, returns set of hypotheses    Find  $\mathbf{h}_c^*$  s.t.  $\mathbf{h}_c < \mathbf{h}_c^* \forall \mathbf{h}_c \in \mathcal{H}_c, \mathbf{h}_c \neq \mathbf{h}_c^*$       Best hypothesis for class  $c$ , see Eqn. (9)     $\Gamma_c \leftarrow \text{ComputeHypothesisArea}(\mathbf{h}_c^*)$       see Eqn. (10)  **end**   $\mathbf{h}^* \leftarrow \text{argmax}_{\mathbf{h}_c^*}(\Gamma_1, \dots, \Gamma_C)$    $h_{win} \leftarrow \text{vs}(\mathbf{h}^*)$    $\mathcal{H}^* \leftarrow \mathcal{H}^* \cup \mathbf{h}^*$ **end****return**  $\mathcal{H}^*$ 

where  $\xi_k(\mathbf{h}_i)$  indicates the value contained in the bin  $k$  of the histogram for the hypothesis  $\mathbf{h}_i$ . Intuitively, the function  $\Delta_r$  measures for which of the hypothetical objects the individual object parts are stronger represented in the voting space. Using Eqn. (9), we can determine the hypothesis  $\mathbf{h}_c^*$  with the highest order of all hypotheses for class  $c$ . In case of ambiguity we use the one with the highest global score  $\text{vs}(\cdot)$ .

However, to determine the strongest hypothesis across all object classes, we can not simply compare the scores, as they are based on different codebooks with different numbers of entries. Instead, we use another measure based on the object *area* that is covered by a hypothesis. The idea here is that all point locations in  $\mathcal{X}_c$  of votes that were responsible for  $\mathbf{h}_c$ , can be viewed as small patches inside an object that contribute to the entire shape of the object, just as pieces of a puzzle. To formulate that, we define a square region  $\gamma(\mathbf{x}_i)$  around each  $\mathbf{x}_i$  with side length proportional to the scale  $s_i$ . For the hypothesis  $\mathbf{h}_c$  we can then define the relative area covered

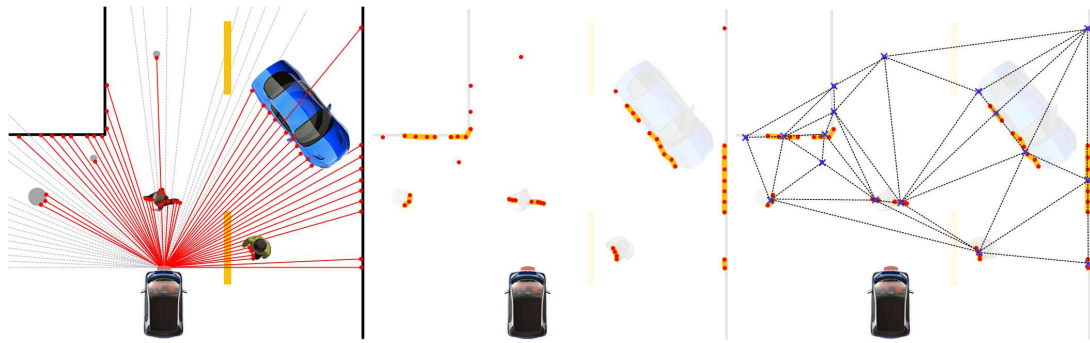


Figure 5: An urban environment with cars, pedestrian and other objects as it is perceived by a 2D laser. **Left:** Laser beams are shown in red, circles represents the measured points. Gray beams indicate out of range data due to material reflections, sun related effects and particular object poses. **Center:** Resulting JDC clustering of the scene. Orange lines depicts consecutive points segmented in the same cluster. **Right:** A Delaunay triangulation is build on the centroids of the segments. This defines a graph among segments.

by all vote patches as

$$\Gamma_c = \frac{\text{area}\left(\bigcup_{\mathbf{x}_i \in \mathcal{X}_c} \gamma(\mathbf{x}_i)\right)}{\|\{(x, y) \mid T(x, y, \mathbf{h}_c) \geq 0.5\}\|}, \quad (10)$$

where the function  $\text{area}(\cdot)$  computes the area of the joint region, and the denominator approximates the area of the object by counting all points in the shape template that are likely to be inside the shape. Care has to be taken in the case of overlapping class hypotheses. Here, we compute the set intersection of the interest points in the overlapping area and assign their corresponding  $\gamma$  values alternately to one and the other hypothesis.

Once an optimal hypothesis  $\mathbf{h}^*$  across all classes is found, we remove all the votes coming from those features that contributed to  $\mathbf{h}^*$ , because we assume that an image feature belongs to just a single object. The scores are then recomputed until a minimum score  $\sigma_{min}$  is reached. Algorithm 2 summarizes the individual steps.

## 5 Structure Based Detection

For the detection of objects in 2D laser range scans, several approaches have been presented in the past (see for example [Arras *et al.*, 2007]). Most of them have the disadvantage that they disregard the conditional dependence between data in a close neighborhood. In particular, they can not model the fact that the label  $l_i$  of a given laser segment  $\mathcal{S}_i$  is more likely to be  $l_j$  if we know that  $l_j$  is the label of  $\mathcal{S}_j$  given that  $\mathcal{S}_j$  and  $\mathcal{S}_i$  are neighbors. One way to model this

conditional dependency is to use Conditional Random Fields (CRFs) [Lafferty *et al.*, 2001], as shown by [Douillard *et al.*, 2008]. CRFs represent the conditional probability  $p(\mathbf{l} | \mathbf{s})$  using an undirected cyclic graph, in which each node is associated with a hidden random variable  $l_i$  and an observation  $\mathbf{s}_i$ . In our case,  $l_i$  is a discrete label that ranges over 3 different classes (pedestrian, car and background) and  $\mathbf{s}_i$  is a feature vector extracted from the 2D segment  $\mathcal{S}_i$  in the laser scan. A preprocessing step on range data has been defined in order to produce segments for the CRF detector. We use a simple clustering technique to group nearby points, called Jump Distance Clustering (JDC). It is fast and simple to implement: if the Euclidean distance between two adjacent data points exceeds a given threshold, a new cluster is generated otherwise the point is added to the current cluster, see Fig. 5-center. Each cluster, or segment, is defined as the set of points  $\mathcal{S}_i$ . Moreover we compute a Delaunay triangulation between the centroids of each segment  $\mathcal{S}_i$  in order to create a graph that connects clusters, see Fig. 5-right.

Assuming a maximal clique size of 2 for the graph, we can compute the conditional probability of the labels  $\mathbf{l}$  given the observations  $\mathbf{s}$  as:

$$p(\mathbf{l} | \mathbf{s}) = \frac{1}{Z(\mathbf{s})} \prod_{i=1}^N \varphi(\mathbf{s}_i, l_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j), \quad (11)$$

where  $Z(\mathbf{s}) = \sum_{\mathbf{l}} \prod_{i=1}^N \varphi(\mathbf{s}_i, l_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j)$  is usually called the *partition function* and  $\mathcal{E}$  is the set of edges in the graph.  $\varphi$  and  $\psi$  represent node and edge potentials.

To determine the node and edge potentials  $\varphi$  and  $\psi$  we use the log-linear model:

$$\varphi(\mathbf{s}_i, l_i) = e^{\mathbf{u}_n \cdot \mathbf{f}_n(\mathbf{s}_i, l_i)} \quad (12)$$

$$\psi(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j) = e^{\mathbf{u}_e \cdot \mathbf{f}_e(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j)} \quad (13)$$

where  $\mathbf{f}_n$  and  $\mathbf{f}_e$  are feature functions for the nodes and the edges in the graph, and  $\mathbf{u}_n$  and  $\mathbf{u}_e$  are feature weights that are determined in the training phase. The computation of the partition function  $Z$  is intractable due to the exponential number of possible labelings  $\mathbf{l}$ . Instead, we compute the *pseudo-likelihood*, which approximates  $p(\mathbf{l} | \mathbf{s})$  and is defined by the product of all likelihoods computed on the *markov blanket* (direct neighbors) of node  $i$ .

$$pl(\mathbf{l} | \mathbf{s}) = \prod_{i=1}^N \frac{\varphi(\mathbf{s}_i, l_i) \prod_{\mathbf{s}_j \in \mathcal{N}(\mathbf{s}_i)} \psi(\mathbf{s}_j, \mathbf{s}_i, l_j, l_i)}{\sum_{\mathbf{l}'} (\varphi(\mathbf{s}_i, l_i') \prod_{\mathbf{s}_j \in \mathcal{N}(\mathbf{s}_i)} \psi(\mathbf{s}_j, \mathbf{s}_i, l_j', l_i'))} \quad (14)$$

Here,  $\mathcal{N}(\mathbf{s}_i)$  denotes the set of direct neighbors of node  $i$ . In the training phase, we compute the weights  $\mathbf{u} = (\mathbf{u}_n, \mathbf{u}_e)$  that minimize the negative log pseudo-likelihood together with a Gaussian shrinkage prior as in [Ramos *et al.*, 2007]:

$$L(\mathbf{u}) = -\log pl(\mathbf{l} | \mathbf{s}) + \frac{(\mathbf{u} - \hat{\mathbf{u}})^T (\mathbf{u} - \hat{\mathbf{u}})}{2\sigma^2} \quad (15)$$

For the minimization of  $L$ , we use the L-BFGS gradient descent method [Liu and Nocedal, 1989]. Once the weights are obtained, they are used in the inference phase to find the labels  $\mathbf{l}$  that maximize equation (11). Here, we do not need to compute the partition function  $Z$ , as it is not dependent on  $\mathbf{l}$ . We use max-product loopy belief propagation (BP) to find the distributions of each label  $l_i$ . The final labels are then obtained as those that are most likely for each node.

In our case the Delaunay triangulation among segments defines the structure of the network. We use a set of statistical and geometrical features for the nodes of the CRF, e.g. width, circularity, standard deviation, kurtosis, etc. (for a full list see [Spinello and Siegwart, 2008]). However, we do not use these features directly in the CRF, because, as stated in [Ramos *et al.*, 2007] and also from our own observation, the CRF is not able to handle non-linear relations between the observations and the labels. Instead, we apply AdaBoost [Freund and Schapire, 1997] to the node features and use the outcome as features for the CRF. For our particular classification problem with multiple classes, we train one binary AdaBoost classifier for each class against the others. As a result, we obtain for each class  $c$  a set of  $M$  weak classifiers  $h_i^c$  (in this case decision stumps) and corresponding weight coefficients  $\alpha_i^c$  so that the sum

$$g_c(\mathbf{s}_i) := \sum_{i=1}^M \alpha_i^c h_i^c(\mathbf{s}_i) \quad (16)$$

is positive for observations assigned with the class label  $c$  and negative otherwise. Note that the absolute value of  $g_c$  can be interpreted as a classification quality. To obtain a classification *likelihood*, we apply the logistic function  $a(x) = (1 + e^{-x})^{-1}$  to  $g_c$ . We do this for two reasons: first the resulting values are between 0 and 1 and can be interpreted as likelihoods of corresponding to class  $c$ . Second, by applying the same technique also for the edge features, the resulting potentials are better comparable. Thus, the node feature function  $\mathbf{f}_n$  of the segment features  $\mathbf{s}_i$  and the label  $l_i$  is computed as:

$$\mathbf{f}_n(\mathbf{s}_i, l_i) = a(g_{l_i}(\mathbf{s}_i)) \quad (17)$$

For the edge features  $\mathbf{f}_e$  we compute two values, namely the Euclidean distance between the centroids  $\mathbf{c}_i$  and  $\mathbf{c}_j$  of the segments  $\mathcal{S}_i$  and  $\mathcal{S}_j$ , along with a value  $g_{ij}$  defined as:

$$g_{ij}(\mathbf{s}_i, \mathbf{s}_j) = \text{sign}(g_i(\mathbf{s}_i)g_j(\mathbf{s}_j)) \cdot (|g_i(\mathbf{s}_i)| + |g_j(\mathbf{s}_j)|). \quad (18)$$

Thus, the value of  $g_{ij}$  has a positive sign if AdaBoost classifies  $\mathbf{s}_i$  and  $\mathbf{s}_j$  into the same class and otherwise it is negative. The absolute value of  $g_{ij}$  is the sum of the classification qualities of AdaBoost. If  $g_i(\mathbf{s}_i)$  and  $g_j(\mathbf{s}_j)$  are far from 0 then  $g_{ij}$  has a high value, and viceversa. To summarize, we define the edge features as:

$$\mathbf{f}_e(\mathbf{s}_i, \mathbf{s}_j, l_i, l_j) = \begin{cases} (a(\|\mathbf{c}_i - \mathbf{c}_j\|) \quad a(g_{i,j}(\mathbf{s}_i, \mathbf{s}_j)))^T & \text{if } l_i = l_j \\ (0 \quad 0)^T & \text{otherwise.} \end{cases} \quad (19)$$

The intuition behind equation (19) is that edges that connect segments with equal labels have a non-zero feature value and thus yield a higher potential. The latter is sometimes referred to as the generalized Potts model (see [Potts, 1952]).

## 6 Object Tracking and Sensor Fusion

In this section we explain how to combine the two sensor modalities together. Range and image data is used for “early fusion” and then combined in the tracking system. The early fusion step consists in a technique for constraining the vision-based detector in salient image regions. The tracker combines detection results from camera and laser and eventually solves data association.

An important factor in our multisensor system is the extrinsic calibration between camera and laser. Internal camera parameters are estimated by computing intrinsic camera calibration [Zhang, 1999]. We employ the method explained in [Pless and Zhang, 2004] to calibrate the 2D laser rangefinder with the camera. The procedure consists in simultaneously collecting image and range data of a planar checkerboard placed in front of a robot at different positions and orientations. For each pose of the planar pattern, the method constrains the extrinsic parameters by registering the laser scanline on the planar pattern with the estimated plane computed from the image. The solution uses nonlinear optimization that minimizes the re-projection error.

### 6.1 Early fusion: using laser segments to bound the voting space

The early fusion method is concerned with the definition of constraints in the ISMe voting space of the image-based detector, in order to generate more precise object hypotheses. The idea is to project segments extracted from the laser data as 3D boxes in the voting space. If we consider a single laser segment, it could be projected as a box with height set to a fixed value, width defined by the extremal points of the segment  $\mathcal{S}_i$  and depth defined by the scale tolerance  $\vartheta_{\mathcal{S}_i}$ . These 3D boxes define boundaries in the voting space for hypothesis selection for the image detector. Before the image hypothesis selection is run, the early fusion takes place and it removes hypotheses that are not compatible with the boundaries. The generous dimensions of the boxes allow the survival of imprecise detections in position and scale.

In order to consider range-image early fusion process, we need to set  $\vartheta_{\mathcal{S}_i}$  for each object class. Precisely, we need to compute  $\vartheta_{\mathcal{S}_i}$  as a function of the laser segment distance. We assume, for practical reasons, that the relationship between two variables is linear, even though this is not true due to lens distortions. The idea is to produce a linear least squares regression that relates the objects pixel heights  $\omega^s$  and objects distances  $\omega^d$ :

$$\omega_i^d = \beta_1 \omega_i^s + \beta_2 \quad (20)$$

where  $(\beta_1, \beta_2)$  are the parameters of the line computed with the regression from a collection of objects measured heights and distances. Thus, we are able to query an *hallucinated* distance for each object category, by providing a scale input (and viceversa).

The scale  $\omega^s$  estimated for each segment distance is then converted in the depth of the 3D region of interest in the ISMe voting space, in order to easily prune false image detection hypotheses:

$$\vartheta_{\mathcal{S}_i} = (\omega_i^s - \vartheta_{\mathcal{S}}^*, \omega_i^s + \vartheta_{\mathcal{S}}^*) \quad (21)$$

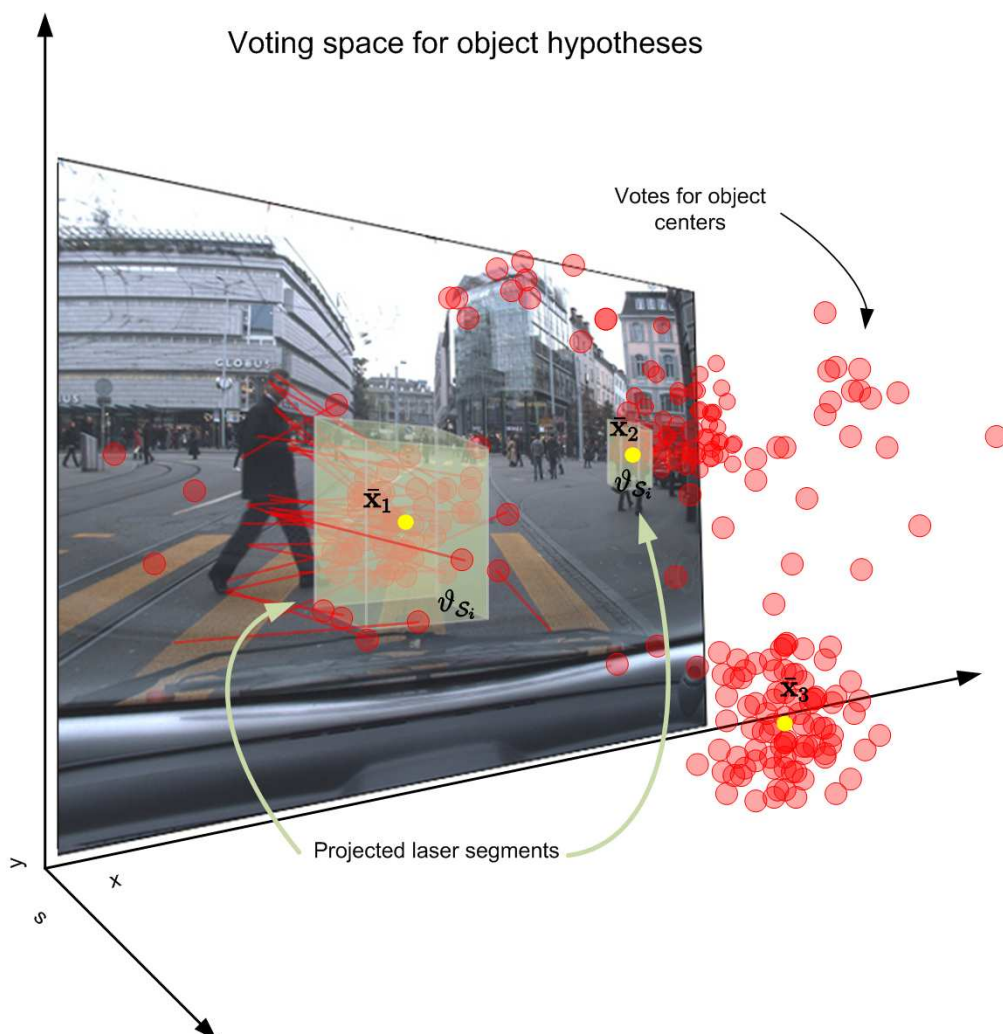


Figure 6: Early laser-camera fusion. Laser segments are projected into the visual-based object detection voting space as 3D boxes. Image detection hypotheses located into one of these regions are considered valid, the others are discarded. Votes are shown as red circles. Object hypotheses  $\bar{x}_i$  are shown in yellow. For clarity, features voting for the object centers (defined by position and scale) are shown only on the left pedestrian.

where  $v_S^*$  is a constant fixed beforehand. An image detection hypothesis is considered valid if it is found inside one of the 3D boxes that define the constraints of the voting space. A visual explanation can be seen in Fig. 6.

The last step of the early fusion technique is to solve the data association problem of the assignment between segments and corresponding image hypotheses. We assume that each segment belongs to a single object. For each segment we compute the distance and compute the hallucinated scale (see Equation 20). We solve the assignment problem in a greedy manner: given a segment  $S_i$ , we assign from all valid image detection hypotheses found in the projected segment volume, that one to the segment  $S_i$ , which minimizes the absolute difference between the scale of the hypothesis and the hallucinated scale. The remaining processing of hypothesis selection for detection in camera images follows the technique explained in Section (4.4).

## 6.2 Combined detection using Kalman Filtering

The aim of multimodal object detection is to provide useful information to a navigation or a driver assistance module. For this reason, a natural output choice for our detector is to label laser segments with their class probability. The proposed fusion method combines the detectors' information and provides output that consists in laser segment positions and object category labels.

We use tracking as a mean of integrating class probabilities over time and as an additional algorithm output, to provide prediction information. We aim to design a reliable tracking method that does not rely on single data association hypotheses and that scales gracefully with the number of objects. Several methods have been developed in the tracking literature for handling complex data association at a high computational cost (see Multi-hypothesis tracking [Reid, 1979; Cox and Hingorani, 2002] and JPDA filters [Bar-Shalom and Li, 1995]). Our tracking algorithm is designed to be computationally inexpensive and copes well with the motion model of several kinds of object categories.

In contrast to cars, which have a comparably simple motion model given by the Ackermann model [Ackermann, 1818], pedestrians are much harder to describe with a single motion model: they can stop, suddenly turn on spot, invert their trajectory etc. Therefore, we use a pedestrian tracker in which each track is described by multiple Kalman Filters, each providing a different motion model. The advantage of this method is that the number of estimating filters scales linearly with the number of objects to track. Moreover, multiple hypotheses regarding object motions are produced for each time step. For this work we employed two kind of motion models, described by linear velocity and Brownian motion. The motivation for selecting Brownian motion is the ability to model sudden direction and speed changes, a condition that occurs especially in case of people tracking. Nevertheless, a constant velocity model, in short intervals, well approximates a variety of smooth curved trajectories. The proposed tracking technique is a way of combining tracking filters and it is very generic: An Extended Kalman Filter (EKF), as well as other motion models, linear and non-linear, could be also used.

Tracks are managed by a *tracking manager* that solves data association, and creates or deletes tracks. We assume that each track is associated at most to one single segment.  $N$  is the number of laser segments present in a laser scan,  $R$  the number of tracks and  $M$  is the number of Kalman Filters present for each track, each with a different motion model. Data association,

i.e. the problem of assigning laser segments to tracks, is solved in two steps. The first step is to compute which motion model to use for each track. In each track, the distance between the Kalman Filters (KF) prediction and the  $N$  laser segments centroids are computed. This process generates for each track  $M$  Mahalanobis distances for each observation [Mahalanobis, 1936]. In each track, the closest distance for each observation is taken and the KF generating that prediction is tagged. At the end of the first step of the association of laser segments, every track obtains a set of  $N$  distances from  $N$  observations.

The second step of data association is used to select which observation is assigned to which track. We want to assign  $N$  hypotheses to  $R$  tracks (where  $N \neq R$ ). A rectangular matrix of size  $R \times N$  is generated in which rows represent track indices and columns observation indices. The previously computed distances are inserted as values of the assignment matrix. The solution of the combinatorial minimal weight assignment has been found with the extension of the Munkres method for rectangular assignment matrices proposed by [Bourgeois and Lassalle, 1971]. If there are more segments than tracks then  $R - N$  new tracks are created. Instead, if more tracks than segments are present in a certain moment, the tracks that are not updated with a new observation are maintained until their variance in  $(x, y)$  reaches a fixed maximum threshold  $\delta_{x,y}$ .

We now give a mathematical formulation for the tracks and for the fusion of the detection outputs. We track cluster centroids in 2D range data using two KF, each with a different motion model:

$$\mathbf{x}_{m1} = \left( (\hat{x}^S, \hat{y}^S), (\dot{x}^S, \dot{y}^S), (p_1, \dots, p_C) \right) \quad (22)$$

$$\mathbf{x}_{m2} = \left( (\hat{x}^S, \hat{y}^S), (p_1, \dots, p_C) \right) \quad (23)$$

$(\hat{x}^S, \hat{y}^S)$  are the coordinates of the cluster centroid,  $(\dot{x}^S, \dot{y}^S)$  is its velocity and  $p_1, \dots, p_n$  are the probabilities of all  $C$  classes. The observation vector  $\mathbf{z}(k)_i$ , at time  $k$ , consists of the position of the cluster centroid and the categories probability estimates for each detection modality:

$$\mathbf{z}_i = \left( \hat{x}_i^S, \hat{y}_i^S, (c_1, \dots, c_n)^1, \dots, (p_1, \dots, p_C)^\zeta \right). \quad (24)$$

Here,  $(\hat{x}^S, \hat{y}^S)$  is an observation of a cluster centroid and  $\zeta$  denotes the number of sensors. Each block  $(p_1, \dots, p_C)$  is the estimate given by the range or image based classifier.

Kalman Filter is formulated by prediction and update step. Prediction at time  $k$  is computed by:

$$\mathbf{x}(k)_{mi}^- = \mathbf{A}_{mi} \mathbf{x}(k-1)_{mi}^- \quad (25)$$

We write the state matrix  $\mathbf{A}_{mi}$  in the case of two motion models and two classes:

$$\mathbf{A}_{m1} = \begin{pmatrix} 1 & 0 & \Delta k & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta k & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{A}_{m2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (26)$$



If the matrix  $\mathbf{V}^1$  indicates the state covariance matrix and  $\mathbf{V}^2$  the sensors covariance matrix, we compute:

$$\mathbf{P}(k)_{mi}^- = \mathbf{A}_{mi}\mathbf{P}(k-1)\mathbf{A}_{mi}^T + \mathbf{V}^1 \quad (27)$$

The tracker manager selects which KF of each track is closer to the observation  $\mathbf{z}_i$ . Then it solves data association between winning KF of each track and observations by using Munkres assignment optimization. Observations are assigned to track and filters are updated. The observation is used to update all the filters of the track. The update step is calculated by computing the Kalman gain  $\mathbf{G}$ , then updating  $\mathbf{x}(k)_{mi}$  and the covariance matrix  $\mathbf{P}$ :

$$\mathbf{K}_{mi} = \mathbf{P}(k)_{mi}^- \mathbf{G}^T (\mathbf{G} \mathbf{P}(k)_{mi}^- \mathbf{G}^T + \mathbf{V}^2)^{-1} \quad (28)$$

$$\mathbf{P}(k)_{mi} = (\mathbf{I} - \mathbf{K}_{mi} \mathbf{G}) \mathbf{P}(k)_{mi}^- \quad (29)$$

$$\mathbf{x}(k)_{mi} = \mathbf{x}(k)_{mi}^- + \mathbf{K}_{mi} (\mathbf{z}(k)_a - \mathbf{G} \mathbf{x}(k)_{mi}^-) \quad (30)$$

where  $\mathbf{z}(k)_a$  represents the assigned observation vector to the track. The matrix  $\mathbf{G}$  models the mapping from states to the predicted observation and is defined as  $\mathbf{G} = (\mathbf{G}_x^T \mathbf{G}_{s1}^T \dots \mathbf{G}_{sc}^T)^T$ , where  $\mathbf{G}_x$  maps to pose observations and the  $\mathbf{G}_{s1}$  map to class probabilities per sensor. For example, for one laser, one camera and constant velocity we have:

$$\mathbf{G}_{s1} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \mathbf{G}_{s1} = \mathbf{G}_{s2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (31)$$

## 7 Experimental Results

In this section, we present experimental results and quantitative comparisons with other techniques in order to validate our method.

### 7.1 Experimental Platform

To acquire the data, we employed our urban mobile platform *Smarter*. The robot is based on a standard Smart car that has been equipped with distance laser sensors, cameras, a differential GPS unit, an Inertial Measurement Unit (IMU), an optical gyroscope and several processing computers. In this case we acquired data by using a camera equipped with a telelens and a 2D laser range finder mounted in front. The camera has been mounted on a metal rig on the rooftop of the vehicle and the logging system has been optimized to reduce frame drops.

### 7.2 Real World Dataset: Urban Scenario

We evaluated our technique on a challenging urban scenario dataset. We have set the laser angular resolution to 0.25 degrees in order to obtain a high resolution laser dataset. Data is collected around Zürich, Switzerland in a loop of circa 1km length for retrieving cars and

Num Frames	1675
Laser range data resolution	0.25deg
Image resolution	640 × 480px
Laser positioning	horizontal, 48cm from ground
Camera lens	Telelens, 45deg f.o.v.
Num of car samples	510
Num of people samples	376

Table 1: Urban Scenario testing dataset, collected in downtown Zürich, Switzerland

pedestrians in a real busy urban environment. We synchronized camera and laser data for a total of 1675 frames. The imagery is manually labeled with rectangle boxes indicating pedestrians and cars. Annotations in images are marked if at least half of an object is shown or the object width in the image is greater than 80 pixels. Laser range data has been manually labeled by using associated image frames as reference for the ground truth. Labeling is obtained by manually selecting and assigning a class label to the segments in the range data. A suite of MATLAB scripts has been used to simplify this process.

### 7.3 ISMe image detector training

Several ISM codebooks need to be trained due to the complexity of the multiclass (cars, pedestrians) classification task. Experience shows [Leibe *et al.*, 2007] that lateral views of pedestrians generalize well to front/back views. Therefore, we used a set composed of 400 images of persons with a height of 200 pixels at different positions, dressed with different clothing and accessories such as backpacks and hand bags in a typical urban environment. The category 'car' has been learned from 7 different viewpoints as in [Leibe *et al.*, 2007] (see also Figure 7, left). 200 training images are used for each view. Car codebooks are learned using Shape Context (SC) descriptors [Belongie *et al.*, 2002] at Hessian-Laplace interest points [Mikołajczyk and Schmid, 2005]. The pedestrian codebook uses lateral views and SC descriptors at Hessian-Laplace and Harris-Laplace interest points for more robustness. We selected SC due to their low dimensionality (36D): this shortens the time for feature extraction, for the agglomerative clustering of the codebook generation and for feature matching with codebooks. In the work of [Leibe *et al.*, 2006], the authors compare several descriptors for object detection and they show that SC descriptors are very good features for object detection.

### 7.4 Boosted CRF range detector training

Our laser training data consists of 600 annotated scans containing pedestrians, cars and background randomly sampled from a typical urban scenario. 5158 car data points, 2379 people

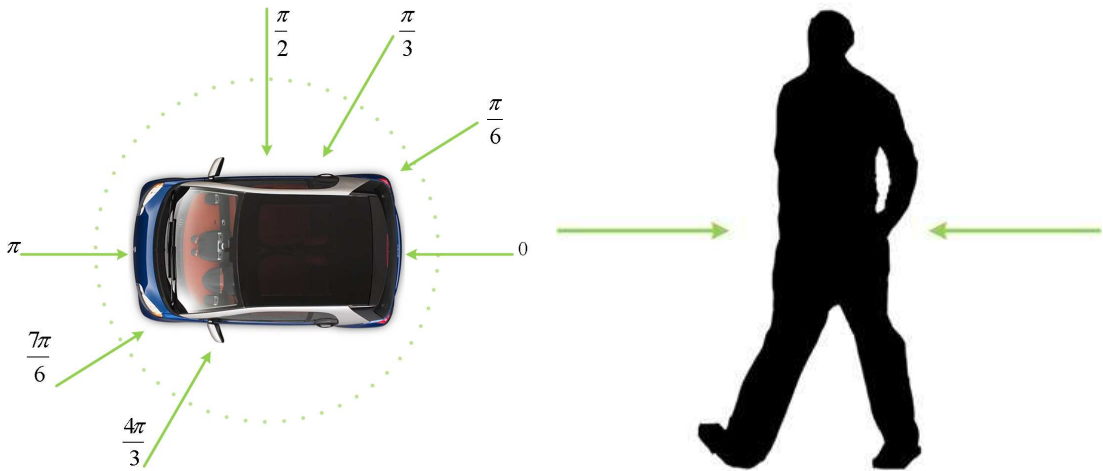


Figure 7: **Left:** For car classification, we use codebooks from 7 different views. For training, mirrored images are included for each view to obtain a wider coverage. **Right:** For pedestrians we use a codebooks of side views with mirroring. Lateral views have sufficient information to generalize frontal/back views.

data points and 25251 background labeled points have been used for training. There is no distinction of car views in the laser data as the variation in shape is low. The range data is limited to a maximum range of 15m. As a first step, the AdaBoost classifier of range data features is trained on this set. Then we use the output of the trained classifier to produce node feature values for the CRF. Then, the CRF is trained in order to set the node and edge features weights.

## 7.5 Quantitative and Qualitative evaluation

In this section we present results in form of precision-recall curves. They summarize the complete performance of a classifier:

$$\text{Precision} = \frac{\text{TruePos}}{\text{TruePos} + \text{FalsePos}} \quad \text{Recall} = \frac{\text{TruePos}}{\text{TruePos} + \text{FalseNeg}} \quad (32)$$

Precision-recall curve has the advantage of computing a classifier performance measure without knowing the number of true negatives. Specially in case of image and range data classification, setting this number can be ambiguous because the quantity of possible true negatives in such data is not easy to define.

We run a comparison of the proposed multiclass image detection algorithms with our previous work [Spinello *et al.*, 2008a], as shown in Figure 13. Our vision based multiclass detection,

named ISMe2.0 in the plots, is compared to the standard ISM, our previous single class detector ISMe1.0 and with an AdaBoost detector trained on Haar features (ABH). We can see that our method yields the best results. It is important to see that the multiclass method obtains higher recall values than the previous ISMe1.0, mostly due to the refinements introduced in the hypothesis selection step, namely the object subparts and the shape templates.

We then run the system for the challenging Urban Scenario dataset. Pedestrian detection with camera is shown in Fig. 8-left.

In the evaluation of results we compare the performance of several detectors by using equal error rate (EER) error metric on a precision-recall graph. EER is a measure to compare the accuracy of classifier. This measure is often used, specially in biometrics [K.P. Li, 1988] and in computer vision [Leibe *et al.*, 2005]. In general, the classifier with the lowest EER is most accurate. EER is the point in which false positive rate and false negative rate have the same value. The lower the EER, the more accurate the system is considered to be. The higher the diagonal crossing point in the precision-recall curve, the lower EER, the less the errors computed by the classifier.

We compared our image detector with respect to an Haar-AdaBoost based classifier and, in case of the pedestrian detector, with the Histogram of Oriented Gradients technique developed by [Dalal and Triggs, 2005]. In case of HOG and ABH we used the early fusion technique explained in Section 6.1 in order to reduce the image search space. Our multiclass detector, shortly named ISMe, clearly outperforms the other methods. Precision at equal error rate (EER) is: 60.01% for ISMe, 52.21% for HOG, 11.17% for ABH. In general, if one is willing to accept a high rates of false positives, the ISMe detector could achieve a  $> 70\%$  Recall. At that values the difference with respect to the other methods is even more evident. We then evaluated the laser based detector for pedestrian detection in Figure 8-right. There we show a comparison between the Boosted CRF and a standard AdaBoost classification of JDC segments (AJDC) in order to visualize the introduced performance enhancements. AJDC classifies JDC segments regardless of the neighborhood state. It is interesting to notice that the consideration of the segments' neighborhood in the CRF plays an important role in the ability to increase the detection rate and reduces the number of false positives, the AJDC curve is always below the CRF one and it decreases earlier than the CRF curve. In this case precision at EER is 64.23% for the CRF and 57.09% for AJDC.

We then evaluated the performance of our system in case of car detection (see Figure 10). The ISMe car image detector outperforms the ABH detector. The latter has been trained on trunks, sides and frontal views of cars. It is important to remark that the results shown in Figure 10-left are averaged between the 7 car views of ISMe. the Equal Error Rate is crossed at 72.54% for ISMe and 18.93% for ABH. The performance of the laser based classifier is compared with AJDC in Fig. 10-right and also in this case CRF has better results with respect to AJDC. Precision at EER is 74.89% for CRF and 70.57% for AJDC. It is interesting to notice that cars are in general easier to detect with respect to pedestrians. Intuitively, cars are rigid objects with much less geometrical and visual variability than visually complex pedestrians.

Tracking and fusion for the pedestrian category is evaluated in Fig. 9. We show the

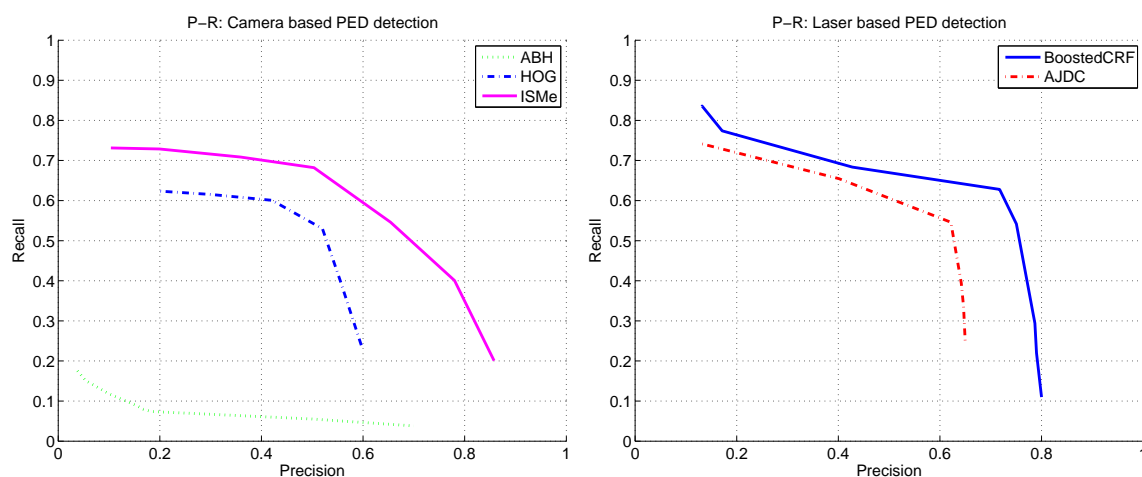


Figure 8: Quantitative evaluation for pedestrian detection. Our approach outperforms the other methods for both sensor modalities. The image based detection is compared with Histogram of Oriented Gradients detector (HOG) and an AdaBoost classifier using Haar features (ABH). We show a comparison between Boosted CRF and AdaBoost classification of JDC segments (AJDC) in order to visualize the introduced performance enhancements.

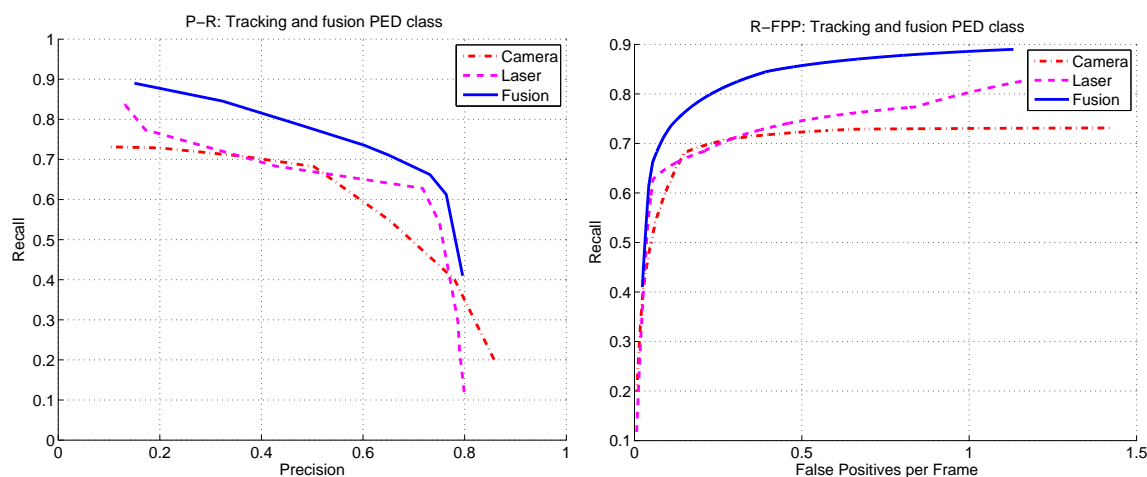


Figure 9: Quantitative evaluation of tracking and fusion for pedestrian detection. Precision-Recall graph (left) and 'Recall-false positives per frame' show that the fusion method enhances the results of single classifiers.

precision-recall graph and a 'Recall-false positives per frame' plot in order to show the performance increase. It is interesting to see in the plot of Fig. 9-left that the camera and laser detectors are very complementary sources of information: their combined contribution allows to have a fused detection that is higher than each single one; this phenomenon is even more evident when precision is low. The tracked and fused precision at EER is 69.8%. In Figure 9-right we show that we improved also that: by fixing a certain false positive rate per frame, we obtain a higher Recall value. Tracking and fusion for cars is shown in Fig. 11. Similar conclusions to the sensor fusion on pedestrians could be given. Tracking allows a better detection rate than each single classifier and a reduced number of false positives per frame; precision at EER is 78.4%. This value of precision is significantly higher than the pedestrian category due to the higher performances of vision and laser detectors.

From these experiments we can draw some interesting conclusions. Image and range data are two very different sensor modalities, with very different characteristics. With these experiments we proved that image and range based detectors can be combined for obtaining a fused detector that is more robust than its components. Range data has the advantage of a precise and instantaneous target localization and it helps to distinguish objects that have a low image information content, for instance people in shadow areas, or partial views of cars. Image, instead, plays an important role when range data is ambiguous, for instance when a person is observed from the side or in presence of clutter. Both of these examples show how single sensor modalities could fail and how the multimodal fusion overcomes these flaws. Moreover it is interesting to notice, that in case of limited visibility, poor/no light conditions or camera failure, this approach still produces a usable output, see for instance Figure 12-middle or Figure 12-bottom.

Certainly, this approach presents shortcomings. The technique is limited to the range of 15m due to sparsity of the retrieved laser data points. At such distance cars and specially people are described by too few points to obtain good range data classification results. Severe street slopes could also contribute to *short-sightedness* of the fused detector. This aspect has been addressed in a previous work [Spinello *et al.*, 2008a] by using 3D ground plane estimate with a 3D laser.

Some qualitative results are shown in Figure 12 where a passing car and a crossing pedestrian are correctly detected and tracked. It is important to notice that even though images and laser data show very low contrast, partial occlusions and clutter, the system manages to detect and track the objects in the scene. For a video extracted from the experiments see Extension 1 (Appendix A).

## 8 Conclusions

We have presented a method to reliably detect and track multiple object classes in outdoor scenarios using vision and 2D laser range data. We showed that the overall performance of the system is improved using a multiple-sensor system. We have introduced several extensions to the ISM based image detection to cope with multiple classes. We showed that laser detec-

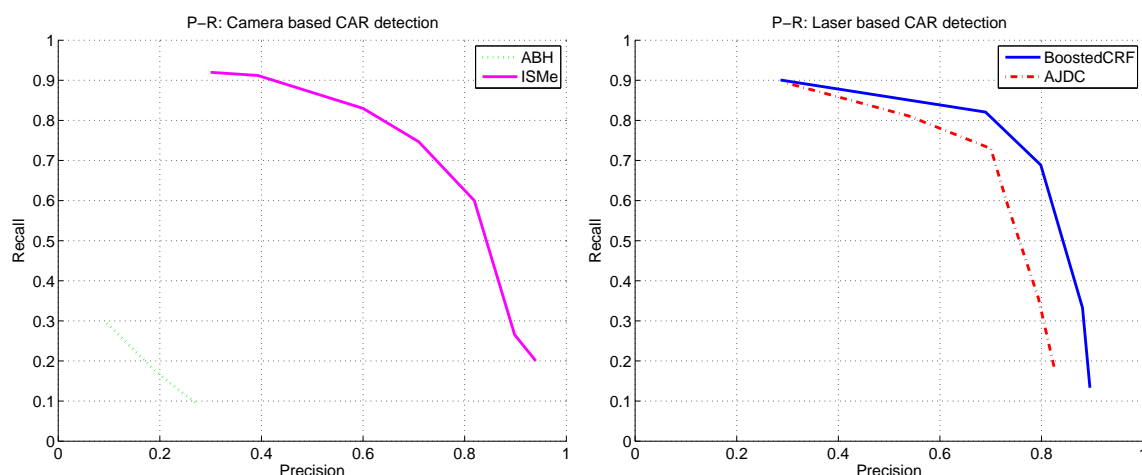


Figure 10: Quantitative evaluation for car detection. Our approach outperforms the other methods for both sensor modalities. The image based detection is compared with an AdaBoost classifier using Haar features (ABH). We show a comparison between Boosted CRF and AdaBoost classification of JDC segments (AJDC) in order to visualize the introduced performance enhancements.

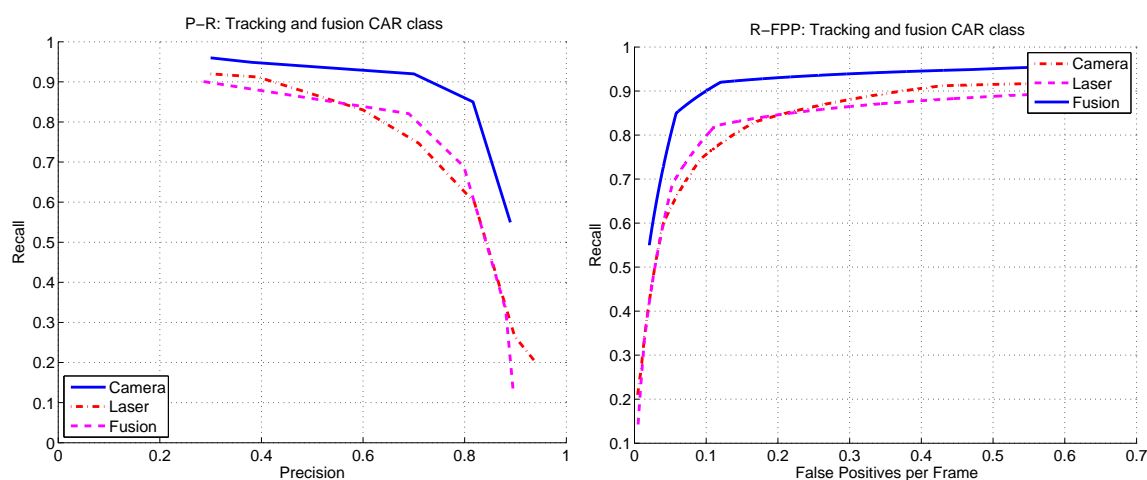


Figure 11: Quantitative evaluation of fusion for car detection. Precision-Recall graph (left) and 'Recall-false positives per frame' show that the fusion method enhances the results of single classifiers.

tion based on Boosted CRFs performs better than a simpler AdaBoost classifier and presented

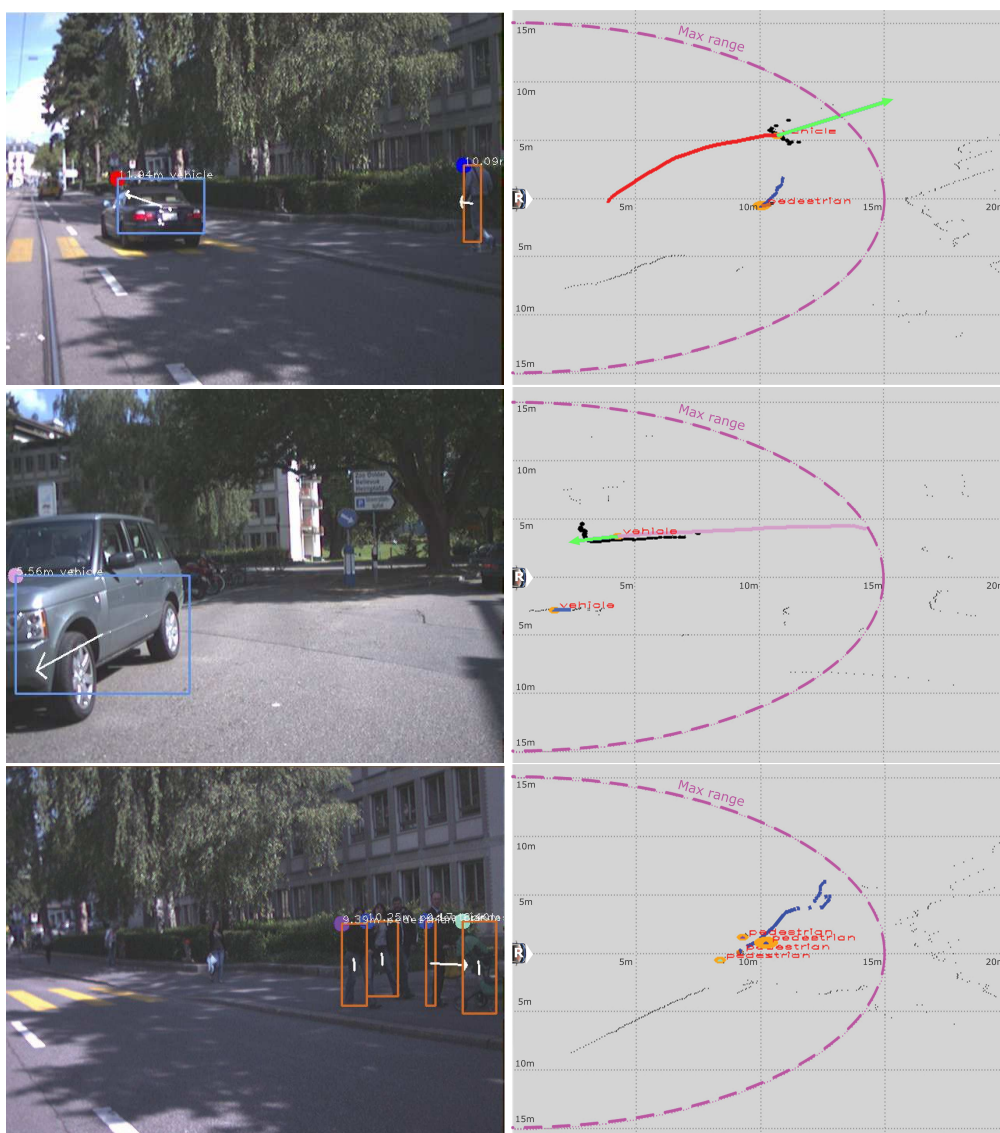


Figure 12: Cars and pedestrian detected and tracked under occlusion, clutter and partial views. In the camera images, left column, blue boxes indicate car detections, orange boxes pedestrian detections. The colored circle on the upper left corner of each box is the track identifier. Tracks are shown in color in the right column and plotted with respect to the robot reference frame. Green vectors show direction of motion for cars.



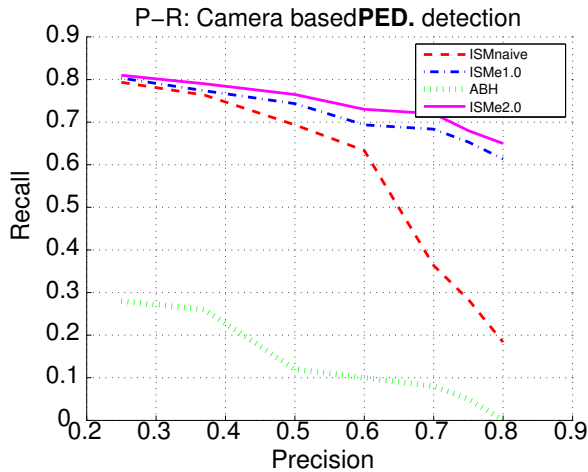


Figure 13: Quantitative evaluation for pedestrian detection. From left to right we compared the newly introduced multiclass technique with other approaches. The multiclass image based detector, ISMe2.0 is evaluated for the pedestrian category and it is compared with standard ISM (ISMnaive), a previous single class pedestrian detector ISMe (ISMe1.0) and AdaBoost with Haar features (ABH) classifier.

tracking results on combined data. Finally, we showed the usefulness of our approach through extended experimental results and comparisons on real-world data.

Future developments of this research are concerned specially with the integration of long range people detection. People in long range is described by few pixels in the image and few to none laser points. The idea is to integrate small scale detection methods [Spinello *et al.*, 2009] in the multimodal system by considering a more advanced tracking able to cope with very unreliable hypothesis. Other research directions involve development of robust data association filters, like MHT or JPDAF modeled for the multimodal detection problem.

## 9 Acknowledgement

This work has been partly supported by European Union under contract numbers BACS-FP6-IST-027140 and EUROPA-FP7-231888.

## A Index to Multimedia Extensions

The multimedia extensions to this article are at: <http://www.ijrr.org>.

Extension	Type	Description
1	Video	Multimodal detection and tracking of people and cars

## References

- [Ackermann, 1818] R. Ackermann. Improvements on axletrees applicable to four-wheeled carriages. In *Patent Nr. 4212*, 1818.
- [Arras *et al.*, 2007] K. O. Arras, Ó. M. Mozos, and W. Burgard. Using boosted features for the detection of people in 2d range data. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2007.
- [Arras *et al.*, 2008] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard. Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2008.
- [Bar-Shalom and Li, 1995] Y. Bar-Shalom and X.R. Li. *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, 1995.
- [Belongie *et al.*, 2002] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 24(4), 2002.
- [Borgefors, 1988] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 10, 1988.
- [Bourgeois and Lassalle, 1971] F. Bourgeois and J. C. Lassalle. An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12), 1971.
- [Comaniciu *et al.*, 2001] D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 2001.
- [Cox and Hingorani, 2002] I.J. Cox and S. L. Hingorani. An efficient implementation of reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *PAMI*, 18(2):138–150, August 2002.
- [Cui *et al.*, 2005] J. Cui, H. Zha, H. Zhao, and Shibasaki. Tracking multiple people using laser and vision. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2005.
- [Dalal and Triggs, 2005] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.
- [Douillard *et al.*, 2008] Bertrand Douillard, Dieter Fox, and Fabio Ramos. Laser and vision based outdoor object mapping. In *Robotics: Science and Systems (RSS)*, 2008.

- [Enzweiler and Gavrilu, 2009] M. Enzweiler and D.M. Gavrilu. Monocular pedestrian detection: Survey and experiments. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 31(12), 2009.
- [Felzenszwalb and Huttenlocher, 2000] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2000.
- [Fod *et al.*, 2002] A. Fod, A. Howard, and M. J. Matarić. A laser-based people tracker. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2002.
- [Freund and Schapire, 1997] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [Gavrila and Philomin, 1999] D. Gavrilu and V. Philomin. Real-time object detection for “smart“ vehicles. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.
- [Hähnel *et al.*, 2003] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2003.
- [Ioffe and Forsyth, 2001] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. Journ. of Computer Vision*, 43(1), 2001.
- [K.P. Li, 1988] J.E. Porter K.P. Li. Normalizations and selection of speech segments for speaker recognition scoring. In *Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, 1988.
- [Lafferty *et al.*, 2001] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In *Int. Conf. on Machine Learning (ICML)*, 2001.
- [Lau *et al.*, 2009] B. Lau, K. O. Arras, and W. Burgard. Tracking groups of people with a multi-model hypothesis tracker. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, Kobe, Japan, 2009.
- [Leibe *et al.*, 2005] B. Leibe, E. Seemann, and B. Schiele. Pedestrian detection in crowded scenes. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.
- [Leibe *et al.*, 2006] B. Leibe, K. Mikolajczyk, and B. Schiele. Segmentation based multi-cue integration for object detection. In *British Machine Vision Conf. (BMVC)*, 2006.
- [Leibe *et al.*, 2007] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2007.
- [Liu and Nocedal, 1989] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)), 1989.

- [Lloyd, 1982] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- [Luber *et al.*, 2008] M. Luber, K. O. Arras, C. Plagemann, and W. Burgard. Classifying dynamic objects: An unsupervised learning approach. In *Robotics: Science and Systems (RSS)*, 2008.
- [Mahalanobis, 1936] P.C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, 1936.
- [Mikolajczyk and Schmid, 2005] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.
- [Navarro-Serment *et al.*, 2009] L. Navarro-Serment, C. Mertz, and M. Hebert. Pedestrian detection and tracking using three-dimensional ladar data. In *Intern. Conf. of Field and Service Robotics (FSR)*, July 2009.
- [Papageorgiou and Poggio, 2000] C. Papageorgiou and T. Poggio. A trainable system for object detection. *Int. Journ. of Computer Vision*, 38(1), 2000.
- [Petrovskaya and Thrun, 2008] A. Petrovskaya and S. Thrun. Model based vehicle tracking for autonomous driving in urban environments. In *Robotics: Science and Systems (RSS)*, Zurich, Switzerland, June 2008.
- [Pless and Zhang, 2004] R. Pless and Q. Zhang. Extrinsic calibration of a camera and laser range finder. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2004.
- [Potts, 1952] R. B. Potts. Some generalized order-disorder transformations. *Cambridge Phil Soc.*, 48, 1952.
- [Premebida *et al.*, 2009] C. Premebida, O. Ludwig, and U. Nunes. Lidar and vision-based pedestrian detection system. *Journal of Field Robotics*, 2009.
- [Ramos *et al.*, 2007] F. Ramos, D. Fox, and H. Durrant-Whyte. CRF-matching: Conditional random fields for feature-based scan matching. In *Robotics: Science and Systems (RSS)*, 2007.
- [Reid, 1979] D. Reid. An algorithm for tracking multiple targets. *IEEE Transactions on Automatic Control*, 24(6), 1979.
- [Scheutz *et al.*, 2004] M. Scheutz, J. Mcraven, and G. Cserey. Fast, reliable, adaptive, bimodal people tracking for indoor environments. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2004.
- [Schulz *et al.*, 2003] D. Schulz, W. Burgard, D. Fox, and A. Cremers. People tracking with mobile robots using sample-based joint probabilistic data ass. filters. *Int. Journ. of Robotics Research (IJRR)*, 22(2), 2003.
- [Schulz, 2006] D. Schulz. A probabilistic exemplar approach to combine laser and vision for person tracking. In *Robotics: Science and Systems (RSS)*, 2006.

- [Schwarz, 1978] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2), 1978.
- [Sirovich and Kirby, 1987] L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4:519–524, 1987.
- [Spinello and Siegwart, 2008] L. Spinello and R. Siegwart. Human detection using multi-modal and multidimensional features. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2008.
- [Spinello *et al.*, 2008a] L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2008.
- [Spinello *et al.*, 2008b] L. Spinello, R. Triebel, and R. Siegwart. Multimodal people detection and tracking in crowded scenes. In *Proc. of the AAAI Conf. on Artificial Intelligence*, 2008.
- [Spinello *et al.*, 2009] L. Spinello, Alberto Macho, R. Triebel, and R. Siegwart. Detecting pedestrians at very small scales. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2009.
- [Spinello *et al.*, 2010] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart. A layered approach to people detection in 3d range data. In *Proc. of the AAAI Conf. on Artificial Intelligence*, 2010.
- [Streller *et al.*, 2002] D. Streller, K. C. Fuerstenberg, and K. Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Int. Conf. on Int. Tranp. Systems (ITSC)*, 2002.
- [Topp and Christensen, 2005] E. A. Topp and H. I. Christensen. Tracking for following and passing persons. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2005.
- [Viola *et al.*, 2003] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 2003.
- [Wender and Dietmayer, 2008] S. Wender and K. Dietmayer. 3D vehicle detection using a laser scanner and a video camera. *Intelligent Transport. Systems (IET)*, 2008.
- [Xavier *et al.*, 2005] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes. Fast line, arc/circle and leg detection from laser scan data in a player driver. In *IEEE Int. Conf. on Rob. and Autom. (ICRA)*, 2005.
- [Zhang, 1999] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE Int. Conf. on Computer Vision (ICCV)*, 1999.
- [Zhao and Thorpe, 1998] L. Zhao and C. Thorpe. Qualitative and quantitative car tracking from a range image sequence. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 1998.
- [Zheng and Liang, 2009] W. Zheng and L. Liang. Fast car detection using image strip features. In *CVPR*, 2009.

- [Zhu *et al.*, 2006] Q. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2006.
- [Zivkovic and Kröse, 2007] Z. Zivkovic and B. Kröse. Part based people detection using 2D range data and images. In *IEEE Int. Conf. on Intell. Rob. and Systems (IROS)*, 2007.

## Semantic Categorization of Outdoor Scenes with Uncertainty Estimates using Multi-Class Gaussian Process Classification

Rohan Paul<sup>‡</sup> Rudolph Triebel<sup>‡</sup> Daniela Rus<sup>†</sup> Paul Newman<sup>‡</sup>

<sup>‡</sup>Mobile Robotics Group, University of Oxford, UK

<sup>†</sup>Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, USA

{rohanp, rudi, pneyman}@robots.ox.ac.uk and rus@csail.mit.edu

**Abstract**—This paper presents a novel semantic categorization method for 3D point cloud data using supervised, multi-class Gaussian Process (GP) classification. In contrast to other approaches, and particularly Support Vector Machines, which probably are the most used method for this task to date, GPs have the major advantage of providing informative uncertainty estimates about the resulting class labels. As we show in experiments, these uncertainty estimates can either be used to improve the classification by neglecting uncertain class labels or - more importantly - they can serve as an indication of the under-representation of certain classes in the training data. This means that GP classifiers are much better suited in a *life-long* learning framework, where not all classes are represented initially, but instead new training data arrives during the operation of the robot.

### I. INTRODUCTION

To be able to perform complex tasks in its environment and at the same time communicate with a human user on a semantic level, any mobile robotic system needs some kind of semantic information about the environment. In most cases, and also in the context of this work, this semantic information is given in terms of object or class labels attached to sensor data that was acquired by the robot. To obtain such a labeling automatically, a mapping is usually learned from a set of feature vectors extracted from the sensor data into a given set of class labels. This can only be done using *supervised learning* methods, where a human expert manually annotates training examples which are then presented to a classification algorithm. The reason is obvious: class labels are defined by humans and can therefore not be “discovered” with unsupervised or similar learning techniques. In the robotics literature, a large body of work is already available on supervised learning methods for semantic annotation (e.g. object detection, scene analysis). Most of them use learning methods such as AdaBoost [1], Support Vector Machines (SVMs) [2], Probabilistic Graphical Models [3], [4], or other techniques such as Implicit Shape Models (ISM) [5]. Despite the impressive results of some of these systems, they all have one major drawback, which is of particular importance in mobile robotics: They assume the number of different class labels to be known beforehand. This means that the training data has to contain examples of all classes that can potentially be encountered during operation of the robot. All instances of unknown classes are then forced to correspond to one of the known classes, which leads to incorrect classifications.

In this paper, we propose a supervised learning method that has the potential to overcome this drawback.

We achieve this with a classifier that is based on a multi-class Gaussian Process (GP) classification algorithm. As we will show in experiments, our GP classifier can report uncertainty estimates about class labels in cases where the training data contained less classes than encountered in the test set. Thus, from these uncertainties there is implicit evidence that the classifier was trained with too few classes. Furthermore, these uncertainties can be used to select the next sensor observation that should be annotated by the human and added to the training data. This is a key requirement for an *active* learning system that is able to adapt its knowledge as it moves into new environments and thus learns during operation. Such an active and *life-long* learning system is currently the major goal of our line of research, which is the motivation for the need of the multi-class GP classification approach presented in this paper.

### II. RELATED WORK

Several methods for classification and labeling of 3D point cloud data have been presented in the literature. Here we review some related efforts. Angelov *et al.* [3] proposed a classifier based on an undirected graphical model (UGM), that automatically distinguishes between buildings, trees, shrubs and ground. This was later extended and applied to indoor data by Triebel *et al.* [4]. Posner *et al.* [6] present a multi-level classification framework for semantic annotation of urban maps using vision and laser features. The algorithm combines a probabilistic bag-of-words classifier with a Markov Random Field (MRF) model to incorporate spatial and temporal information. Xiong *et al.* [7] explicitly avoid the use of graphical models and suggest learning contextual relationships between 3D points based on logistic regression.

In contrast, Nüchter and Hertzberg [2] use a Support Vector Machine (SVM) to classify indoor objects in 3D range data. Marton *et al.* [8] introduce global radius-based surface descriptors (GRSD) and then use also an SVM for object classification. Golovinskiy *et al.* [9] segment 3D point clouds and compare several classifiers such as SVMs and random forests to detect objects in urban environments.

Several researchers have applied GPs in robotics mostly for regression applications rather than classification problems. For example, Plagemann *et al.* [10] and Vasudevan

*et al.* [11] use GPs for terrain modeling. Krause *et al.* [12] use GP regression for the problem of optimal placement of sensors and present a near-optimal mutual information based selection criterion. Stachniss *et al.* [13] employ a GP regression to determine a two-dimensional spatial model of gas distributions.

Classification using GPs has been addressed by Murphy and Newman [14], who present an approach for planning paths using terrain classification information from overhead imagery. Image regions are first classified using a multi-class GP classifier followed by spatial interpolation of uncertain terrain costs. Furthermore, Kapoor *et al.* [15] use GPs in an active learning framework for object categorization. However, in contrast to our approach, the problem there is not explicitly modelled as a GP classification problem, but rather as a GP regression where the labels are determined by *least-squares* classification. Also, the authors use a one-vs-all strategy based on binary classification rather than an explicit multi-class classifier.

### III. SEGMENTATION AND FEATURE EXTRACTION

Our algorithm operates on 3D point clouds acquired with a rotating laser scanner device. The first step in our tool chain after acquiring a new 3D point cloud, is to produce a triangular mesh by connecting neighboring data points if they are closer than a given threshold. We then compute normal vectors for all triangles and apply a segmentation algorithm based on the work of Felzenszwalb and Huttenlocher [16], where the similarity of two adjacent triangles is defined by the angles of their normal vectors. Each resulting mesh segment consists of a single connected component and is consistent with respect to the orientation of the triangles it contains. Thus, segments are consistently shaped, e.g. all triangles are all mostly co-planar or they are all similarly distributed in orientation. An example result of our segmentation algorithm can be seen in Figure 1.

In the next step, we compute feature vectors for all mesh segments. We use similar features as in earlier work [17], namely *shape factors*, *shape distributions* based on Euclidean distance, on angles between normal vectors and on the elevation of the normal vectors, and finally *spin images*, where the latter are computed per data point and then an average is computed per mesh segment. As a result, we obtain a 113 dimensional feature vector for each mesh segment, where 50 account for the  $5 \times 10$  spin image, 20 for each shape distribution (i.e. the number of histogram bins) and 3 for the shape factors. These feature vectors, together with a set of ground truth class labels are then fed into the training algorithm of the GP multiclass classifier as described next.

### IV. MULTI-CLASS CLASSIFICATION USING GAUSSIAN PROCESSES

Let  $\mathbf{x} = \mathbf{x}_1, \dots, \mathbf{x}_n$  be a given set of  $n$  feature vectors with dimensionality  $d$  and  $\mathbf{y} = y_1, \dots, y_n$  corresponding class labels where  $y_i \in \{1, \dots, k\}$  and  $k$  is the number of classes. To formulate the multi-class classification problem using a Gaussian Process (GP), a *latent* function  $f_j(\mathbf{x})$  is introduced

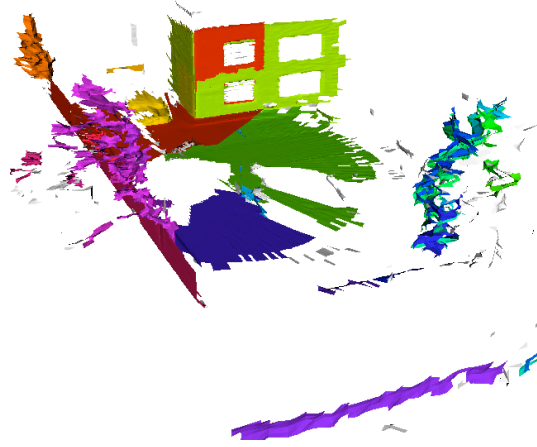


Fig. 1: Mesh segmentation. Every segment has a different color assigned. In this example, the mesh has self-overlapping parts, which is why e.g. the building is split into several segments. Note that “rough” surfaces such as those on the trees are segmented as well as smoother regions such as the ground.

for each class along with the *probit regression* model. The probability of a class label  $y_i$  for a given feature vector  $\mathbf{x}_i$  is defined as:

$$p(y_i = j | \mathbf{x}_i) = \Phi(f_j(\mathbf{x}_i)) \quad i = 1, \dots, n, \quad j = 1, \dots, k, \quad (1)$$

where  $\Phi$  denotes the standard normal cumulative distribution function, i.e.  $\Phi(z) = \int_{-\infty}^z \mathcal{N}(x | 0, 1) dx$ . The latent function  $f$  is represented by a Gaussian Process, determined by a mean function – which in our case is the zero function – and a covariance function  $k(\mathbf{x}_p, \mathbf{x}_q)$ , usually denoted as the *kernel function*. Several different kinds of kernel functions are used in the literature, where the most common ones are the squared exponential, which is also denoted as the Gaussian kernel function. It is defined as

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp\left((\mathbf{x}_p - \mathbf{x}_q)^T D (\mathbf{x}_p - \mathbf{x}_q)\right) \quad p, q = 1, \dots, n, \quad (2)$$

where  $D$  is a  $d \times d$  diagonal matrix. The diagonal entries of  $D$  are known as the *hyper-parameters* of the model.

In contrast to other supervised learning methods for classification such as Support Vector Machines (SVMs), Gaussian Processes are non-parametric, which means that there is no explicit computation of model parameters in the training step. However, in GP classification there still needs to be done some training to obtain the hyperparameters of the covariance function and the posterior distribution of the latent function. More specifically, the aim of the classifier is to find the distribution over values of the latent function given the training data and some test input  $\mathbf{x}^*$ , i.e.

$$\begin{aligned} p(f^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}, \mathbf{x}^*) \\ = \\ \int p(f^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{x}^*, \mathbf{f}) p(\mathbf{f} | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}) d\mathbf{f}, \end{aligned} \quad (3)$$



where we use the notation  $\mathbf{x}^*, f^*$  to refer to the test input and its function value. This distribution is then used to compute the class probabilities:

$$p(y^* = j | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}, \mathbf{x}^*) = \int p(y^* | f_j^*) p(f_j^* | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y}) df_j^*. \quad (4)$$

The main problem here is that the latent posterior  $p(\mathbf{f} | \mathbf{x}_1, \dots, \mathbf{x}_n, \mathbf{y})$  is not Gaussian and hence Equation (3) can not be computed in closed form. Therefore, approximations need to be done, and the main approaches to do this are the Laplace approximation and Expectation Propagation, as described in [18]. In this paper, we follow the approach of Girolami and Rogers [19], where a variational Bayes formulation is used. During training the hyperparameters are learned by gradient ascent on the estimated marginal likelihood. Once the hyperparameters and the latent posterior are obtained from training data, inference is performed on new test input by applying Equation (4).

The full GP classification procedure scales as  $O(kn^3)$  where  $k$  is the number of classes and  $n$  is the total number of sample points. The scaling is dominated by the cubic dependence on  $n$  due to the matrix inversion required to obtain the posterior mean for the GP variables. The variational bayes multi-class GP formulation [19] is amenable to a sparse approximation by constraining the maximum number of samples  $s$  included in the model. This results in an  $O(kns^2)$  scaling where  $s \ll n$ . The informative points are picked according to the posterior predictive probability of the target value, intuitively picking points from class boundary regions which are most influential in updating the approximation of the target posteriors. For a detailed exposition please refer to [19], [20] and [21].

Compared to a discriminative classifier like SVM, the GP classification framework offers certain benefits making it particularly suitable for our application. GPs possess a probabilistic formulation and express belief over the latent function via *marginalization* as opposed to *minimization* in SVMs and hence provide uncertainty estimates for the distribution over classification labels [22]. This is more principled than a heuristic approach of using the distance from the classification boundary (margin). A high uncertainty in the GP classification output distribution can give evidence for a category not modeled during training and hence can be used to actively seek examples for incremental training. Additionally, the GP kernel parameters and noise models are interpretable and can be learned without cross validation, which is significant if less data is available for a rare category.

## V. EXPERIMENTAL RESULTS

In the following experiments, four statements will be shown. First, the multi-class GP classifier gives very good classification results on our 3D outdoor data. Second, misclassifications can be detected, because the GP classifier provides uncertainty estimates about the resulting labels. This



Fig. 2: Our robotic car, equipped with a 3D laser range finder on the top of the roof.

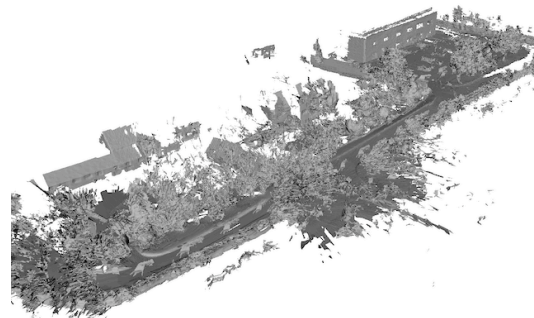


Fig. 3: Mesh representation of our test site. The area mainly consists of buildings, trees, hedges, and roads. The data was acquired with our robotic car “Wildcat”, which is equipped with a 3D laser scanning device and very accurate positioning sensors.

can be used to improve the precision of the classifier even further. Third, in comparison with SVMs, which are probably the most often used method in robotics, GPs perform at least equally well, even if they are chosen to be sparser than the SVM. And finally and most importantly, when trained with too few classes (in our case two instead of six), the estimated class label uncertainties are much higher when using GPs, making them much more useful for detecting unknown classes in the training set.

### A. Data sets and training

We acquired data with our autonomous car *Wildcat* (see Figure 2), equipped with a 3D scanning device consisting of three SICK LMS-151 laser scanners that are mounted vertically on a rotating turn table. The rotation frequency was set to  $0.1\text{Hz}$ . For our experiments, we drove the car slowly ( $\approx 15\text{km/h}$ ) around our research site at Begbroke science park in Oxfordshire. A mesh representation of the acquired data is shown in Figure 3. The data we obtained is comparably dense: each point cloud consists of 100,000 to 150,000 points.

PCA was used for dimensionality reduction retaining 10 principal components from the original 113 dimensional feature vector. Figure 4 (left) plots the eigen magnitudes obtained. Note that very few principal directions capture most of the data variance. To perform the multi-class GP classification we used the Variational Bayes Sparse Gaussian Process approach by Girolami and Rogers [19]. During training the (in-sample) marginal likelihood was monitored

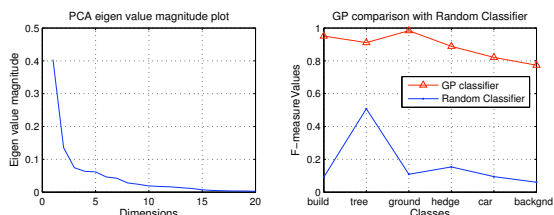


Fig. 4: Left: Plot of eigen value magnitudes after PCA on 113 dimensional feature vectors. Note that very few principal directions capture most of the data variance. Right:  $F_{0.5}$ -measure comparison of GP classifier with a naive classifier making random decisions based on relative sample frequency. The random classifier performs much worse than the GP classifier.

for convergence within 1% increase tolerance. The process converged for all runs within 45 conjugate gradient iterations.

For evaluating the classification performance of the system, a subset consisting of 1497 segments from 53 lidar point clouds was hand-labeled into six categories frequently encountered in outdoor urban scenes: *building*, *tree*, *ground*, *hedge*, *car* and *background*. The data set was randomly split into test and training with test fractions varying as 0.3, 0.5 and 0.8. Note that our test data was unbalanced, there were more segment instances for some classes like *trees* and *ground* compared to *building* and *cars* due to their shape complexity (reflecting in the number of segments) and natural occurrence frequency in the environment. As suggested in [6], for a more realistic evaluation of the classifier in real settings the data set was not equalized. Thus, we report classifier performance per-class instead of average due to unbalanced class sizes.

### B. Quantitative results

The GP classifier gives a distribution over labels for test data. By taking the maximum-likelihood class assignment, the per-class precision and recall values were estimated and listed in Table I for the run with test fraction 0.5. Precision and recall can be combined into a  $F_\beta$ -measure as given in Equation 5. Here, parameter  $\beta$  refers to the relative importance assigned to recall performance over precision. As suggested in literature [6], we use  $\beta = 0.5$  assigning greater importance to precision accuracy over recall.

$$F_\beta = \frac{(1 + \beta^2)(\text{precision} \times \text{recall})}{(\beta^2 \text{precision} + \text{recall})} \quad (5)$$

The classifier attains high  $F_{0.5}$ -measure performance for *ground* (0.98) and *building* (0.95) and lower accuracy for classes *hedge* (0.89), *car* (0.82) and *background* (0.77).

Figure 4 (right) compares the GP classifier performance ( $F_{0.5}$ -measure) with a naive classifier making random decisions based on class frequencies in training data. The accuracy of the random classifier is much worse than the GP classifier.

Figure 5 visualizes the confusion matrix where values are normalized along rows. Hence, diagonal values represent per-class recall, indicating the extent to which the ground truth assignments are retrieved. Note that categories *car*, *background* and *hedge* are confused in recall with the *tree*

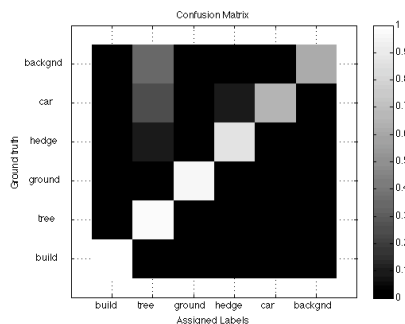


Fig. 5: Confusion matrix (normalized) resulting from the GP classifier. Recall values appear along the diagonal. Results with test fraction: 0.5.

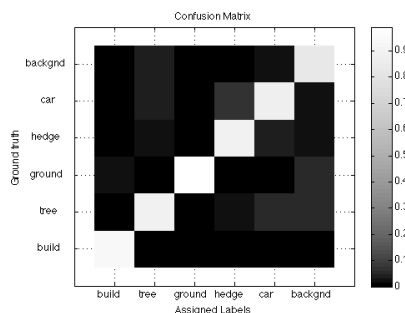


Fig. 6: Confusion matrix (normalized) resulting from the GP classifier. Precision values appear along the diagonal. Results with test fraction: 0.5.

class. Figure 6 presents the confusion matrix with values normalized vertically. Each column represents the accuracy of the classifier labeling and the diagonal values represent precision. Overall, the classifier shows good precision performance. Some confusion is observed between *hedge* and *car* categories.

Next, we calculated the entropy values for each label distribution for a segment to quantify the uncertainty in the classification of that segment. This was normalized to 0 to 1 range by dividing by  $\log(k)$  the maximum entropy of a uniform distribution over  $k$  class labels. An incorrect maximum likelihood assignment frequently results when the label distribution entropy is high. By thresholding the normalized entropy in increments from 0 to 1 and considering assignments only where the classifier is certain above the threshold, the class-specific precision and recall values are calculated. Figure 7 plots the precision-recall curves for two runs with test fractions 0.5 and 0.8. For the case with test

TABLE I: Precision, Recall and  $F_{0.5}$ -measure performance for GP classification for all six categories in the data set. Test fraction: 0.5

Name	train:test	Precision	Recall	$F_{0.5}$ -measure
<b>Building</b>	70 : 62	0.94	1.00	0.95
<b>Tree</b>	362 : 357	0.90	0.98	0.91
<b>Ground</b>	114 : 115	0.99	0.96	0.98
<b>Hedge</b>	91 : 100	0.90	0.86	0.89
<b>Car</b>	74 : 75	0.88	0.64	0.82
<b>Background</b>	37 : 45	0.82	0.62	0.77

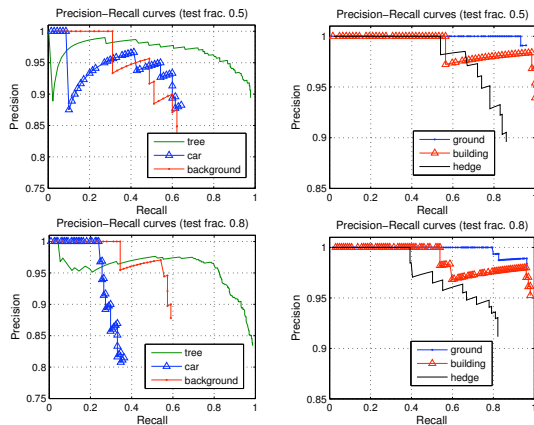


Fig. 7: Precision-recall curves (per-class) obtained by thresholding on the normalized entropy of the label distribution for classes. Left: Classes *tree*, *car* and *background*. Right: Classes *building*, *ground*, and *hedge*. Top: Plots with test fraction: 0.5. Below: Plots with test fraction: 0.8. Note the scale on y-axis.

fraction 0.5, Figure 7 (top), classes *ground*, *building* and *hedge* attain 100% precision at a maximum recall of 93%, 56% and 54% respectively. By accepting a slightly lower precision of 90%, nearly 100% of the ground truth can be retrieved for *building* and *ground* classes and 86% for class *hedge*. The curves for classes *car* and *background* are lower. At 90% precision both classes have a lower recall of 60%.

When the GP is trained with a higher test fraction of 0.8 a general decline is observed in the precision-recall performance as shown in Figure 7 (bottom). The decrease is small for classes like *building*, *ground* and *hedge* and is more significant for the class *car* for which recall decreases from 60% to 27% at 90% precision level. In both experiments, class *tree* displays a gradually declining curve which may be attributed to significant variation in the entropy values from a large number and varied segments obtained as category samples. In general, using a lower normalized entropy threshold was found to improve precision at the cost of lowering recall since uncertain true positive class labels are also suppressed. This allows the user to obtain an application specific value of the entropy threshold.

### C. Qualitative results

Figure 8 shows an example of the classification result for one triangle mesh from our data set. The left image shows the ground truth labeling obtained from manual annotation. The center image depicts our classification result using the multi-class GP classifier. One can clearly see that there are only minor classification errors. The most obvious ones are in the front on the hedge surrounding the car park. Here, the classifier generated the label *car*. However, the labels in that area are not very certain, which can be seen from the right image in the figure. Here, the normalized entropy is visualized with color values between green (no entropy) and red (entropy equal to 1). We can see that the class label distributions of the segments in the front have a much higher entropy than others such as those on the ground. This

means that the classification result can be improved even further, if required, by neglecting all label assignments where the entropy of the label distribution is too high. Of course, such a conservative classifier will have a weaker recall performance (as shown in the previous section), but in some applications the reduction of false-positive classifications is more important.

Figure 9 shows the classification result of 9 consecutive meshes in one common image. We note that there are slight labeling errors even in the ground truth (left image). This is caused by imperfections in the segmentation process, which lead to under-segmentation. For example, some few segments contain sensor readings from the building and the ground. As it is impossible to determine a single *true* label for these segments, we decided to assign the ground truth label based on a majority voting during annotation. From the figure we can see that the qualitative result corroborates the outcome of the quantitative evaluation: in general, the classification is very good, only the under-represented classes such as *car* and *hedge* are classified slightly worse.

### D. Classifier Comparison

We compared the generative GP classifier with a discriminative SVM classifier using the LIBSVM implementation of Chang and Lin [23]. In all cases, we employed the squared exponential kernel to facilitate comparison. Table II compares the  $F_{0.5}$ -measure performance of the two classifiers with test fractions: 0.3, 0.5 and 0.8. The total number of support vectors (indicating model sparsity) obtained during SVM training were noted for each run. The GP classifier sparsity parameter  $S$  was set to a value close to but smaller than the number of support vectors used by SVM. The  $F_{0.5}$ -measure performance for the GP and SVM classifiers was very similar, even with a sparser representation used for the GP. The experimental result accords with similar findings by Naish-Guzman *et al.* in [24].

Next, we compared the uncertainty estimates of the probabilistic classification output of the two classifiers to new object classes not used in training. We trained the GP and SVM classifiers only on segments from two classes (randomly picked): *building* and *ground*. Data from the remaining *un-modeled* classes was presented to both classifiers for inference, resulting in a classification distribution over binary labels. The normalized entropy values measuring uncertainty in the classification decision were computed for each label distribution.

Figure 10 presents the normalized entropy histograms for the inference set. The SVM classifier commits a large majority of the *un-modeled* points to one of the *modeled* classes with high certainty, resulting in a peaked distribution over one of the two labels. As a result, for a majority of the data points, the label distribution has lower normalized entropy. In contrast, the GP classifier assigns higher normalized entropy for a majority of the test points. The same pattern was found consistent for other choices of training and testing classes. The classifier uncertainty for the test points from new classes is expressed as a more uncertain (uniform) distribution over

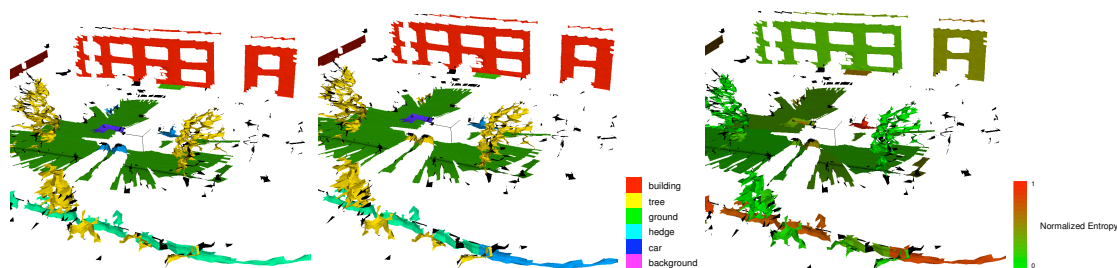


Fig. 8: Classification result and normalized entropy for one example mesh. **Left:** Ground truth labeling. In this scene, no background objects were present. **Center:** Classification result using multi-class GP classification. Note the classification error of the *hedge* in the front, which is classified as *car*. **Right:** Normalized entropy of the class label distributions for each mesh segment. For most segments the classifier is very confident. For some, such as the (wrong classified) *hedge* in the front, the normalized entropy is high and thus the classification confidence low.

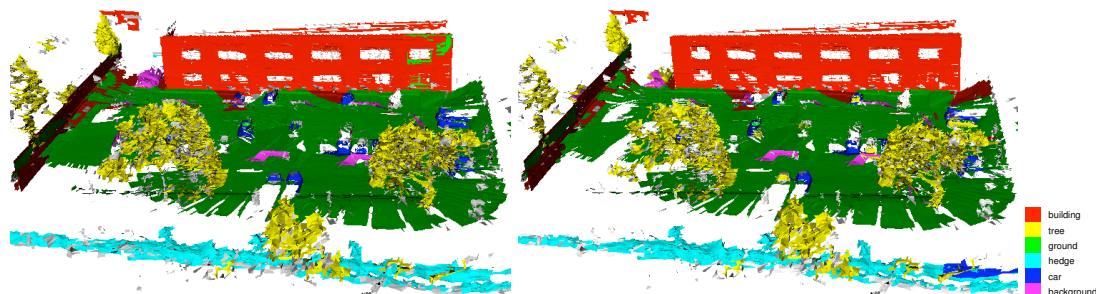


Fig. 9: Classification result after 9 point clouds (time steps). **Left:** Ground truth. Note that even in the ground truth some areas are not labeled correctly, e.g. on the ground close to the *building*. This is due to the fact that the mesh segmentation is not perfect and a correct manual labeling of segments that actually correspond to more than one class is not possible. In our evaluation we abstract from such segmentation errors. **Right:** Classification result. Only minor errors are visible. Note again the *hedge* in the front, but also on some *cars*.

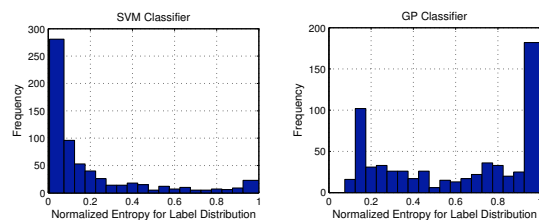


Fig. 10: Histogram of normalized entropy values of the label distribution for SVM and GP classifiers. Both classifiers were trained explicitly on two classes. The data from the remaining classes was presented for inference. **Left:** SVM classifier assigns a majority of the points to a particular class with high certainty. **Right:** As a contrast, GP classifier assigns greater classification uncertainty to a majority of the points, providing evidence for a potential new class. Note the scale on y-axis.

labels, indicating the presence of one or more potentially *un-modeled* classes.

## VI. CONCLUSIONS AND FUTURE WORK

The mid-term goal in our current research is an actively learning mobile robotic system that acquires semantic knowledge by supervision, but during system operation, i.e. in a *life-long* learning framework. However, this knowledge needs to be added *incrementally* and *selectively*, because no human would be willing to annotate all new sensor observations from the robot. Unfortunately, none of the currently used supervised learning algorithms can provide sufficient means

to select the next observation that needs human annotation. In this paper, we show that when using multi-class GP classification this selection actually *can* be done based on the uncertainty estimates of the class labels that the GP classifier inherently provides. We also show that there is no loss in performance when using a GP classifier, even if a higher level of sparsification is chosen. These results demonstrate the power of the GP classifier for this purpose, and thus provide an important step towards *life-long* learning robot systems.

## VII. ACKNOWLEDGEMENTS

Authors wish to thank Dr. Ingmar Posner for insightful discussions and Dr. Benjamin Davis for maintaining the robotic platform used for this work. Paul Newman was supported by an EPSRC Leadership Fellowship, EPSRC Grant EP/I005021/1. Daniela Rus was supported for this work in parts by the MAST Project under ARL Grant W911NF-08-2-0004 and ONR MURI Grants N00014-09-1-1051 and N00014-09-1-1031.

## REFERENCES

- [1] Ó. M. Mozas, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Journal on Robotics and Autonomous Systems (RAS)*, vol. 55, no. 5, pp. 391–402, 2007.
- [2] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Journal of Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008.

TABLE II:  $F_{0.5}$ -measure classification performance comparison for GP and SVM with varying test data fractions.

Experiment	Test Fraction 0.3		Test Fraction 0.5		Test Fraction 0.8	
	GP #s : 450	SVM #sv : 547	GP #s : 300	SVM #sv : 411	GP #s : 100	SVM #sv : 179
<b>Building</b>	0.99	0.99	0.95	0.95	0.96	0.96
<b>Tree</b>	0.89	0.91	0.92	0.92	0.86	0.88
<b>Ground</b>	0.97	0.97	0.98	0.97	0.98	0.98
<b>Hedge</b>	0.83	0.82	0.87	0.87	0.88	0.91
<b>Car</b>	0.66	0.67	0.84	0.82	0.63	0.68
<b>Background</b>	0.62	0.73	0.75	0.82	0.79	0.77

- [3] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d scan data," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005, pp. 169–176.
- [4] R. Triebel, R. Schmidt, O. M. Mozas, and W. Burgard, "Instance-based AMN classification for improved object recognition in 2d and 3d laser range data," in *Proc. of the Intern. Joint Conf. on Artificial Intell.*, 2007.
- [5] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3d range data," in *special track on Physically Grounded AI of AAAI*, 2010.
- [6] I. Posner, M. Cummins, and P. Newman, "A generative framework for fast urban labeling using spatial and temporal context," *Autonomous Robots*, vol. 26, no. 2, pp. 153–170, 2009.
- [7] X. Xiong, D. Munoz, J. A. Bagnell, and M. Hebert, "3-d scene analysis via sequenced predictions over points and regions," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.
- [8] Z. C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D-3D Categorization and Classification for Multimodal Perception Systems," *The International Journal of Robotics Research*, 2011, accepted for publication.
- [9] A. Golovinskiy, V. G. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *International Conference on Computer Vision (ICCV)*, 2009.
- [10] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, "Learning predictive terrain models for legged robot locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.
- [11] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte, "Gaussian process modeling of large scale terrain," *Journal of Field Robotics*, vol. 26, no. 10, pp. 812–840, 2009.
- [12] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *The Journal of Machine Learning Research*, vol. 9, pp. 235–284, 2008.
- [13] C. Stachniss, C. Plagemann, and A. J. Lilienthal, "Gas distribution modeling using sparse gaussian process mixtures," *Autonomous Robots*, vol. 26, no. 2-3, pp. 187–202, April 2009.
- [14] L. Murphy and P. Newman, "Planning most-likely paths from overhead imagery," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010.
- [15] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.
- [16] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.
- [17] R. Triebel, J. Shin, and R. Siegwart, "Segmentation and unsupervised part-based discovery of repetitive objects," *Proceedings of Robotics: Science and Systems*, Jan 2010.
- [18] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006, no. ISBN 0-262-18253-X.
- [19] M. Girolami and S. Rogers, "Variational bayesian multinomial probit regression with gaussian process priors," *Neural Computation*, vol. 18, no. 8, pp. 1790–1817, 2006.
- [20] M. Seeger and M. Jordan, "Sparse gaussian process classification with multiple classes," Citeseer, Tech. Rep., 2004.
- [21] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," *Advances in neural information processing systems*, vol. 15, pp. 609–616, 2002.
- [22] C. Williams and C. Rasmussen, "Gaussian processes for machine learning," 2006.
- [23] C. Chang and C. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.
- [24] A. Naish-Guzman and S. Holden, "The generalized fitc approximation," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1057–1064, 2008.

# Knowing When We Don't Know: Introspective Classification for Mission-Critical Decision Making

Hugo Grimmer Rohan Paul Rudolph Triebel Ingmar Posner

Mobile Robotics Group, University of Oxford, UK

{hugo, rohanp, rudi, ingmar}@robots.ox.ac.uk

**Abstract**—Classification *precision* and *recall* have been widely adopted by roboticists as canonical metrics to quantify the performance of learning algorithms. This paper advocates that for robotics applications, which often involve mission-critical decision making, good performance according to these standard metrics is desirable but *insufficient* to appropriately characterise system performance. We introduce and motivate the importance of a classifier's *introspective* capacity: the ability to mitigate potentially overconfident classifications by an appropriate assessment of how qualified the system is to make a judgement on the current test datum. We provide an intuition as to how this introspective capacity can be achieved and systematically investigate it in a selection of classification frameworks commonly used in robotics: support vector machines, LogitBoost classifiers and Gaussian Process classifiers (GPCs). Our experiments demonstrate that for common robotics tasks a framework such as a GPC exhibits a superior introspective capacity while maintaining commensurate classification performance to more popular, alternative approaches.

## I. INTRODUCTION

The semantic mapping of a robot's workspace has become a popular line of research in recent years. A rich body of work now exists in which semantic labels are generated based on a variety of sensor modalities and classification frameworks (see, for example, [1]–[7]). Often, this is done with an implicit understanding that the application is agnostic to the classification method used: after all, for a number of classification frameworks the resulting *precision* and *recall* — quantities commonly used to characterise performance — are often commensurate across a wide variety of applications.

Contrary to this now established status-quo, we advocate that high precision and recall are desirable but do not suffice to fully characterise classification performance in robotics. The dimension missed is that spawned by a robot's ability to take action in ambiguous situations. For example, the robot may query a human operator or seek additional data for disambiguation rather than committing to a potentially incorrect class decision. Crucially, and central to this paper, this requires the classifier output to reflect an amount of ambiguity appropriate to a given situation. Even when hard class assignments are avoided by optimising an expected cost or reward, as is the case for most mission-critical decision making, a *realistic* estimate of uncertainty when modelling the state of the world is crucial; an autonomous car that misses a single traffic light with high confidence can suffer disastrous consequences (see, for example, Fig. 1).

We argue that a classifier which is uncertain when it makes mistakes but certain when classification is correct, is more desirable than a classifier which makes correct and



Fig. 1: Uncertainty in classification output as measured using normalised entropy for traffic light detectors based on five different classification frameworks applied to the window shown in green. Note that *all classifiers incorrectly* label this window as *background* (class decisions are not shown). However, the GPC variants do so with a significant amount of uncertainty while the others are inappropriately overconfident. Mission-critical decisions based on overconfident output will lead to catastrophic failure while an appropriately high amount of uncertainty when committing a mistake allows for remedial action to be taken. Providing this more germane output is the introspective quality we seek.

incorrect decisions with similarly high confidence. We are therefore looking for a classifier's capacity to mitigate its assessment by an appropriate measure as to how 'qualified' it is to make a call given its own prior experience, usually in the form of training data. Following classical decision theory (e.g. [8]), mistakes are penalised by means of a loss function. However, if the underlying classification framework leads to an overconfident estimate of the class label, then it will often be ineffective regardless of the high costs imposed. Our work investigates this *introspective* capacity in a number of classification frameworks commonly used in robotics: support vector machines (SVMs), LogitBoosting and Gaussian Process classifiers (GPCs). Our treatment and findings apply to any aspect of robotics where action is required based on inference driven by raw sensor data. Here we choose to frame our exposition in the domain of autonomous driving, where mission-critical decisions equate to *safety*-critical decisions. To the best of our knowledge this is the first work in robotics characterising the introspective properties of commonly used classification frameworks.

## II. RELATED WORKS

For a number of years now robots have routinely generated and consumed higher-order abstractions from raw sensor data. Successful applications are as diverse as the detection of ground traversability (e.g. [9]), the detection of lanes for autonomous driving (e.g. [10]), the consideration of classifier output to guide trajectory planning and exploration (see, for example, [11], [12]) or the active disambiguation of human-robot dialogue [13]. These works commonly exploit classification output on a model-trust basis: systems are optimised with respect to precision and recall and egregious misclassifications — including vastly over-confident marginal distributions obtained from some classification frameworks — are accepted as par for the course. However, the suitability of the classification framework employed with respect to its introspective capacity has not previously been considered in robotics. Thus, we consider motivating, defining and investigating introspection in a robotics context to be the primary contribution of our work.

The concept of introspection as introduced here is closely related to considerations in active learning, where uncertainty estimates and model selection steps are often employed to guide data selection and gathering for an incremental learning algorithm. Kapoor *et al.* [14], for example, present an active learning framework for object categorization using a GPC where classifications of large uncertainty (as judged by posterior variance) lead to a query for a ground-truth label and are subsequently used to improve classification performance. Joshi *et al.* [15] address multi-class image classification using SVMs and propose criteria based on entropy and best-versus-second-best measures (see Section III) for disambiguating uncertain classifications. Holub *et al.* [16] propose an information-theoretic criterion that maximises expected information gain with respect to the entire pool of unlabeled data available. Hospedales *et al.* [17] discuss optimising rare class discovery and classification using a combination of generative and discriminative classifiers.

Our treatment of introspection is further informed by an ongoing discussion in the machine learning community regarding how to best account for variance in the space of feasible classifier models when training on, essentially, an incomplete set of data. For example, Tong and Koller [18] present an incremental algorithm for text classification using SVMs and the notion of a *version space*, the set of consistent hyperplanes separating the data in a feature space induced by the kernel function. Zhang *et al.* [19] introduce a max-margin classifier achieving better generalisation to unseen test data given a limited training corpus. Here, distinctiveness of training instances is assessed using the local classification uncertainty. A global classifier then incorporates these uncertainties as margin constraints, yielding a classifier that places less confusing instances farther away from the global decision boundary. We share the intuition that accounting for variance in version space when selecting a model leads to an increased introspective capacity. As a secondary contribution, therefore, our results serve to further corroborate this intuition.

## III. INTROSPECTION AND UNCERTAINTY

Introspection concerns not the final class decision but rather the confidence with which this decision is made. The concept is motivated by the desire to take appropriate action when a classifier indicates high uncertainty. Our approach to introspection is grounded in the fact that the often cited assumption of independent and identically distributed (*iid*) training and test data is routinely violated in robotics: in the limit of continuous operation in the real world, one-shot classifier training is unlikely to be performed on a complete (or even fully representative) set of data.

Let a classifier map an input  $\mathbf{x} \in \mathbb{R}^d$  to one of a set of classes  $C = \{C_1, \dots, C_{|C|}\}$  via an associated label  $y \in C$ . Prior to training, domain specific knowledge is often used to constrain the family of classification models employed (for example in the form of a kernel, a covariance function or a type of base classifier). Classifier training then involves learning a set of (hyper-) parameters given a training dataset  $\{\mathcal{X}, \mathcal{Y}\}$ , where  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{|\mathcal{X}|}\}$  denotes the set of feature vectors and  $\mathcal{Y}$  denotes the set of corresponding class labels. The training data implicitly give rise to a probability distribution over the set of all possible models within the chosen family,  $\mathcal{M}$ , such that

$$\{\mathcal{X}, \mathcal{Y}\} \rightarrow p(m | \mathcal{X}, \mathcal{Y}), \quad m \in \mathcal{M}. \quad (1)$$

With a slight abuse of notation,  $m$  here denotes any member of the family of possible models,  $\mathcal{M}$ . In reality it is a function of the datum evaluated. In the following we make this relationship explicit by conditioning on both a model (or family of models) as well as on a test datum  $\mathbf{x}_*$ . Typically, training leads to the selection of a *single* model,  $\tilde{m}$  from  $\mathcal{M}$  such that a prediction  $y_*$  for a new, unseen feature vector  $\mathbf{x}_*$  is obtained by approximating

$$p(y_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* | \tilde{m}, \mathbf{x}_*), \quad \tilde{m} \in \mathcal{M}. \quad (2)$$

This is illustrated in Figure 2(a). Common examples of this type of classification framework include SVMs and Boosting classifiers, where an optimisation is performed to select the best model given the training data (see Section IV). The *iid* assumption here is inherent since it is assumed that  $\tilde{m}$  remains the best model for all predictions of unseen data. Breaking this assumption therefore often renders the chosen model suboptimal.

An alternative to the single model approach are classification frameworks which take into account the *entire set* of possible models in the specified family, such that

$$p(y_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) \approx p(y_* | \mathcal{M}, \mathbf{x}_*). \quad (3)$$

This case is illustrated in Figure 2(b). Here the shading indicates the distribution  $p(m | \mathcal{X}, \mathcal{Y})$  with darker shades indicating increased probability. To aid intuition, predictions of four randomly selected members of  $\mathcal{M}$  are also illustrated. Final predictions are made by taking into account opinions from all members of  $\mathcal{M}$ , often via the computation of an expectation such as for a GPC (see Section IV). Crucially, when considering an expectation over all of  $\mathcal{M}$ , the increased variance in feasible (and therefore likely) models at a distance from the training data leads to a moderation of the class predictions. This is the introspective quality we seek.

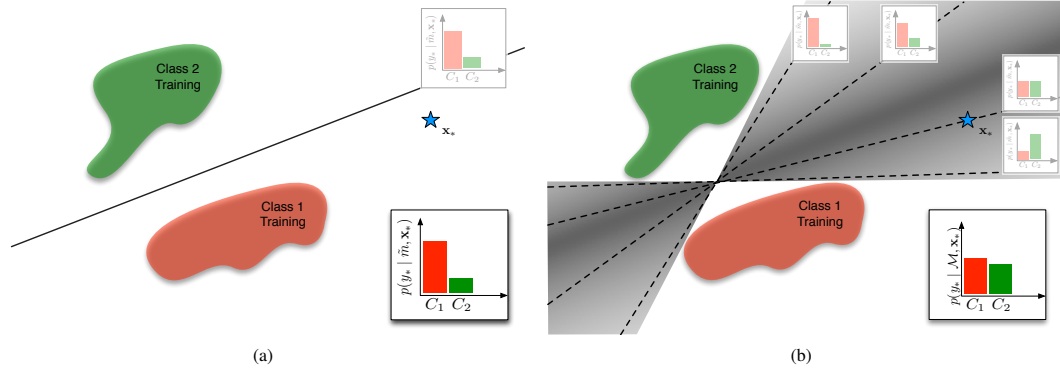


Fig. 2: An illustration of the two types of classification frameworks considered: (a) during training a *single* model is selected to classify an unknown datum  $\mathbf{x}_*$  some way removed from the training data; (b) training leads to a distribution over models which is considered entirely to arrive at the final prediction. This illustration is for the family of linear models (indicated by solid (a) and dashed (b) lines). Each predictor is further annotated with its individual prediction. The overall predictive distribution is shown in the bottom right of each subplot. The shading in part (b) indicates the probability weights associated with individual models. Note that the overall predictive distribution in (a) stems from the single model used and is, in this case, inappropriately confident. In part (b), however, the overall predictive distribution is moderated by computing the expectation over all models. This gives rise to a much more appropriate uncertainty estimate — the introspective quality we seek. (Best viewed in colour.)

#### A. Quantifying Introspection

In order to characterise the introspective capacity of a classification framework a transferable measure of the inherent uncertainty in the classification output is required. For this purpose, we use an information-theoretic quantity known as normalised entropy,  $H_N$ , defined as

$$H_N = - \sum_{C_i \in \mathcal{C}} p(y = C_i | \mathbf{x}) \log_{|C|} [p(y = C_i | \mathbf{x})]. \quad (4)$$

This is equivalent to the Shannon entropy measure normalised by its maximum, which is the entropy of the  $|C|$ -dimensional uniform distribution,  $\log(|C|)$ . The result is a measure ranging between 0 and 1 where a *higher* value indicates *greater* uncertainty in the classifier's belief.

An alternative uncertainty measure proposed in the active learning literature is the best-versus-second-best (BvSB) heuristic [15] calculated as the difference between the largest and the second largest class likelihood estimates. This measure attempts to characterise the reliability of the maximum likelihood estimate rather than encoding the shape of the full distribution over class labels. The BvSB and normalised entropy measures are closely related in the binary-classification setting which is the case in this paper. We use normalised entropy throughout the remainder of this work due to its appealing information-theoretic interpretation.

#### IV. CLASSIFICATION FRAMEWORKS

We now present a brief overview of the specific classification frameworks considered in this work: SVMs, LogitBoost classifiers and GPCs. We focus on properties pertinent to introspection. Specifically, we describe the mechanism by which parameters are learned and how probabilistic output is obtained. For simplicity, but without loss of generality, this work considers predominantly binary classification such that  $\mathcal{C} = \{C_1, C_2\}$ . For consistency we adhere to notation commonly found in the literature where a discriminant function

is often denoted as  $f(\cdot)$ . We note that this is equivalent to a particular model  $m$  as described in the previous section.

#### A. Support Vector Classification

SVM classification is well established in robotics so that we provide here only an overview<sup>1</sup>. SVMs are based on a linear discriminant framework which aims to maximise the margin between two classes. The model parameters are found by solving a convex optimisation problem, thereby guaranteeing the final classifier to be the best feasible discriminant given the training data. Once training is complete, predictions on future observations are made based on the signed distance of the observed feature vector from the optimal hyperplane, such that

$$f(\mathbf{x}_*) = \sum_{i=1}^N \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_*) + b, \quad (5)$$

where  $N$  is the size of the training set,  $\alpha_i$  refers to a Lagrange multiplier associated with datum  $i$ ,  $b$  denotes a bias parameter and  $k(\mathbf{x}_i, \mathbf{x}_j)$  denotes the kernel function. Both  $\alpha_i$  and  $b$  are obtained by training, and  $\alpha_i$  is then non-zero only for *support vectors*  $\mathbf{x}_i$ . The kernel function amounts to a scalar product between two data, which have been transformed from  $d$ -dimensional feature space into some higher dimensional space. The nature of this mapping between spaces is inherent in the choice of kernel and need not be specified explicitly (the kernel trick). The regularization and kernel parameters are learnt using cross-validation. We discuss our choices of kernel functions in Section IV-D.

In its original form, the SVM classifier output is an uncalibrated real value. A common means of obtaining a probabilistic interpretation is Platt's method [21]. Here, using a hold-out set not used for classifier training, a parametric sigmoid model is fit directly to the class posterior

<sup>1</sup>For a detailed account the reader is referred to, for example, [20].



$p(y_* = C_1 | f(\mathbf{x}_*))$ , such that

$$p(y_* = C_1 | f(\mathbf{x}_*)) = \frac{1}{1 + \exp(af(\mathbf{x}_*) + b)}. \quad (6)$$

The sigmoid parameters  $a$  and  $b$  are chosen via cross-validation using a model-trust optimisation procedure. Note that class likelihoods are derived here using only a *single* estimate of the discriminative boundary obtained from the classifier training procedure. No other feasible solutions are considered. Hence, the predictive variance of the discriminant  $f(\mathbf{x})$  is not taken into account while determining probabilistic output [22]. Further, no guarantees exist that the optimisation itself is well-behaved<sup>2</sup>.

### B. LogitBoosting Classifiers

Boosting is a widely used classification framework which involves training an ensemble of weak learners in sequence. The error function used to train a particular weak learner depends on the performance of the previous models [8]. Each weak learner,  $h(\mathbf{x})$  is trained using a weighted form of the dataset in which the data weights depend on the performance of the previous classifiers. Predictions from a boosted classifier are obtained using a weighted combination of the individual weak learner outputs such that

$$\text{sgn}(f(\mathbf{x}_*)) = \text{sgn}\left(\sum_{i=1}^M w_i h_i(\mathbf{x}_*)\right), \quad (7)$$

where  $M$  is the number of weak learners used.

LogitBoost [24] is a popular choice for a boosting-based classifier as it directly outputs class probability estimates. Weak learners are often chosen to be decision trees and training is conducted by fitting additive logistic regression models by stage-wise optimisation (using Newton steps) of the Bernoulli log-likelihood. The algorithm works in the logistic framework and yields a predictor function  $f(\mathbf{x})$  learnt from iterative hypothesis training. Cross-validation is used to set hyper-parameters like the learning rate, tree depth, and the number of boosting rounds. The class-conditional probabilities are obtained from the predictor function as

$$p(y_* = C_1 | \mathbf{x}_*) = \frac{\exp(f(\mathbf{x}_*))}{\exp(f(\mathbf{x}_*)) + \exp(-f(\mathbf{x}_*))}. \quad (8)$$

The procedure possesses asymptotic optimality as a maximum likelihood predictor [24], [25]. However, the method of converting the output of the predictor function to class-conditional probabilities is not fully probabilistic and does not account for variance in the underlying predictor function<sup>3</sup>.

### C. Gaussian Process Classification

Binary classification using a Gaussian Process (GP) [22], [26] is formulated by first introducing a *latent* function  $f(\mathbf{x})$  and then applying a logistic function  $\sigma$  to obtain the prediction  $p(y_* = C_1 | \mathbf{x}_*) = \sigma(f(\mathbf{x}_*))$ . A GP prior

<sup>2</sup>Throughout this work we use LIBSVM [23] for SVM training, calibration and testing.

<sup>3</sup>Throughout this work we use the Matlab implementation of LogitBoost for classifier training and testing.

is placed on the latent function  $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$  characterized by a *mean* function  $\mu(\mathbf{x})$  and a *covariance* (or kernel) function  $k(\mathbf{x}, \mathbf{x}')$ . GPC training establishes values for the hyper-parameters specifying the kernel function  $k$  by maximising the log marginal likelihood of the training data.

Probabilistic predictions for a test point are obtained in two steps. First, the distribution over the latent variable corresponding to the test input is obtained using Equation (9). Here,  $p(f | \mathcal{X}, \mathcal{Y}) = p(\mathcal{Y} | f)p(f | \mathcal{X})/p(\mathcal{Y} | \mathcal{X})$  is the posterior distribution over latent variables.

$$p(f_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int p(f_* | \mathcal{X}, \mathbf{x}_*, f)p(f | \mathcal{X}, \mathcal{Y})df. \quad (9)$$

This is followed by *marginalising* over the latent  $f_*$  to yield the class likelihood  $p(y_* = C_1 | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$  as

$$p(y_* = C_1 | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*) = \int \sigma(f_*)p(f_* | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)df_*. \quad (10)$$

Exact inference is analytically intractable due to the non-Gaussian logistic likelihood function. Instead we leverage expectation propagation (EP) [27], a method widely used for this purpose.

The GPC framework offers two key benefits over the other approaches discussed here [22]. Firstly, the classification output has a clear probabilistic interpretation as it directly results in the class likelihood. In contrast, neither the SVM nor the Boosting framework provide inherently probabilistic output but instead estimate a suitable calibration. Secondly, and crucially, the GP formulation addresses uncertainty or *predictive variance* in the latent function  $f(\mathbf{x})$  via *marginalisation* (or averaging) over all models induced by the training set resulting in the estimate  $p(y_* = C_1 | \mathcal{X}, \mathcal{Y}, \mathbf{x}_*)$  from Equation (10)<sup>4</sup>. Again this is in contrast to the SVM or Boosting estimate  $p(y = C_i | \hat{f}, \mathbf{x}_*)$  that rely on a single discriminant estimate  $\hat{f}: \mathcal{X} \rightarrow \mathcal{Y}$  learnt via minimisation. In the context of introspection, the ability to account for predictive variance is a key advantage of generative classification approaches<sup>5</sup>.

### D. Kernel Types

Evaluation of the discriminant function for an SVM and the covariance matrix for GPC inference both require the specification of a kernel function,  $k(\cdot, \cdot)$ . A rich body of literature exists on different choices of kernels for both frameworks. However, since our focus here is on a like-for-like comparison of different classification frameworks we choose two representative kernels rather than performing exhaustive model selection to optimise performance for a particular application. Firstly, as an example of the simplest kernel function available, we consider the linear kernel defined as

$$k_{LIN}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + c, \quad (11)$$

where  $c$  is a constant real number. The linear kernel is an apt choice where a linear separation of the data in feature space leads to adequate performance or where computational

<sup>4</sup>This process also gives rise to the well known property of increased variance while far away from the data in GP regression.

<sup>5</sup>Throughout this work we use the GPML toolbox [28] for GPC training and testing.

efficiency is of the essence. Often, however, a more sophisticated, non-linear kernel is required. In this category we use the *squared exponential* (SE) function as a canonical representative. The SE kernel with length scale parameter  $l$  is defined as

$$k_{SE}(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2l^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right). \quad (12)$$

In the context of an SVM, the SE function is more commonly known as a *radial basis function* (RBF). In the following we will adhere to convention and refer to SE GPCs and RBF SVMs.

## V. EXPERIMENTAL RESULTS

Our experiments investigate the introspective capacity of the classifiers introduced in Section IV in an autonomous driving setting. Specifically, we focus on the *classification* of road signs and the *detection* of traffic lights. In investigating both classification and detection we aim to address the full spectrum of applications commonly encountered in robotics. The two are distinct in that classification addresses the case where a decision is made between two, well-defined classes (e.g. two types of traffic signs) and investigates classifier performance as a third, previously unseen class is presented. The detection case is arguably the more common one in semantic mapping where a single class is separated from a broad (in terms of intra-class variation) *background* class. Here, the concept of a previously unseen class does not exist but the inherent assumption is that the data representing the background class are sufficiently representative to capture any non-class object likely to be encountered. In practice, this is often not the case, leading to a significant number of misclassifications. While it could be argued that this problem is ameliorated somewhat by expanding the dataset used for training, we propose that the complexity of the workspaces encountered during persistent, long-term autonomy will keep perplexing even the most rigorously trained classifier.

A rich body of work on the detection and classification of road signs and traffic lights has established a successful track record of template-based features for this purpose. Specifically, we leverage the approach proposed by Torralba *et al.* [29] in which a dictionary of partial templates is constructed, against which test instances are matched. A single feature consists of an image patch (ranging in size from  $8 \times 8$  to  $14 \times 14$  pixels) and its location within the object as indicated by a binary mask ( $32 \times 32$  pixels). For any given test instance, the normalised cross-correlation is computed for each feature in the dictionary. Therefore, per instance (or window, in the detection case) a feature vector of length  $d$  is obtained, where  $d$  is the size of the dictionary. We found empirically that  $d > 200$  leads to negligible performance increase in classification. Throughout our experiments we therefore set  $d = 200$ .

### A. Introspection in Classification

This section investigates classification output when a third, previously unseen class is presented to the classifier. As examples of classes typically encountered in autonomous driving applications we use a subset of the German Traffic

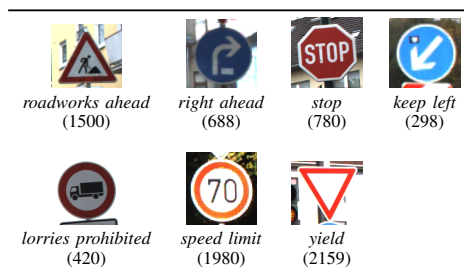


TABLE I: The seven classes of the German Traffic Sign Recognition Benchmark dataset considered in our work. The numbers in brackets indicate the number of data available per class.

Classifier	Precision	Recall	$F_1$
SE GPC	1.000	0.990	0.995
RBF SVM	1.000	0.995	0.997
Linear GPC	1.000	0.990	0.995
Linear SVM	1.000	0.990	0.995
LogitBoost	1.000	0.965	0.982

TABLE II: Classification performance when separating *stop* sign from the *lorries prohibited* signs. Note that different class combinations were found to yield classifiers of similar quality.

Sign Benchmark (GTSRB) dataset [30], which comprises over 50,000 loosely-cropped images of 42 classes of road signs, with associated bounding boxes and class labels. From this dataset we specifically focus on the seven classes shown in Table I. We arbitrarily select two classes for training: *stop* and *lorries prohibited*. To investigate the efficacy of the features used and training procedures employed, classifiers were trained separating these two classes using a balanced training set of 400 data (200 per class) and applying a canonical training procedure for each classifier type, including five-fold cross-validation where appropriate. Classifier performance was evaluated using standard metrics on a hold-out set of another 400 class instances (200 of each class) of the same two classes. The results are shown in Table II. Classification performance is commensurate across all classifiers. The corresponding precision-recall curve confirms the near-perfect separation of the classes and has been omitted here as it is otherwise uninformative. The classifiers are next retrained using the full 800 training data (400 per class) and the same canonical training procedures. They are then applied to 500 instances of the previously unseen class *roadworks ahead*. The resulting normalised entropy histograms are shown in Figure 3. The mean normalised entropies for the GPC-based classifiers are significantly higher than those of the other classification frameworks, indicating that the the GPC-based classifiers exhibit greater uncertainty in their judgement. Conversely, the RBF SVM and the LogitBoost classifier are extremely confident in their classifications with a very narrow distribution around a relatively low value of normalised entropy. This was an effect consistently observed throughout our experiments, which we attribute to the relatively gradual decay of the estimated class posterior probabilities through feature space often encountered far away from the decision boundary. Features from an unseen class which are located in feature space at a distance from the decision boundary

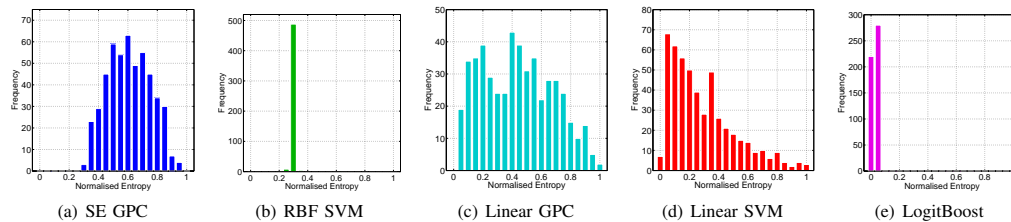


Fig. 3: Normalised entropy histograms of the marginal probabilities for five classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on 500 instances of the unseen class *roadworks ahead*. Higher normalised entropy implies more uncertainty in classifier output. Note that the mean normalised entropy for the SE GPC is higher than that of the others.

Test Class	Classifier	Normalised Entropy	
		$\mu \pm \text{std. err.}$	$\sigma \pm \text{std. err.}$
	SE GPC	<b>0.504</b> $\pm$ <b>1.92E-03</b>	0.110 $\pm$ 9.35E-05
	RBF SVM	0.313 $\pm$ 1.33E-04	0.012 $\pm$ 2.12E-06
	Lin GPC	0.245 $\pm$ 9.34E-04	0.173 $\pm$ 9.19E-05
	Lin SVM	0.106 $\pm$ 4.77E-04	0.107 $\pm$ 2.51E-04
	Logit	0.015 $\pm$ 2.72E-05	0.009 $\pm$ 4.11E-05
		SE GPC	<b>0.487</b> $\pm$ <b>1.70E-03</b>
RBF SVM		0.310 $\pm$ 1.13E-04	0.017 $\pm$ 3.72E-06
Lin GPC		0.286 $\pm$ 8.09E-04	0.179 $\pm$ 6.24E-05
Lin SVM		0.076 $\pm$ 3.72E-04	0.097 $\pm$ 2.43E-04
Logit		0.012 $\pm$ 1.79E-05	0.007 $\pm$ 1.27E-05
		SE GPC	<b>0.723</b> $\pm$ <b>4.91E-04</b>
	RBF SVM	0.306 $\pm$ 1.03E-04	0.095 $\pm$ 7.96E-05
	Lin GPC	0.680 $\pm$ 4.75E-04	0.235 $\pm$ 1.11E-04
	Lin SVM	0.634 $\pm$ 7.29E-04	0.267 $\pm$ 4.28E-05
	Logit	0.021 $\pm$ 1.07E-04	0.031 $\pm$ 7.10E-04
		SE GPC	0.804 $\pm$ 6.08E-04
RBF SVM		0.335 $\pm$ 1.43E-04	0.050 $\pm$ 1.26E-05
Lin GPC		<b>0.811</b> $\pm$ <b>4.39E-04</b>	0.184 $\pm$ 1.66E-04
Lin SVM		0.642 $\pm$ 3.24E-04	0.294 $\pm$ 9.19E-05
Logit		0.017 $\pm$ 3.62E-05	0.018 $\pm$ 2.06E-04
		SE GPC	<b>0.259</b> $\pm$ <b>2.36E-03</b>
	RBF SVM	0.255 $\pm$ 1.28E-04	0.027 $\pm$ 5.26E-06
	Lin GPC	0.155 $\pm$ 9.27E-04	0.140 $\pm$ 2.61E-04
	Lin SVM	0.043 $\pm$ 7.82E-05	0.059 $\pm$ 1.26E-04
	Logit	0.007 $\pm$ 1.29E-07	0.007 $\pm$ 2.31E-05

TABLE III: Mean and standard deviation normalised entropies (including standard errors) from ten iterations of classifier training and testing, each with a randomly created dictionary and both training and test datasets resampled. Results are presented for classifiers trained on the road sign classes *stop* and *lorries prohibited* and tested on five different unseen classes as shown.

therefore only span a very narrow range of estimated class posterior probabilities.

In order to mitigate any influences of the specific feature set used and the specific training and test data selected we repeated the above experiment across a number of random dictionaries, data samples and unseen classes. Specifically, for each of five different unseen classes, we perform ten iterations of classifier training and testing with a random dictionary and training and test datasets resampled for each run. The results, presented in Table III, are consistent with those in Figure 3 in that the GPCs tend to be more uncertain while SVM and LogitBoost are more confident with an often significantly narrower distribution of normalised entropy values. The results in Table III indicate that the gap in uncertainty between the different frameworks is more pronounced for some unseen classes than for others. We attribute this to the varying degree of similarity in feature space between the unseen class and the classes in the training set. A more in-depth analysis of this phenomenon remains future work.

Classifier	Precision	Recall	$F_1$
SE GPC	0.976	0.909	0.941
RBF SVM	0.982	0.931	0.956
Linear GPC	0.970	0.912	0.940
Linear SVM	0.979	0.929	0.953
LogitBoost	0.963	0.928	0.945

TABLE IV: Performance on a holdout set of 2000 instances of classifiers trained on data from the TLR data set.

### B. Introspection in Detection

We investigate the same classification frameworks as before on the task of traffic light detection. To this end we use the Traffic Lights Recognition (TLR) dataset [31], which is a sequence of colour images taken by a monocular camera from a car driving through central Paris. The TLR dataset comprises of just over 11,000 frames, where most of the traffic lights have been labeled with bounding boxes and further metadata such as the status of the signal or whether a particular label is ambiguous (e.g. the image suffers from motion blur, the scale is inappropriate or a traffic light is facing the wrong way). A few traffic lights have been omitted altogether. As suggested by the authors of [31], we exclude from our experiments any labels of class *ambiguous* or *yellow signal* and any instances which are partially occluded. We also remove any section of the sequence where the car is stationary and the lights are not changing. We split the dataset into two parts (at frame 7,200 of 11,178), with an approximately equal number of remaining labels in each part and with no physical traffic lights in common. Positive data are extracted as labeled. Negative *background* data are extracted by sampling patches of random size and position from scenes in the dataset while ensuring that the patches do not overlap with positive instances. The data are then split into training and test sets and classifiers are trained as before.

Again, we first verify the efficacy of the features selected and the training procedures employed. Table IV shows the classification performance for classifiers trained on 1,000 examples and evaluated on a hold-out set of 2,000 data. For completeness, Figure 4 shows the corresponding precision-recall curve. As before, classification performance according to conventional metrics is commensurate across all frameworks. In Figure 5, however, we demonstrate how the lack of introspection can impact classification performance when accept/reject decisions are guided by classification confidence. Specifically, we show the *cumulative* effect of accepting classifications below a given uncertainty threshold. First we note that when classifications are accepted at any level of uncertainty (i.e. up to and including unity normalised

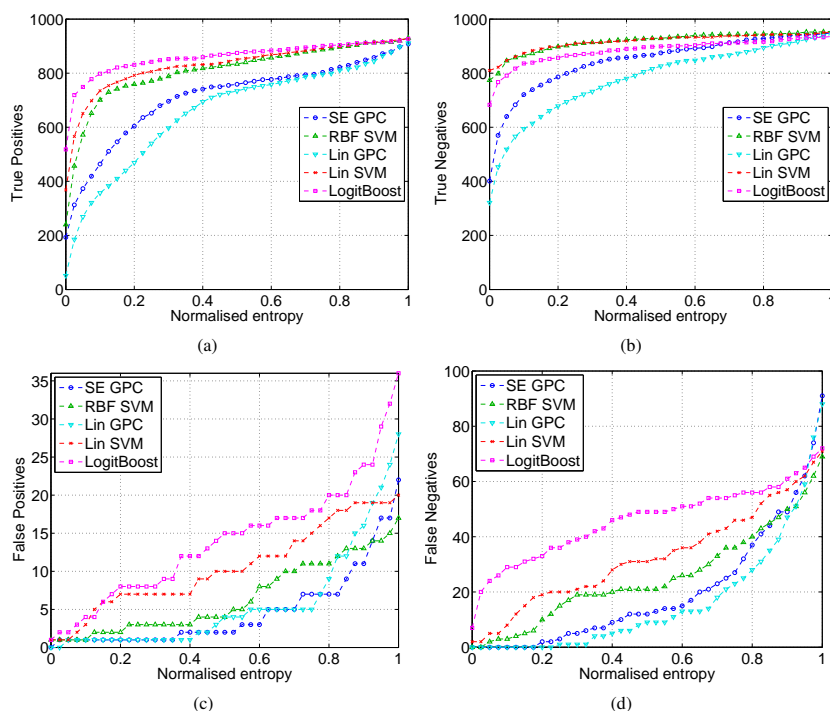


Fig. 5: Cumulative frequency plots of classification confusion (true positives, true negatives, false positives, and false negatives) against normalised entropy. The classifiers have been trained on 500 traffic lights against 500 background patches, and tested on 1,000 instances of each. Note that lower normalised entropy implies more certainty in classification. A more introspective classifier is one that exhibits higher uncertainty (as witnessed by larger normalised entropy in its output) when processing difficult instances. Consequently, class decisions on output above a given normalised entropy threshold are deferred since the output is deemed ambiguous. This is desirable since a single bad decision can have disastrous consequences. (Best viewed in colour.)

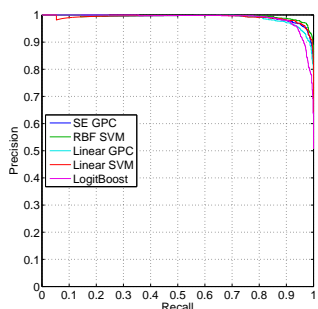


Fig. 4: Precision-recall graph for traffic light detection. Classifier performance is commensurate for all frameworks. (Best viewed in colour.)

entropy) all classification frameworks are commensurate in terms of true positives and true negatives (top row of Figure 5). This further corroborates the accuracy figures in Table IV. However, true positive and negative classifications occur generally at higher certainty (i.e. as normalised entropy tends to zero) for SVMs and LogitBoost classifiers than for the GPC variants. The latter are overall less certain about a significant number of correct classifications. The bottom row of Figure 5 indicates that SVMs and LogitBoost classifiers are also significantly more confident when *misclassifying* data (an example of this is also shown in Fig. 1). Significant numbers of mistakes are made at relatively low normalised entropy thresholds. The GPC variants, in contrast, accu-

rate comparable numbers of classification errors only at higher normalised entropy thresholds. The price paid for this more realistic assessment of the classification output is a reduction in correct classifications above the normalised entropy threshold. Note that this does not mean that subsequent samples are misclassified. It only implies that some other remedial action might be taken — for example obtaining label confirmation from a human or gathering otherwise additional data to aid disambiguation.

## VI. CONCLUSIONS

This work demonstrates how performance metrics traditionally used in machine learning for classifier training and evaluation may be insufficient to characterise system performance in a robotics context, where a single misjudgement can have disastrous consequences. To remedy this shortcoming, we propose the concept of *introspection*: the ability to mitigate potentially overconfident classifications by a realistic assessment of predictive variance. Our experimental results imply that, despite commensurate performance as measured by more conventional metrics, GPCs possess a more pronounced introspective capacity than other classification frameworks commonly employed in robotics. We attribute this to their accounting, at test time, for predictive variance over the space of feasible classification models. This is in contrast to other commonly employed classification frameworks which often only consider a one-shot (ML or

MAP) solution. GPCs appear therefore better suited than the other frameworks investigated to applications where a realistic assessment of classification accuracy is required. Crucially, this includes many decision-making problems commonly encountered in robotics.

We have not, at this stage, considered the computational complexity of the approaches presented. Though GPCs in their basic form are notoriously expensive, more elaborate schemes exist which reduce the computational burden required for GPC inference. Our future work will investigate a variety of these schemes for suitability for real-time performance in autonomous driving tasks. Our work also holds implications for robotic active learning and exploration, which opens up additional avenues of research we intend to explore.

#### VII. ACKNOWLEDGEMENTS

This work is funded under the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916 and by the UK EPSRC Grant Number EP/J012017/1. We gratefully acknowledge advice from Ian Baldwin on parameter selection for boosting trees in Matlab. We thank Prof. Paul Newman for his support and encouragement for this work.

#### REFERENCES

- [1] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Y. Ng, "Discriminative Learning of Markov Random Fields for Segmentation of 3D Scan Data." in *CVPR (2)*, 2005, pp. 169–176.
- [2] O. Martínez-Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Robot. Auton. Syst.*, vol. 55, no. 5, pp. 391–402, 2007.
- [3] I. Posner, M. Cummins, and P. Newman, "A generative framework for fast urban labeling using spatial and temporal context," *Autonomous Robots*, 2009. [Online]. Available: <http://dx.doi.org/10.1007/s10514-009-9110-6>
- [4] B. Douillard, D. Fox, and F. Ramos, "Laser and vision based outdoor object mapping," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.
- [5] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Intl. Conf. on Robotics and Automation*, 2012, pp. 3515–3522.
- [6] S. Sengupta, P. Sturgess, L. Ladick, and P. H. S. Torr, "Automatic dense visual semantic mapping from street-level imagery," in *Robotics and Autonomous Systems, IEEE International Conference on*, 2012.
- [7] R. Paul, R. Triebel, D. Rus, and P. Newman, "Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification," in *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, 2012, to appear.
- [8] C. Bishop, *Pattern recognition and machine learning*. springer New York, 2006, vol. 4.
- [9] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [10] A. Huang and S. Teller, "Probabilistic Lane Estimation using Basis Curves," in *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010.
- [11] D. Meger, P. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. Little, and D. Lowe, "Curious george: An attentive semantic robot," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 503–511, 2008.
- [12] J. Velez, G. Hemann, A. Huang, I. Posner, and N. Roy, "Planning to perceive: Exploiting mobility for robust object detection," in *Proc. ICAPS*, 2011.
- [13] S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy, "Toward information theoretic human-robot dialog," in *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.
- [14] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian processes for object categorization," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 169–188, 2010.
- [15] A. Joshi, F. Porikli, and N. Papanikolopoulos, "Multi-class active learning for image classification," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 2372–2379.
- [16] A. Holub, P. Perona, and M. Burl, "Entropy-based active learning for object recognition," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–8.
- [17] T. Hospedales, S. Gong, and T. Xiang, "Finding rare classes: Active learning with generative and discriminative models," *Knowledge and Data Engineering, IEEE Transactions on*, no. 99, pp. 1–1, 2011.
- [18] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *The Journal of Machine Learning Research*, vol. 2, pp. 45–66, 2002.
- [19] W. Zhang, X. Stella, and Y. S. Teng, "Power svm: Generalization with exemplar classification uncertainty," in *Computer Vision and Pattern Recognition, 2012. CVPR 2012. IEEE Conference on*. IEEE, 2012.
- [20] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
- [21] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances In Large Margin Classifiers*. MIT Press, 1999, pp. 61–74.
- [22] C. Rasmussen and C. Williams, "Gaussian processes for machine learning. 2006," *The MIT Press, Cambridge, MA, USA*.
- [23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [24] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, p. 2000, 1998.
- [25] T. Hastie and R. Tibshirani, *Generalized additive models*. Chapman & Hall/CRC, 1990.
- [26] C. Williams and D. Barber, "Bayesian classification with gaussian processes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 12, pp. 1342–1351, 1998.
- [27] T. Minka, "Expectation propagation for approximate bayesian inference," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 362–369.
- [28] C. E. Rasmussen and H. Nickisch, "Gaussian processes for machine learning (GPML) toolbox," *J. Mach. Learn. Res.*, vol. 11, pp. 3011–3015, Dec. 2010. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1756006.1953029>
- [29] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 854–869, May 2007.
- [30] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, no. 0, pp. –, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0893608012000457>
- [31] R. C. of Mines ParisTech, "Traffic lights recognition (TLR) data set," <http://www.lara.prd.fr/benchmarks/trafflightsrecognition>.

# Confidence Boosting: Improving the Introspectiveness of a Boosted Classifier for Efficient Learning

Rudolph Triebel Hugo Grimmett Ingmar Posner

Mobile Robotics Group, Dep. of Engineering Science, Univ. of Oxford, OX1 3PJ Oxford, UK  
{rudi, hugo, ingmar}@robots.ox.ac.uk

**Abstract**—This paper concerns the recently introduced notion of introspective classification. We introduce a variant of the point-biserial correlation coefficient (PBCC) as a measure to characterise the introspective capacity of a classifier and apply it to investigate further the introspective capacity of boosting – a well established, efficient machine learning framework commonly used in robotics. While recent evidence suggests that boosting is prone to providing overconfident classification output (i.e. it has a low introspective capacity), we investigate whether optimising this criterion directly leads to an improved introspective capacity. We show that with only a slight modification in the AdaBoost algorithm the resulting classifier becomes less confident when making incorrect predictions, rendering it significantly more useful when it comes to efficient robot decision making.

## I. INTRODUCTION

Machine learning algorithms are changing the face of mobile robotics. Their reach now extends to a significant number of robotic tasks including perception, planning, navigation, and manipulation. As a result, most modern mobile robotic systems already rely to a significant degree on machine learning methods. However, for these methods to be useful in robotics, some very specific requirements have to be beyond those commonly considered in other research areas. In particular, these requirements include efficiency in terms of memory requirements, computation time and energy consumption as well as plasticity and robustness in order to provide true long-term autonomy. Algorithms that particularly meet this latter criterion are often identified as *online* or *life-long* learning methods.

In this work, we focus on *boosting*, a machine learning framework commonly used in robotics both for its efficiency and often competitive classification performance. Here we investigate its usefulness in *mission-critical* applications, where a single error in the classification can have disastrous consequences for the entire mission. As an example, consider an autonomous car that fails to detect a red traffic light. As was shown by Grimmett *et al.* [1], these applications require, in addition to a low rate of false detections (especially false negatives), a classifier that is able to provide a realistic estimate of its classification *confidence* along with the predicted class label. Classifiers with this capability are denoted as *introspective*. In this work, we investigate this further and particularly address two main questions. Firstly, what could be a good measure of this relationship between confidence and correctness? And secondly, can we use such

a measure to improve the introspective capabilities of one of the most prevalent classification algorithms in robotics? The preliminary findings we present here suggest that while boosting may not be as intrinsically introspective as, for example, a Gaussian Process Classifier (GPC) [1], its introspective capacity can be increased significantly by optimising it explicitly as part of the standard Boosting framework. We point out that we do not provide any theoretical proofs here, but instead present empirical results along the lines of those presented in [1].

Our modification specifically applies to the standard AdaBoost [2] algorithm. However, our findings are also likely to be replicable for other variants of boosting such as LogitBoost, GentleBoost [3], or robust variants [4].

### A. Related Work

Apart from the seminal work on boosting in general [2]–[4], most of the references related to this work are already given in [1]. Boosting has a long and successful track record in mobile robotics (see, for example, the work of Martinez Mozos *et al.* [5], [6]). Introspection has been recently introduced by Grimmett *et al.* [1].

This paper further investigates the introspective capabilities of one particular classification framework.

## II. APPROACH

In this section we first describe the standard binary AdaBoost [2] algorithm (see Algorithm 1). Then, we introduce Confidence Boosting, our new introspective variant of AdaBoost. It uses an empirical measure of “introspectiveness”, which is a new idea to quantify and assess this property in classifiers. We propose a variant of the point-biserial correlation coefficient (PBCC) for this measure, which we briefly explain.

### A. Standard Boosting

The main principle of boosting is to assign weights to the  $n$  training data points  $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  with  $\mathbf{x}_i \in \mathbb{R}^d$  and to run a given number  $m$  of training rounds through the data, where at each training round a classifier is obtained that particularly focusses on the misclassified samples from the previous rounds. This is done by updating the data weights according to a classification loss function, which is usually the 01-loss. In more detail the steps are: first, the weights are all equally initialized with  $1/n$ . Then, in each round a

weak classifier  $f_i : \mathbb{R} \rightarrow \{-1, 1\}$  is learned from the weighted training data, and its training error  $\epsilon_i$  is computed. Here,  $I()$  denotes the identity function, which is 1 if the argument is true and 0 otherwise. From the training error the coefficient  $\alpha_i$  is computed and the data weights are updated so that the misclassified points obtain a higher weight while the weights of the other points remain unchanged. The obtained coefficients  $\alpha_i$  are then used to classify a new test datum  $\mathbf{x}^*$  using the weighted sum  $\sum_{i=1}^m \alpha_i f_i(\mathbf{x}^*)$ , which is simply tested for its sign: if it is positive, the predicted class label is 1, otherwise it is -1.

---

**Algorithm 1:** AdaBoost for binary classification

---

**Data:** training data  $(\mathcal{X}, \mathbf{y})$  consisting of  $n$  labeled feature vectors, where  $y_j \in \{-1, 1\}$   
**Input:** Number  $m$  of training rounds  
**Output:** coefficients  $(\alpha_1, \dots, \alpha_m)$

```

1  $\mathbf{w}^{(1)} \leftarrow (1/n, \dots, 1/n)$ 
2 for  $i \leftarrow 1$  to  $m$  do
3    $f_i \leftarrow \text{LearnWeakClassifier}(\mathbf{w}, \mathcal{X}, \mathbf{y})$ 
4    $\epsilon_i \leftarrow \frac{\sum_{j=1}^n w_j^{(i)} I(f_i(\mathbf{x}_j) \neq y_j)}{\sum_{j=1}^n w_j^{(i)}}$ 
5    $\alpha_i \leftarrow \ln\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$ 
6   for  $j \leftarrow 1$  to  $n$  do
7      $w_j^{(i+1)} \leftarrow w_j^{(i)} \exp(\alpha_i I(f_i(\mathbf{x}_j) \neq y_j))$ 
8   end
9 end

```

---

### B. Confidence Boosting

The main benefits of the boosting algorithm are its arbitrarily small training error (it decreases monotonically with the number of training rounds), and its very efficient inference step. However, as was shown in [1], in terms of introspective the standard boosting algorithm performs much worse than other classification algorithms such as the Gaussian Process classifier (GPC), which means that it tends to be overconfident in its class predictions. This can for example be seen when the algorithm is first trained on two classes, and then elements of a third, unseen class are presented in the classification step. In that case, the algorithm returns class predictions with a very high certainty, although all predicted class labels must be incorrect. As a consequence, standard boosting can not be used in situations where the class label uncertainty is needed for further processing, e.g. to detect potential misclassifications or for active learning. In the experimental section, we will give more evidence for this.

To address this issue, we first have a closer look at line 3 in Algorithm 1: Here, a weak classifier  $f_i$  is determined that assigns class labels to given input data. The only requirement for  $f_i$  to be a weak classifier is that the weighted training error  $\epsilon_i$  computed in line 4 is not larger than 0.5. One simple example for a weak classifier, which is often used in boosting, is the *decision stump*, which operates on a projection of the data onto a single feature dimension  $k$  and determines a threshold  $\theta$  and an orientation  $s \in \{-1, 1\}$  so

that most of the positively labeled training points are on the positive side of the resulting decision boundary, i.e.

$$|\{(x_j^k, y) \forall j = 1, \dots, n \mid s(x_j^k - \theta) \geq 0\}| \geq \frac{n}{2}, \quad (1)$$

where  $k$  is a fixed dimension of the feature vector  $\mathbf{x}_j$ , and  $|\cdot|$  denotes the size of a set. To find a weak classifier  $f_i$ , one common method is to loop over all feature dimensions  $k = 1, \dots, d$  and to use the decision stump that provides the smallest weighted training error, i.e.  $\epsilon_i$  is then the smallest over all dimensions. The benefit of this is that the number of training points that need re-weighting is smallest and that the overall training error decreases fast. However, for an introspective classifier, one is more interested in a realistic relation between a correct classification and one with low uncertainty, rather than in a low training error. In the next section, we will give more details how this “introspectiveness” relation can be formalized. For now, we just state that all decision stumps can be used as a weak classifier, because they all return a weighted training error less than 0.5. Thus, if we choose the one that is most introspective in a given sense, instead of the one with the smallest training error, then the strong classifier that results from boosting is more likely to be introspective, too. This is the main idea of confidence boosting.

Of course, this brings also some drawback: as we don’t choose the optimal decision stump in terms of classification performance, the resulting strong classifier will usually also perform worse. However, this can be addressed by simply increasing the number of used decision stumps, because the training error still decreases in each training round, although at a slower rate. Thus, the aim of obtaining an introspective classifier is traded off with the need to reduce the training error. This suggests a weighted sum of classification performance and introspectiveness as an assessment method for decision stumps, however in this first version of the algorithm we only consider the introspective part to avoid introducing a parameter for the algorithm. To measure introspectiveness, we suggest a function similar to the point-biserial correlation coefficient, which is described next.

### C. The Point-Biserial Correlation Coefficient

In many probabilistic reasoning applications the problem arises how to measure the relation between two random variables, and there are a number of different measures in the literature that can provide such a relation. In principle, these measures try to answer questions like: “how strong is the statistical dependence between the variables?”, or “how much information does one variable give about the other?”, or “how much are the variables correlated?”. Examples of these measures are the Kullback-Leibler (KL-) divergence, the mutual information (MI), or the correlation coefficient. However, most of these measures require both random variables to be continuous, whereas in our case we want to relate the discrete, binary variable of “classification correctness” with the continuous variable “classification uncertainty”. The intuition behind this is that a classifier that is very often correct when it is certain and only incorrect when it is uncertain should be denoted as introspective. One way to

determine such a relation is by using the *Point-Biserial Correlation Coefficient* (PBCC), a variant of the standard correlation coefficient. The PBCC is defined as follows:

$$r_{pb} := \frac{\mu_1 - \mu_2}{\sigma_n} \sqrt{\frac{n_1 n_2}{n^2}}, \quad (2)$$

where  $\mu_1$  and  $\mu_2$  are the mean values of the continuous variable for those parts of the data, for which the binary variable is either 0 or 1, respectively. In our case, these are the average uncertainties for the incorrectly and the correctly classified data points. Furthermore,  $\sigma_n$  is the standard deviation of the continuous variable, i.e. the classification uncertainty, and  $n_1$  and  $n_2$  are the numbers of incorrectly and correctly classified samples with  $n = n_1 + n_2$ .

While the PBCC provides a good measure of introspectiveness in cases where there are enough correctly and incorrectly classified samples, it has the drawback that its range decreases when these numbers are very unbalanced, for example when there are no incorrectly classified samples. In that case, the PBCC is 0, although the correctly classified samples can all be very certain, in which case the classifier would be more introspective than the PBCC suggests. Therefore, in our experiments, we use a simpler version of the PBCC, which only considers the first term, i.e.:

$$r_{pb}^* := \frac{\mu_1 - \mu_2}{\sigma_n}. \quad (3)$$

In the following, we will refer to this measure as the simplified PBCC (sPBCC).

To summarize, the modified version of the function `LearnWeakClassifier` trains a decision stump for a projection of the training data onto each dimension  $k = 1, \dots, d$  and chooses the one that maximises either  $r_{pb}$  or  $r_{pb}^*$ . This requires a measure of uncertainty from a class label prediction returned from a decision stump, but these only return either 0 or 1, depending on whether the feature is on the positive or the negative side of the decision boundary. To obtain a probability value, we apply a sigmoid function to the distance of the feature value from the decision boundary of the stump, specifically we use the cumulative Gaussian function. As a result, we obtain a probability of a given test point  $\mathbf{x}_j$  to have label 1. From this probability we then compute the *normalized entropy* to obtain an uncertainty estimate for the predicted class label, as was already done in [1].

### III. EXPERIMENTAL RESULTS

The aim of our experiments is two-fold: first we see whether our simplified PBCC measure is consistent with our intuitive notion of introspection. Secondly, we show that ConfidenceBoost, our modified version of AdaBoost, leads to a more introspective classifier. Our feature selection is the same as presented in [1], namely a dictionary-based correlation of randomly chosen image patches (originally introduced by Torralba *et al.* [7]). We also use the GTSRB data set, which comprises images of road signs in urban environments.

In our first experiment, we compute the sPBCC value for the classifiers that were already used in [1]: the GPC and SVM, both with linear and squared-exponential (SE) kernels,

Classifier	Precision	Recall	Accuracy	sPBCC
SE GPC	1.000	1.000	1.000	-0.720
RBF SVM	1.000	1.000	1.000	-0.959
Linear GPC	1.000	1.000	1.000	-0.863
Linear SVM	1.000	1.000	1.000	-1.270
LogitBoost	1.000	1.000	1.000	-196189.581
ConfBoost	1.000	0.995	0.997	9.113

TABLE I: Classification performance when separating *stop* sign from the *lorries prohibited* signs. In addition to precision, recall, and accuracy, we also report the sPBCC value, which quantifies the introspective capabilities of a classifier. We can see that the GP classifiers are more introspective than the SVMs according to that measure, which underlines our findings from [1].

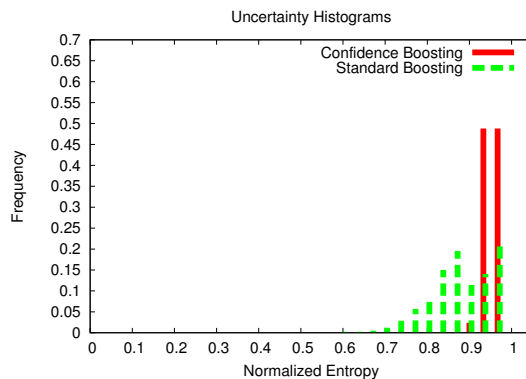


Fig. 1: Histogram of normalised entropies for the probabilities of the predicted class labels, where an unknown class was shown to the AdaBoost and the ConfidenceBoost classifier. Both classifiers are unconfident about the given class labels, but ConfidenceBoosting shows this to a larger extent. Note that the histograms are normalized so that the sum of all bins is 1.

LogitBoost, and our new ConfidenceBoost algorithm. We train each on 200 instances of *stop* and *lorry prohibited* sign, and test on an equally-sized set of the same classes. The results are shown in Table I.

We can see that both GPCs have a higher correlation between false classification and uncertainty than their SVM equivalents, as we would expect given their already established introspective capacity. We also see that the ConfBoost algorithm also performs very highly in this regard.

In the second experiment, we train both the standard AdaBoost and ConfidenceBoost on the same two classes of road sign, and test on a novel third class – we use the *roadworks ahead* sign – computing the histogram of normalised entropies for both. These histograms compare the distribution over the uncertainties in the class predictions of the classifiers, and can be seen in Fig. 1. They show that both classifiers are fairly uncertain about the predicted class labels, which is reasonable given that the presented data are from a class unseen during training. However, the labels returned from ConfidenceBoost are even more uncertain, leading to a more realistic assessment of the classification result. Hence, we can conclude that the ConfidenceBoost algorithm leads to a more introspective classifier.



### IV. CONCLUSIONS AND FUTURE WORK

The two contributions of this work are a way to quantify the introspectiveness of a classifier – a notion that only recently has been introduced, and a simple method to improve the introspective capabilities of the standard AdaBoost algorithm. However, many new questions arise from that, where one is that of a more theoretical foundation for the experimental results shown here. Another one addresses the implications of our findings in a more general way, such as: can this method be applied to other classification algorithms? For example, one could think of using other established methods for weak classification, thereby optimizing them in the introspective sense. Our preliminary results at least justify some further research along these lines.

### ACKNOWLEDGMENT

This work is funded under the European Community's Seventh Framework Programme (FP7/2007-2013 V-CHARGE) under Grant Agreement Number 269916 and by the UK EPSRC Grant Number EP/J012017/1.

### REFERENCES

- [1] H. Grimmer, R. Paul, R. Triebel, and I. Posner, "Knowing when we don't know: Introspective classification for mission-critical decision making," in *Proc. of the IEEE International Conference on Robotics & Automation (ICRA)*, 2013, to appear.
- [2] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, 1997.
- [3] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.
- [4] P. Li, "Robust logitboost and adaptive base class (abc) logitboost," in *Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010.
- [5] O. M. Mozos and W. Burgard, "Supervised learning of topological maps using semantic information extracted from range data," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Beijing, China, October 2006, pp. 2772–2777.
- [6] O. M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Robotics and Autonomous Systems (RAS)*, vol. 55, no. 5, pp. 391–402, May 2007.
- [7] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing visual features for multiclass and multiview object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 5, pp. 854–869, May 2007.

## Exploiting Repetitive Object Patterns for Model Compression and Completion

Luciano Spinello<sup>1,2</sup>, Rudolph Triebel<sup>2</sup>, Dizan Vasquez<sup>3</sup>,  
Kai O. Arras<sup>1</sup>, and Roland Siegwart<sup>2</sup>

<sup>1</sup> Social Robotics Lab, University of Freiburg, Germany

<sup>2</sup> Autonomous Systems Lab, ETH Zurich, Switzerland

<sup>3</sup> ITESM Campus Cuernavaca, Mexico

**Abstract.** Many man-made and natural structures consist of similar elements arranged in regular patterns. In this paper we present an unsupervised approach for discovering and reasoning on repetitive patterns of objects in a single image. We propose an unsupervised detection technique based on a voting scheme of image descriptors. We then introduce the concept of *latticelets*: minimal sets of arcs that generalize the connectivity of repetitive patterns. Latticelets are used for building polygonal cycles where the smallest cycles define the sought groups of repetitive elements. The proposed method can be used for pattern prediction and completion and high-level image compression. Conditional Random Fields are used as a formalism to predict the location of elements at places where they are partially occluded or detected with very low confidence. Model compression is achieved by extracting and efficiently representing the repetitive structures in the image. Our method has been tested on simulated and real data and the quantitative and qualitative result show the effectiveness of the approach.

### 1 Introduction

Man-made and natural environments frequently contain sets of similar basic elements that are arranged in regular patterns. Examples include architectural elements such as windows, pillars, arcs, or structures in urban environments such as equidistant trees, street lights, or similar houses built in a regular distance to each other. There are at least two applications where models of repetitive structures are useful pieces of information: occlusion handling and data compression. For the former, pattern information can be used to predict the shape and position of occluded or low confidence detections of objects in the same scene. This introduces a scheme in which low-level detections are mutually reinforced by high-level model information. For model compression, representing the repetitive structure by a generalized object and pattern description makes it possible to represent the structure of interest in the image very efficiently.

In this paper, we present a technique to find such repetitive patterns in an unsupervised fashion and to exploit this information for occlusion handling and compression. Specifically, we evaluate our method on the problem of building facade analysis.

The contributions of this paper are:

1. Unsupervised detection of mutually similar objects. Closed contours are extracted and robustly matched using a growing codebook approach inspired by the Implicit Shape Models (ISM) [1].

2 L. Spinello, R. Triebel, D. Vasquez, K. O. Arras, R. Siegwart

2. Analysis of pattern repetitions by the concept of *latticelets*: a selected set of frequent distances between elements of the same object category in the Cartesian plane. Latticelets are generalizations of the repetition pattern.
3. A probabilistic method to geometrically analyze cyclic element repetitions. Using Conditional Random Fields (CRF) [2], the method infers missing object occurrences in case of weak hypotheses. Element detection probability and geometrical neighborhood consistency are used as node and edge features.

Our method is a general procedure to discover and reason on repetitive patterns, not restricted to images. The only requirement is that a method for detecting similar objects in a scene is available and that a suitable latticelet parameterization is available in the space of interest, e.g. the image or Cartesian space.

To the authors' best knowledge, there is no other work in the literature that pursues the same goals addressing the problem in a principled way.

This paper is organized as follows: the next section discusses related work. Section 3 gives an overview of our technique while in Section 4, the process of element discovery is explained. Section 5 presents the way we analyze repetitive patterns and Section 6 describes how to use CRFs for the task of repetitive structure inference. Section 7 shows how to obtain an high-level image compression with the proposed method. In Section 8 the quantitative and qualitative experiments are presented followed by the conclusions in Section 9.

## 2 Related Work

In this work we specifically analyze repetitions from a single static image. The work of [3] uses Bayesian reasoning to model buildings by architectural primitives such as windows or doors parametrized by priors and assembled together like a 'Lego kit'. The work of [4] interprets facades by detecting windows with an ISM approach. A predefined training set is provided. Both works address the problem with a Markov Chain Monte Carlo (MCMC) technique. Unlike our approach, they do not exploit information on the connectivity between the detected elements. Our work uses ISM in an unsupervised fashion without a priori knowledge. We consider closed contours to create codebooks that generalize the appearance of repeated elements. Thereby, we are able to recognize such elements with high appearance variability thanks to the Hough-voting scheme. In the field of computer graphics, grammar based procedural modeling [5–7] has been formally introduced to describe a way of representing man-made buildings. Most of these works do not discover patterns but reconstruct the 3D appearance of the facade and require human intervention.

Approaches based on RANSAC [8] and the Hough transform [9] have been used to find regular, planar patterns. More sophisticated methods relax the assumption of the regular pattern using Near-Regular Textures (NRT) [10, 11]. Similar to our work is [12] in which the authors propose a method to find repeated patterns in a facade by using NRT with MCMC optimization using rules of intersection between elements. They are able to extract a single pattern based on a 4-connectivity lattice. Our approach allows detection of arbitrary patterns without relying on a fixed model. Further, it can detect multiple object categories and associate for each category multiple repetition patterns.

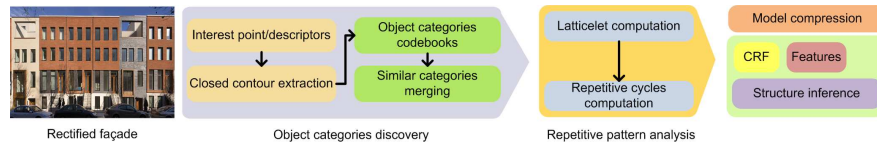


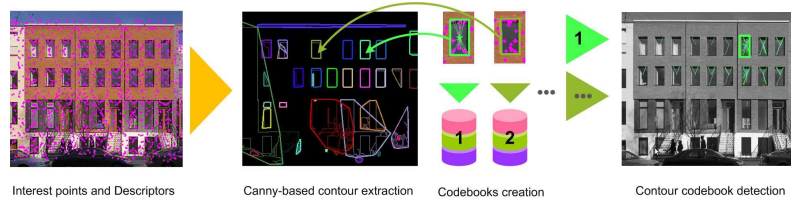
Fig. 1. Schematic overview of the algorithm.

### 3 Overview

The first step of our algorithm (see Fig. 1) is to compute a set of standard descriptors on a given input image. Then, we compute closed contours that represent the candidates for repetitive objects such as windows or pillars. The key idea is that we do not classify these objects using a model that was previously learned from training data, but instead, obtain evidence of their occurrence by extracting similarities directly from the given scene. The advantage of this is twofold: first, we are independent of a previously hand-labeled training data set. Second, by grouping similar objects into categories and considering only those categories with at least two object instances, we can filter out outlier categories for which no repetitive pattern can be found. Our measure of mutual similarity is based on the object detection approach by Leibe *et al.* [1]. In the next step, we analyze repetitive patterns inside each category. This is done by analyzing the Euclidean distances between elements in the image accumulated in a frequency map. These relative positions are represented as edges in a lattice graph in which nodes represent objects positions. The most dominant edges by which all nodes in this graph can be connected are found using a Minimum Spanning Tree algorithm and grouped into a set that we call latticelet. For reasoning on higher-level repetitions we extract a set of polygonal repetitions composed of latticelet arcs. Such polygonal repetitions are used to build a graph for predicting the position of occluded or weakly detected elements. An inference engine based on CRFs is used to determine if the occurrence of an object instance at a predicted position is likely or not. In an image compression application, we use a visual template of each object category, the medium background color and the lattice structure to efficiently store and retrieve a given input image.

### 4 Extraction of Mutually Similar Object Instances

In this section we explain the process of discovering repetitive elements present in an image based on closed contours. As first step of the algorithm, Shape Context descriptors [13] are computed at Hessian-Laplace interest points. Contours are computed by using the binary output of the Canny edge detector [14] encoded via Freeman chain code [15]. We refer to the content in each contour as an object instance  $O_e$ . Matching contours in real world images can be very hard due to shadows and low contrast areas. We therefore employ an Implicit Shape Model-like (ISM) technique in which the contours act as containers to define a codebook of included descriptors. This way, we can robustly match objects. In summary, an ISM consists of a set of local region descriptors, called *codebook*, and a set of displacements, usually named *votes*, for each descriptor.



**Fig. 2.** Extraction of mutually similar objects. For each closed contour, a codebook of descriptors is created that contains relative displacements to the object centers (votes). Then, the descriptors of each object are matched against the descriptors in the image.

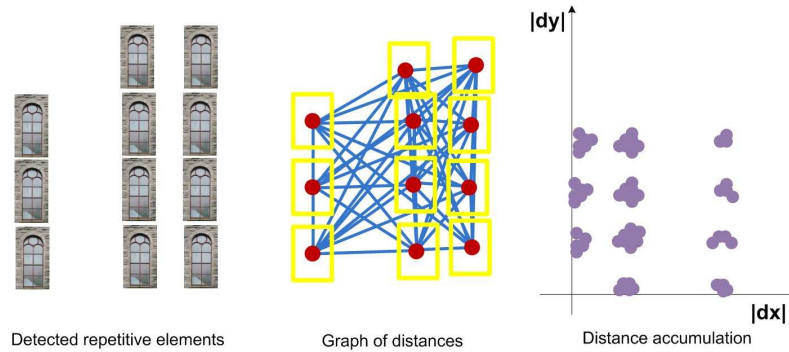
The idea is that each descriptor can be found at different positions inside an object and at different scales. Thus, a vote points from the position of a matched descriptor to the center of the object as it was associated in the codebook construction. In our case all the descriptors found inside a contour are included in the codebook  $\mathcal{C}_e$  as well as the relative displacement of the respective interest points with respect to the center of the contour. To retrieve objects repetitions we match objects in the following way:

1. All descriptors found in the image are matched against an object's codebook  $\mathcal{C}_e$ . Those with a Euclidean distance to the best match in  $\mathcal{C}_e$  that is bigger than a threshold  $\theta_d$  are discarded.
2. Votes casted by the matching descriptors are collected in a 2D voting space
3. We use mean shift mode estimation to find the object center from all votes. This is referred to as an object hypothesis.

To select valid hypotheses we propose a quality function that balances the strength of the votes with their spatial origin. Votes are accumulated in a circular histogram around the hypothetical object center. The detection quality function is given by:

$$q_i = w_a \cdot \frac{f_h(\alpha_i, \alpha_e)}{f_h(\alpha_e, \alpha_e)} + (1 - w_a) \cdot \frac{s_i}{s_e} \quad q_i \in [0, 1] \quad (1)$$

where  $\alpha_e$  is the vote orientation histogram of the object  $\mathcal{C}_e$ ;  $\alpha_i$  is the vote orientation histogram of the hypothesis  $i$ ;  $f_h$  is a function that applies an *AND* operator between the bins of two histograms and sums the resulting not empty bins.  $s_i, s_e$  are respectively the score (number of votes received for the hypothesis) and the score of  $O_e$ .  $w_a$  is the bias that is introduced between the two members. This is a simplified version of the cost function explained in [16]. Detected objects are selected by a simple minimum threshold  $\theta_q$  on the detection quality  $q_i$ . All the objects matching with  $O_e$  constitute the object category  $\tau$  that is defined by a codebook composed by descriptors that contributed to each match and all the entries of  $\mathcal{C}_e$ . Thus, a more complete description of the visual variability of the initial object instance  $O_e$  is achieved. It is important to notice that it is not required that every object in the image has a closed contour as soon as there is at least one of its category. In other words: if an image of a facade contains several windows of the same type, only one of them is required to have a closed contour. In this work we aim to match objects with the same scale. Same objects present at different scales in the image are treated as different object categories.



**Fig. 3.** Latticelet discovery process. Objects of the same category are detected. A complete graph is built and the relative distances are accumulated in the Cartesian plane.

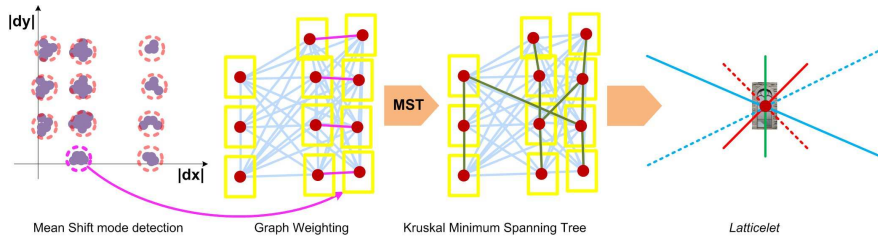
As a last step we use an hierarchical agglomerative clustering with average linkage to group visually similar categories by using a measure described by their codebook entries  $d(\tau_{\ell}^i, \tau_{\ell}^j) = \frac{L(\tau_{\ell}^i, \tau_{\ell}^j)}{\min(|\tau_{\ell}^i|, |\tau_{\ell}^j|)}$  where  $L$  computes the number of corresponding descriptors from the two codebooks with a Euclidean distance of less than  $\theta_d$  and  $|\tau_{\ell}^i|$  the number of codebook entries.

## 5 Analysis of Repetitive Objects

### 5.1 Latticelets

In this section we introduce the space frequency analysis for the discovered object categories. We name the detected object locations in the image as *nodes*. In order to analyze the repetition pattern of each object category we build a complete graph that connects all the nodes. Our aim is to select in this graph edges that have a repeated length and orientation. Moreover, we require our arc selection to include all the nodes. Our proposed solution is based on the use of a Minimum Spanning Tree (MST). From the complete graph we build a frequency map (see scheme Fig. 3 and Fig. 4), in which we store the distances  $|dx|, |dy|$  in pixels between nodes of the graph. The map represents the complete distance distribution between the nodes. We therefore have to select from this map the most representative modes. In order to estimate local density maxima in the frequency map we employ a two dimensional mean shift algorithm, with a simple circular kernel. Each convergence mode is expressed by a point in the map  $d\hat{x}, d\hat{y}$  and its score repetitiveness that is given by the number of points contributing to the basin of attraction. All the graph edges that contribute to each mode convergency are then labeled with their associated distance. At the end of this process we have obtained a graph in which the distances between the nodes have been relaxed by averaging similar consistent distances/orientations. Each edge is tagged with its repetitiveness score.

6 L. Spinello, R. Triebel, D. Vasquez, K. O. Arras, R. Siegwart



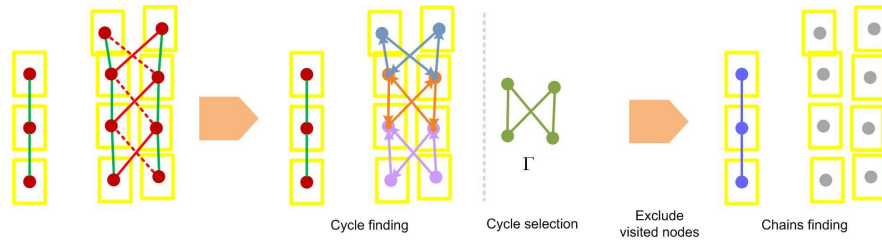
**Fig. 4.** Repetitive distances in  $x$  and  $y$  are clustered via mean-shift, the arcs are reweighed by their mode convergency score. The solid and dotted lines in the latticelet figure represent the possible directions expressed by the selected  $|dx|$  and  $|dy|$ .

As last step of this processing we employ Kruskal’s algorithm [17] to find the minimum spanning tree by using the nodes, their edge connectivity and the weight of the arcs. The resulting tree represents the most repetitive arcs sufficient to connect all the nodes. In order to compact the information we select each kind of arc just once. We call it latticelet, the minimal set of repetitive arcs that are needed to represent the original lattice. Each object category is associated to a latticelet that generalize its repetition pattern. Our method is able to cope with small perspective distortions thanks to the relaxation step. For larger deviations from a fronto-parallel image view, the problem of perspective estimation can be naturally decoupled from the one of analyzing repetitive patterns. The problem of image rectification could be addressed with many existing methods (e.g. [18]) that are far beyond the scope of this paper.

## 5.2 Cycles and chains

Latticelets contain very local information, they explain the direction of a possible predicted element from a given position. In order to incorporate higher level knowledge of the repetitive pattern of the neighborhood, we use cycles composed of latticelets arcs. Our aim is to find minimal size repetitive polygons. They provide the effective object repetition that is used in later stages to obtain prediction and simplification. For each category we sort the the weight of its latticelet arcs and we select the one with highest weight. We compose a new graph by using the selected arc to build connection between nodes and compute the smallest available cycle by computing its girth (i.e. length)  $\gamma$ .

A cycle  $\Gamma$  is computed by using an approach based on a Breadth-first Search algorithm. Starting from a node of choice in the graph, arcs are followed once, and nodes are marked with their number of visits. A cycle is found as soon as the number of visits for a node reaches two. This is done for all the nodes present in the object category detection set. We then collect all the cycles, and we select the one with the smallest number of nodes. We create a graph by using the connectivity offered by  $\Gamma$  and mark as removed the nodes that are connected by it. Thus, we add another latticelet arc until all the nodes are connected or all the latticelet arcs are used. We obtained a polygon set composed of frequent displacements suitable to describe the object distribution in the image (see scheme Fig. 5) and to generalize higher orders repetitions. An object category is therefore associated to  $k$  small cycles:  $\mathcal{S} = \{\Gamma_1, \dots, \Gamma_k\}$ .



**Fig. 5.** From the graph created by an incremental set of latticelet’s arcs, small repetitive cycles  $\Gamma$  are selected by using a Breadth-first Search algorithm. Chains are created on the remaining nodes that have not been satisfied by any polygonal cycles  $\mathcal{G}$ .

In addition to what has been explained above, the algorithm tries to represent with chains the nodes that cannot be described with polygonal cycles. The procedure is analogous to the former one: chain arcs are selected by using the sorted latticelet set. The procedure is run for each object category.

## 6 Structure Inference using Conditional Random Fields

So far, we showed our method to detect objects represented as closed contours and to find repetitive patterns in the occurrence of such objects. However, in many cases, objects can not be detected due to occlusions or low contrast in the image. In general, the problem of these false negative detections can not be solved, as there is not enough evidence of the occurrence of an object. In our case, we can use the additional knowledge that similar objects have been detected in the same scene and that all objects of the same kind are grouped according to a repetitive pattern. Using these two sources of information, we can infer the existence of an object, even if its detection quality is very low. We achieve this by using a probabilistic model: each possible location of an object of a given category  $\tau$  is represented as a binary random variable  $l_\tau(\mathbf{x})$  which is true if an object of category  $\tau$  occurs at position  $\mathbf{x}$  and false otherwise. In general, the state of these random variables can not be observed, i.e. they are *hidden*, but we can observe a set of features  $\mathbf{z}(\mathbf{x})$  at the given position  $\mathbf{x}$ . The features  $\mathbf{z}$  here correspond to the detection quality defined in Eqn. (1). The idea now is to find states of all binary variables  $\mathbf{l}_\tau = \{l_\tau(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$  so that the likelihood  $p(\mathbf{l}_\tau \mid \mathbf{z})$  is maximized. In our formulation we will not only reflect the dependence between the variables  $l$  and  $\mathbf{z}$ , but also the *conditional dependence* between variables  $l_\tau(\mathbf{x}_1)$  and  $l_\tau(\mathbf{x}_2)$  given  $\mathbf{z}(\mathbf{x}_1)$  and  $\mathbf{z}(\mathbf{x}_2)$ , where  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are positions that are very close to each other. The intuition behind this is that the occurrence probability of an object at position  $\mathbf{x}_1$  is higher if the same object already occurred at position  $\mathbf{x}_2$ . We model this conditional dependence by expressing the overall likelihood  $p(\mathbf{l}_\tau \mid \mathbf{z})$  as a CRF.

### 6.1 Conditional Random Fields

A CRF is an undirected graphical model that represents the joint conditional probability of a set of hidden variables (in our case  $\mathbf{l}_\tau$ ) given a set of observations  $\mathbf{z}$ . A node in



the graph represents a hidden variable, and an edge between two nodes reflects the conditional dependence of the two adjacent variables. To compute  $p(\mathbf{l}_\tau | \mathbf{z})$ , we define *node potentials*  $\varphi$  and *edge potentials*  $\psi$  as

$$\varphi(\mathbf{z}_i, l_{\tau i}) = e^{\mathbf{w}_n \cdot \mathbf{f}_n(\mathbf{z}_i, l_{\tau i})} \quad \text{and} \quad \psi(\mathbf{z}_i, \mathbf{z}_j, y_i, y_j) = e^{\mathbf{w}_e \cdot \mathbf{f}_e(\mathbf{z}_i, \mathbf{z}_j, l_{\tau i}, l_{\tau j})}, \quad (2)$$

where  $\mathbf{f}_n$  and  $\mathbf{f}_e$  are feature functions for the nodes and the edges in the graph (see below), and  $\mathbf{w}_n$  and  $\mathbf{w}_e$  are the feature weights that are determined in a training phase from hand-labeled training data. Using this, the overall likelihood is computed as

$$p(\mathbf{l}_\tau | \mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{i=1}^N \varphi(\mathbf{z}_i, l_{\tau i}) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{z}_i, \mathbf{z}_j, l_{\tau i}, l_{\tau j}), \quad (3)$$

where  $Z$  is the *partition function*,  $N$  the number of nodes, and  $\mathcal{E}$  the set of edges in the graph. The computation of the partition function  $Z$  is intractable due to the exponential number of possible states  $\mathbf{l}_\tau$ . Instead, we compute the *log-pseudo-likelihood*, which approximates  $\log p(\mathbf{l}_\tau | \mathbf{z})$

In the training phase, we compute the weights  $\mathbf{w}_n$  and  $\mathbf{w}_e$  that minimize the negative log pseudo-likelihood together with a Gaussian shrinkage prior. In our implementation, we use the Fletcher-Reeves method [19]. Once the weights are obtained, they are used in the detection phase to find the  $\mathbf{l}_\tau$  that maximizes Eq. (3). Here, we do not need to compute the partition function  $Z$ , as it is not dependent on  $\mathbf{l}_\tau$ . We use max-product loopy belief propagation [20] to find the distributions of each  $l_{\tau i}$ . The final classification is then obtained as the one that is maximal at each node.

## 6.2 Node and Edge Features

As mentioned above, the features in our case are directly related to the detection quality obtained from Eqn. (1). In particular, we define the node features as  $\mathbf{f}_n(q_i, l_{\tau, i}) = 1 - l_{\tau, i} + (2l_{\tau, i} - 1)q_i$ , i.e. if the label  $l_{\tau, i}$  is 1 for a detected object, we use its detection quality  $q_i$ , otherwise we use  $1 - q_i$ . The edge feature function  $\mathbf{f}_e$  computes a two-dimensional vector as follows:

$$\mathbf{f}_e(q_i, q_j, l_{\tau i}, l_{\tau j}) = \begin{cases} \frac{1}{\gamma} \begin{pmatrix} f_{e1} & f_{e2} \end{pmatrix} & \text{if } l_{\tau i} = l_{\tau j} \\ \begin{pmatrix} 0 & 0 \end{pmatrix} & \text{else} \end{cases} \quad \text{with} \quad \begin{cases} f_{e1} = \max(\mathbf{f}_n(q_i, l_{\tau i}), \mathbf{f}_n(q_j, l_{\tau j})) \\ f_{e2} = \max_{G \in \mathcal{G}_{ij}}(\mathbf{f}_n(\eta(G), l_{\tau i})), \end{cases}$$

where  $\mathcal{G}_{ij}$  is the set of (maximal two) minimum cycles  $\Gamma$  that contain the edge between nodes  $i$  and  $j$ , and  $\eta(\Gamma)$  is a function that counts the number of detected objects along the cycle  $\Gamma$ , i.e. for which the detection quality is above  $\theta_q$ .

## 6.3 Network Structure

The standard way to apply CRFs to our problem would consist in collecting a large training data set where all objects are labeled by hand and for each object category  $\tau$  a pair of node and edge features is learned so that  $p(\mathbf{l}_\tau | \mathbf{z})$  is maximized. However, this approach has two major drawbacks:

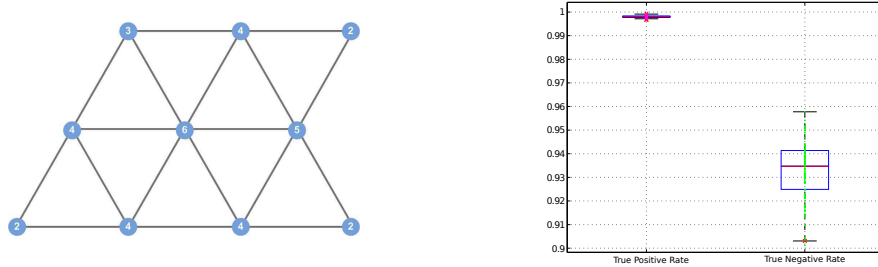
- For a given object category  $\tau$ , there are different kinds of lattice structures in which the objects may appear in the training data. This means that the connectivity of a given object inside its network varies over the training examples. Thus, the importance of the edges over the nodes can not be estimated in a meaningful way.
- In such a supervised learning approach, only objects of categories that are present in the training data can be detected. I.e., if the CRF is trained only on, say, some different kinds of windows, it will be impossible to detect other kinds of objects that might occur in repetitive patterns in a scene. Our goal however, is to be independent of the object category itself and to infer only the structure of the network. In fact, the object category is already determined by the similarity detection described above.

To address these issues, we propose a different approach. Considering the fact that from the training phase we only obtain a set of node and edge weights  $\mathbf{w}_n$  and  $\mathbf{w}_e$ , which do not depend on the network geometry but only on its topology, we can artificially generate training instances by setting up networks with a given topology and assigning combinations of low and high detection qualities  $q_i$  to the nodes. The advantage of this is that we can create a higher variability of possible situations than seen in real data and thus obtain a higher generalization of the algorithm. The topology we use for training has a girth  $\gamma$  of 3 and is shown in Fig. 6 on the left. Other topologies are possible for training, e.g. using squared or hexagonal cycles, but from experiments we carried out it turns out that the use of such topologies does not increase the classification result. The graph in Fig. 6 right illustrates that. It shows the true positive and the true negative rates from an experiment with 100 test data sets, each consisting of networks with a total of 5000 to 10000 nodes. The training was done once only with a triangular topology (TriTop) and once also including square and hexagonal topologies (MixTop), which represent all possible regular tessellations of the plane. As the graph shows, there is no significant difference in the two classification results. In contrast to the topology, the number of outgoing edges per node, i.e. the *connectivity*, has a strong influence on the learned weights. Thus, we use a training instance where all possible connectivities from 2 to 6 are considered, as shown in Fig. 6 left.

In the inference phase, we create a CRF by growing an initial network. From the analysis of repetitive patterns described above, we obtain the set  $\mathcal{G}$  for each category, the topology and edge lengths of the lattice. By subsequently adding cycles from  $\mathcal{G}$  to the initial network obtained from the already detected objects, we grow the network beyond its current borders. After each growing step, we run loopy belief propagation to infer the occurrence of objects with low detection quality. The growth of the network is stopped as soon as no new objects are detected in any of the 4 directions from the last inference steps.

## 7 Model Compression

One aim of our work is to show that the information contained in an image (e.g. a facade) can be compressed using the proposed repetition detection technique. We reduce the image to a simple set of detected object categories, their repetition scheme, and a simplified background extraction. More in detail: each object category is stored as a



**Fig. 6. Left:** Triangular lattice topology used for training the CRF. The numbers inside the nodes show the connectivity of the nodes. **Right:** Comparison of CRF performances using TriTop and MixTop datasets for training. True positive and the true negative rates are evaluated. The result from the TriTop data are shown in box-and-whiskers mode, the MixTop result as dots. We can see that using different topologies for learning gives no significant change in the classification result.

set of codebook descriptors and vote vectors, a rectangular colorscale bitmap resulting from averaging the image areas inside the detected elements bounding boxes. To visually simplify the image background, we assume that the space between detected elements in a category is covered by textures of the same kind. We sort object categories by their cardinality. Then, as a texture simplification, we compute the median color between the elements by sampling squared image patches. This color is assigned to a rectangle patch that extends from top to the bottom of each category. We iterate this procedure until all the image is covered. Missing empty spaces are filled with the color of the most populous group. Some examples are shown in the right part of Fig. 9.

An image compressed with our method can be used in a number of applications such as visual based localization, in which information is extracted only from the repeated pattern, or low-bitrate storage for embedded systems (e.g. UAV) that have to store/transmit large urban environments. In a more general fashion we consider that our approach should be useful in all those cases where the main goal is to identify places where repetitive patterns are present, although it is not as well suited to provide detailed reconstructions of the represented objects.

## 8 Experiments

The goal of our experimental evaluation is to investigate to which extent the proposed algorithm is capable to detect different categories of objects, to detect repetition rules and to run inference based on that information.

In order to obtain rich statistics on a wide range of object categories we prepared an image evaluation set composed of high contrast polygons at different sizes. 150 pictures of  $450 \times 150$  pixels size have been computer generated, each one containing 2 to 8 different object categories. An object category is defined by a type of a polygon. It is important to stress that such set evaluates not the detection capabilities but the capacity

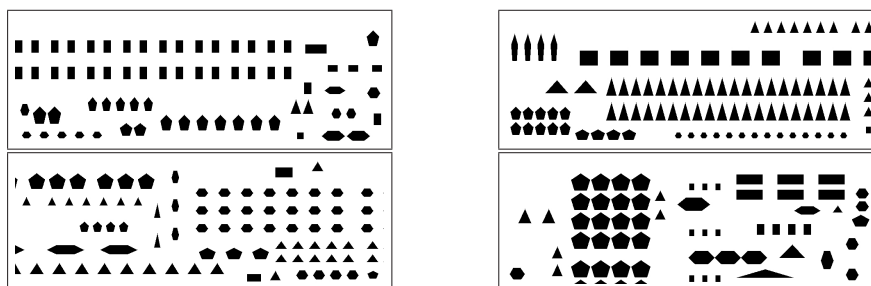


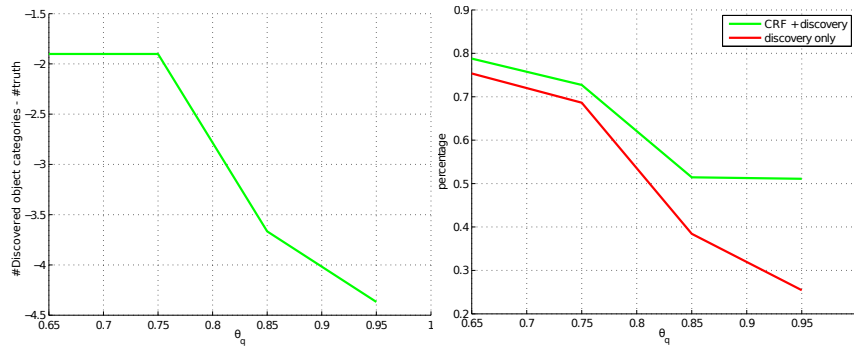
Fig. 7. Samples from the evaluation data set

of grouping similar elements, detecting latticelets and inferring high level cycles and chains for model compression and completion. Polygons are described by few pixels to introduce ambiguity in the description of repetitive elements. Fig. 7 shows some samples from the evaluation dataset.

One of our goals is to assess the quality of object category distinction and grouping, that is fundamental for the creation of the graph, as well as its analysis. It is important to note that the angle difference between a hexagon and a pentagon is just  $12^\circ$  and in small scales, due to pixel aliasing, this difference may not be easy to distinguish. Fig. 8 left shows the average difference between the number of detected categories and annotated categories. The graph is plotted with respect to the minimum detection quality  $\theta_b$  needed for each node. We can notice that the algorithm tends to under-explain the data trying to not overfit single detections. This is the result of the soft detection and grouping strategy we use that favors the merging of similar categories to the creation of a new one.

Moreover, we evaluate the contribution of the CRF to the detection rate of repetitive elements present in the image. We plot, in Fig. 8 right, this measure with respect to  $\theta_b$  and we overlay the results using CRF. The left side of the graph shows the CRF contribution (4%) when many annotated objects have been already detected by the discovery process, the right one shows the performance when just few elements are detected. In the latter case, a sound 20% detection rate improvement is achieved: it suffices that a small group of elements is detected for generating a set of  $\mathcal{G}$  used for inferring many missing aligned low-detection nodes. Important to mention is the average of false positives per image: 0.2. CRF therefore increases the true positive rate and it guarantees a very low false positive rate.

We also performed a quantitative analysis of compression ratio for the images in the evaluation set and the real-world images displayed in Fig. 9-right. The resulting compressed image is very compact and it stores just one bitmap for each object category and a list of 2D coordinates of elements locations. If we employ the ratio in bytes between the compressed image and the raw input image for the testing set images we obtain 1.4% ratio, for the pictures displayed in Fig. 9 (top to bottom order), we obtain: 2%, 1.2%, 2.3%, 0.8%, 2.8%, 8%. Even though this method aggressively reduces the amount of image details, the salient repetitive pattern is preserved.



**Fig. 8. Left:** Average difference between the number of detected categories and annotated categories. The algorithm tends to under-explain the data trying to not overfit single detections. **Right:** Discovery only detection and discovery + CRF detection. The contribution of CRF for detecting missing elements is particularly evident when a low detection rate is obtained. Graphs are plotted with respect to the minimum detection quality  $\theta_b$  needed for each node.

A set of images of facades and other repetitive elements have been downloaded from internet and treated as input for our algorithm, Fig. 9. On each of the examples the difference from discovery and CRF-completed image is shown. It is interesting to notice that the algorithm works also for not rectified facades and several kind of architectural or repetitive elements. In the scope of this work it is evident that training on a simulated data, sufficiently rich in variability, satisfies also real world examples.

## 9 Conclusions

In this paper we presented a probabilistic technique to discover and reason about repetitive patterns of objects in a single image. We introduced the concepts of latticelets, generalized building blocks of repetitive patterns. For high-level inference on the patterns, CRFs are used to soundly couple low-level detections with high-level model information.

The method has been tested on simulated and real data showing the effectiveness of the approach. From a set of synthetic images, it was verified that the method is able to correctly learn different object categories in an unsupervised fashion regardless the detection thresholds. For the task of object detection by model prediction and completion, the experiments showed that the method is able to significantly improve detection rate by reinforcing weak detection hypotheses with the high-level model information from the repetitive pattern. This is especially true for large thresholds for which detection only, without our method, tends to break down. For the task of model compression, i.e. retaining and efficiently representing the discovered repetitive patterns, a very high compression ratio of up to 98% with respect to the raw image has been achieved.

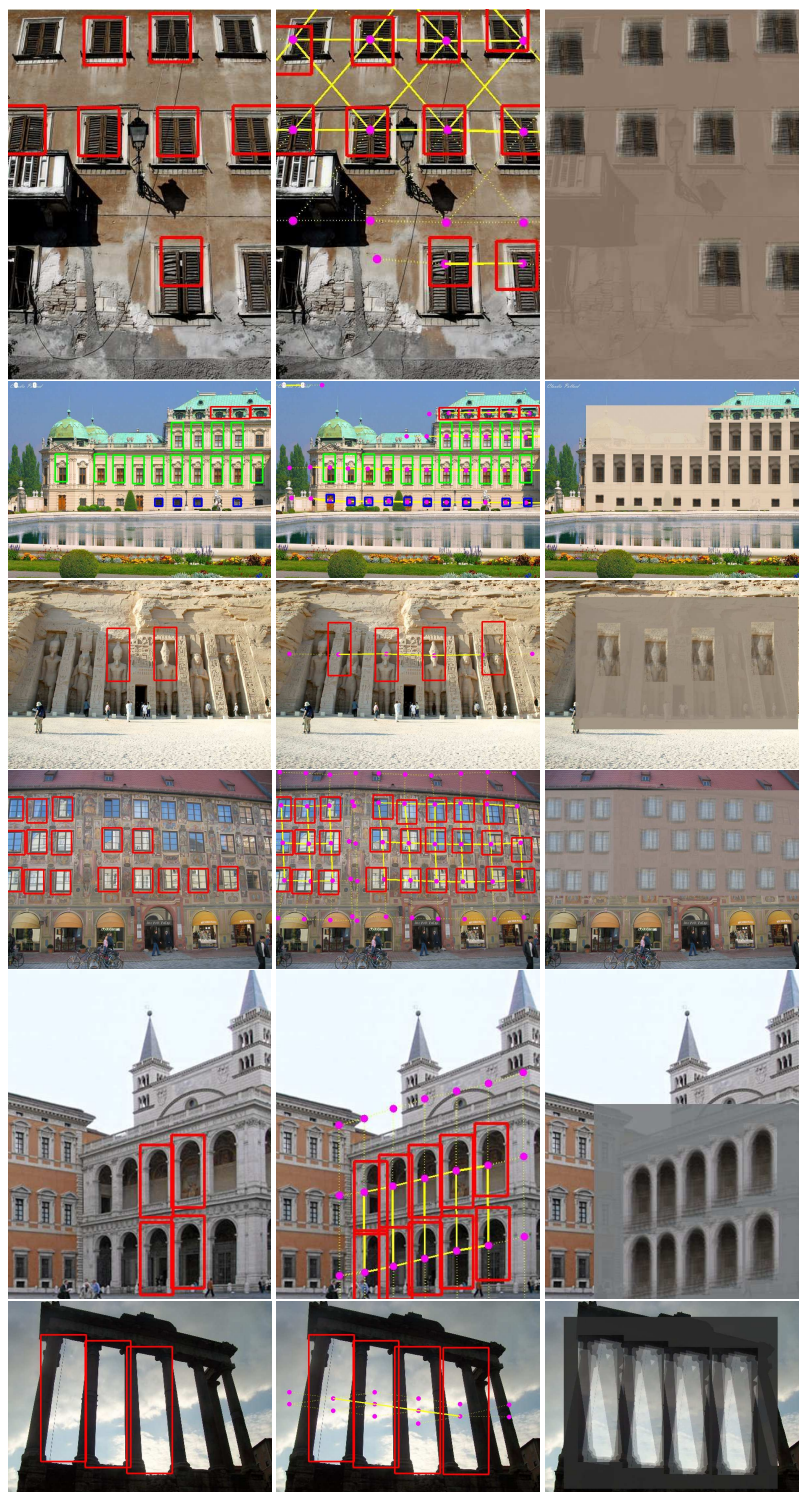
Beyond the tasks of model completion and compression, we see applications of this method in image inpainting, environment modeling of urban scenes and robot navigation in man-made buildings.

## Acknowledgements

This work has been partly supported by German Research Foundation (DFG) under contract number SFB/TR-8 and EU Project EUROPA-FP7-231888.

## References

1. Leibe, B., Leonardis, A., Schiele, B.: Combined object categorization and segmentation with an implicit shape model. In: ECCV Workshop on Stat. Learn. in Comp. Vis. (2004)
2. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmentation and labeling sequence data. In: Int. Conf. on Mach. Learn. (ICML). (2001)
3. Dick, A.R., Torr, P.H.S., Cipolla, R.: Modelling and interpretation of architecture from several images. *Int. Journ. of Comp. Vis.* **60** (2004) 111–134
4. Mayer, H., Reznik, S.: Building facade interpretation from image sequences. In: ISPRS Workshop CMRT 2005. (2005)
5. Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., Quan, L.: Image-based façade modeling. In: ACM SIGGRAPH Asia. (2008)
6. Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W.: Instant architecture. *ACM Trans. Graph.* **22** (2003) 669–677
7. Müller, P., Zeng, G., Wonka, P., Van Gool, L.: Image-based procedural modeling of facades. *ACM Trans. Graph.* **26** (2007) 85
8. Schaffalitzky, F., Zisserman, A.: Geometric grouping of repeated elements within images. In: British Machine Vision Conf. (1999)
9. Turina, A., Tuytelaars, T., Van Gool, L.: Efficient grouping under perspective skew. In: IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). (2001)
10. Liu, Y., Collins, R.T., Tsin, Y.: A computational model for periodic pattern perception based on frieze and wallpaper groups. *IEEE Trans. Pattern An. & Mach. Intell.* **26** (2004) 354–371
11. Hays, J., Leordeanu, M., Efros, A.A., Liu, Y.: Discovering texture regularity as a higher-order correspondence problem. In: European Conf. on Comp. Vision (ECCV). (2006)
12. Korah, T., Rasmussen, C.: Analysis of building textures for reconstructing partially occluded facades. In: European Conf. on Comp. Vision (ECCV). (2008)
13. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern An. & Mach. Intell.* **24** (2002) 509–522
14. Canny, J.: A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **8** (1986) 679–698
15. Freeman, H.: On the encoding of arbitrary geometric configurations. *IEEE Trans. Electr. Computers* **EC-10** (1961) 260–268
16. Spinello, L., Triebel, R., Siegwart, R.: Multimodal people detection and tracking in crowded scenes. In: Proc. of the AAAI Conf. on Artificial Intelligence. (2008)
17. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* **7** (1956) 48–50
18. Collins, R., Beveridge, J.: Matching persp. views of compl. structures using projective un-warping and sim. matching. In: IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR). (1993)
19. Fletcher, R.: *Practical Methods of Optimization*. Wiley (1987)
20. Pearl, J.: *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc. (1988)



**Fig. 9.** **Left Column:** Extracted self-similar objects (red boxes). Note that often only a few number of instances are found. **Center Column:** Final CRF lattice (dots and lines) and inferred position of objects (boxes). **Right Column:** Reconstruction of images based on our model compression.

## Inferring the Semantics of Direction Signs in Public Places

Jérôme Maye\*, Luciano Spinello\*<sup>†</sup>, Rudolph Triebel\*, and Roland Siegwart\*

\* Autonomous Systems Lab, ETH Zurich, Switzerland

email: {jerome.maye, rudolph.triebel, roland.siegwart}@mavt.ethz.ch

<sup>†</sup> Social Robotics Lab, Department of Computer Science, University of Freiburg, Germany

email: spinello@informatik.uni-freiburg.de

**Abstract**—Most large-scale public environments provide direction signs to facilitate the orientation for humans and to find their way to a goal location in the environment. Thus, for a robot operating in the same environment, it would be beneficial to interpret such signs correctly for a safe and efficient navigation.

In this work, we propose a novel approach to infer the meaning of direction signs and to use that for navigation, i.e., to find a mapping of a detected sign to a motion direction. Our method uses a hierarchical extension of the Implicit Shape Model framework called HISM that does not require any hand-labeled training data to detect the signs. On the lower level of this two-stage hierarchy, ISM is applied to image descriptors as in the standard approach. On the higher level, ISM operates on subparts of signs called tokens, using weights learned from data. The interpretation of the signs is inferred by associating navigation data to direction instructions. We conducted experiments from image data acquired in an airport terminal, aiming towards the implementation of a robotic guide, with promising results.

### I. INTRODUCTION

Human beings can relatively easily find their way in an unknown environment using direction signs. The first use of these artifacts goes back to the Roman Empire, where milestones were placed along the dense road network to indicate the distances to the nearby major cities. Since that time, signs evolved in a more convenient form which generally consists of a symbol suggesting the direction, a distance indicator, and a part identifying the destination. Signs are nowadays not only restricted to roads. They actually serve as the main cues for navigation in most public places like train stations, airport terminals, or event venues. Given the high density of direction signs in our daily environment, it would thus be beneficial for a robot to be able to read these signs correctly for an efficient navigation. This paper explores this idea and applies it to a typical environment where the signs are of particular importance: an airport terminal.

Our approach addresses this challenge by reasoning on single images. The method presented in this paper is not based on models that are manually designed beforehand, but instead infers them from data. Moreover, we design a system that is able to generalize over several categories of signs by using object detection techniques. Whenever a sign is presented to the system, it is divided into subparts or *tokens*. The sign is therefore characterized by a hierarchical method which builds on a hierarchy of Implicit Shape Models (ISM) [1]. Specifically, a sign is defined by a geometrical

arrangement of subparts and these subparts are again defined by a geometrical arrangement of primitive image feature descriptors. We term this method *Hierarchical Implicit Shape Models (HISM)*. Furthermore, if several kinds of signs are presented to the robot, this method allows to understand which are the most distinctive subparts by using a smart weighting approach. For further learning, we use our sign detection technique to find a mapping between detections and motion directions, thus inferring the meaning of direction arrows. This is achieved by analyzing the frequency of certain subparts related to signs.

In this paper, we show possible applications for a robotic guide that navigates in an unknown environment or for a robotic assistant that identifies which sign to follow to reach a desired goal.

In particular, the major contributions of this work are:

- Hierarchical Implicit Shape Models (HISM): a hierarchical subdivision and voting strategy of a sign. This allows robustness and subpart weighting.
- Unsupervised subpart clustering and description as an object: uniformly clustered color regions are described by a geometrical voting model of standard image features.
- Mapping actions to detections to learn semantical sign information: unsupervised detection of direction arrows.

The rest of the paper is organized as follows. Section II reviews the related work in the domain. Section III describes our approach for learning signs. Section IV shows how we achieve sign detection. Section V demonstrates how to map actions to sign detections. Section VI presents experiments. Section VII outlines our conclusions and future work.

### II. RELATED WORK

To our knowledge, there has been little work in the topic of unsupervised sign analysis. The work of Quingji *et al.* [2] is based on a similar experimental environment. It makes use of a Pan-Tilt-Zoom camera for obtaining detection based on SIFT features [3] without any further reasoning.

Other literature focuses on traffic signs: either their detection, their recognition, or both. For detection, the most interesting approaches are inspired by the object detection method proposed by [4], which is based on a cascade of boosted classifiers working on Haar-like features. Several authors report promising results with this technique [5], [6], [7]. Some other detection algorithms are based on color



segmentation or geometrical features [8] [9], and finally on Distance Transform (DT) matching [10].

For traffic sign recognition, several machine-learning methods have been experimented, including Support Vector Machines [7], [8], a Bayesian generative model [6] and an Error-Correcting Output Code (ECOC) framework [5].

Most relevant to HISM is the work by Andriluka *et al.* [11]. There, the authors propose a part-based model for pedestrian detection in which each part votes for the object center. With our method, we overcome the need of manual subpart annotation: subparts are individuated automatically by a consistency segmentation rule. Moreover, we do not employ any supervised learning technique: our method is able to generalize recognition of the same sign with signs that consist of similar appearance and similar geometrical subparts distributions in an unsupervised manner. Furthermore, [11] learns a Gaussian distribution of the position of each manually labeled part relative to the object center that acts as a soft skeletal model; we do not impose any subpart spatial distribution and each subpart can independently vote for an object center.

### III. LEARNING HUMAN SIGNS

Our algorithm takes as input a picture of a sign, called *target sign*, that describes a desired destination. The method is then able to analyze any newly recorded image for the presence and the position of such a sign, probabilistically and without any hand-tuned models. Techniques like chamfer matching [12] or cross-correlation [13] aim at matching the exact visual appearance of the target sign with a test image. However, in cluttered and crowded environments these methods are prone to fail due to the weak description of the object. Moreover, the user might show a target sign not precisely aligned or totally visible. This can produce problems to a simple gradient matching method. Instead, we can achieve a higher level of robustness by describing a target sign with standard local image descriptors [14]. By matching such local descriptors, we obtain a far more reliable correspondence. Unfortunately, such an approach does not take into account that some parts of a sign are more important than others. As an example, numbers contain just a few features but they are a very descriptive part of a sign. Therefore, we developed a novel unsupervised matching method that overcomes these problems by producing a hierarchy in the object: a sign is automatically divided into subparts, described by standard local image descriptors; subparts compose a sign by defining geometrical constraints. Furthermore, the subpart description of our method allows importance weighting of each single subpart.

#### A. Features Extraction

The first step to robustly describe signs is to extract a set of local image descriptors. As we expect to match basic geometrical shapes contained in the signs, we make use of Shape Context descriptors [15] computed at Hessian-Laplace interest points [16]. This allows a quite dense description of the sign and, at the same time, a robust representation of local

image regions. We denote interest points as vectors  $\mathbf{x}_i = (x, y)$ , their scales  $\gamma_i$ , and a set of interest points as  $\mathcal{X}$ . A descriptor computed at  $\mathbf{x}_i$  is represented as a  $d$ -dimensional vector  $\mathbf{h}_i$  that is part of a descriptor set  $\mathcal{H}$ .

It is important to notice that rotational invariance of the descriptors is not desired in our case, since we want to distinguish a 6 from a 9. We also note that our method is not only restricted to Shape Context descriptors, but can work properly with any other robust image features.

#### B. Sign Decomposition

In the next step of our algorithm, we divide a given sign into smaller subparts, called *tokens*. Signs are intrinsically designed to be clearly distinguishable by human eyes in any environment. For example, for traffic signs, the color schemes are regulated by an international convention [17]. Signs for public places generally follow the same rules. The colors used for the shapes are chosen to have a high contrast with respect to the background. By following these reasons, we define that uniform coherent and high-contrast areas delineate subparts. We thus proceed by binarizing the sign. Otsu's segmentation method [18] is able to accomplish this task in a robust and fast manner. It automatically selects the best thresholding value by minimizing the intra-class variance of foreground and background estimated pixels.

Then, a region growing algorithm [19] is applied to the active pixels of the binary image in order to group adjacent neighboring pixels into coherent regions. In order to remove inherent noise, we fix a minimum number of pixels  $\sigma$  per region. Finally, an agglomerative hierarchical clustering algorithm with average link [20] groups nearby regions into clusters by considering the distance between centers of gravity and using a threshold  $\theta_d$ . This process avoids that a sign is decomposed into too many small regions. We then compute the bounding box  $B$  for each cluster and define the set of all interest points and corresponding descriptors inside  $B$  as a *sign token*  $\mathcal{T}$ :

$$\begin{aligned} \mathcal{T} &:= \{(\mathbf{x}_1, \mathbf{h}_1), \dots, (\mathbf{x}_m, \mathbf{h}_m), B\} \\ \text{where } \mathbf{x}_i &\in B \quad \forall i = 1, \dots, m \\ \text{and } \mathbf{h}_i &\text{ is computed at } \mathbf{x}_i \quad \forall i \end{aligned} \quad (1)$$

It is important to remark that this method does not exploit the color or the shape of the cluster as a subpart descriptor, but it just uses the cluster as a coherent container for robust local image descriptors. Furthermore, our algorithm is not restricted to this procedure. Actually, any fast and robust segmentation method could be used at this stage.

#### C. Learning the Sign Hierarchy

Once a sign has been divided into tokens  $\mathcal{T}$ , we extract *geometrical information* in addition to the *appearance information* given by the descriptors  $\mathbf{h}_i$  in each  $\mathcal{T}$ . An elegant and well established way to achieve this is by means of Implicit Shape Models (ISM) [1]. An Implicit Shape Model describes an object by a *codebook* of local appearance, i.e., a collection of local image descriptors, and the displacements between

their associated interest points and the object center. In our case, this means that a codebook  $\mathcal{C}^T$  for a token  $\mathcal{T}$  consists of all image descriptors  $\mathbf{h}_i \in \mathcal{T}$  and all corresponding displacement vectors  $\mathbf{v}_i$ , where  $\mathbf{v}_i = \mathbf{x}_i - \mathbf{c}^T$  and  $\mathbf{c}^T$  is the center of gravity of  $\mathcal{T}$ . Thus, we define:

$$\mathcal{C}^T := \{(\mathbf{h}_1, \mathbf{v}_1), \dots, (\mathbf{h}_m, \mathbf{v}_m)\}. \quad (2)$$

Now, using  $\mathcal{C}^T$ , we can describe sign tokens, but for a reasoning on the higher level of signs, we need more information about the arrangement of the tokens in a sign. We proceed by introducing a hierarchical ISM of tokens, called *HISM*. A codebook  $\mathcal{C}^S$  of this hierarchical ISM is defined as the set of all tokens  $\mathcal{T}_i$  of a sign  $\mathcal{S}$  along with the displacements between the tokens' centroids  $\mathbf{c}^{\mathcal{T}_i}$  and the center  $\mathbf{c}^S$  of the sign, and a weighting factor  $g_i$ , i.e.:

$$\begin{aligned} \mathcal{C}^S &:= \{(\mathcal{T}_1, \mathbf{w}_1, g_1), \dots, (\mathcal{T}_n, \mathbf{w}_n, g_n)\} \\ \text{where } \mathbf{w}_i &= \mathbf{c}^{\mathcal{T}_i} - \mathbf{c}^S \forall i = 1, \dots, n \end{aligned} \quad (3)$$

and  $g_i$  is the weight of  $\mathcal{T}_i$  (defined below)

At this point, we note that the boundaries of a sign  $\mathcal{S}$  must be given to be able to compute its center  $\mathbf{c}^S$ . Also, information about which tokens belong to which sign must be available to be able to create the codebook  $\mathcal{C}^S$ . In this paper, we assume that a sign is always given in form of its bounding box. This ensures that  $\mathbf{c}^S$  and the tokens that belong to a sign are uniquely defined. This is nevertheless possible to remove this limitation with a clever segmentation algorithm, but this was not the focus of our work. The entire target sign processing is illustrated in Fig. 1.

#### D. Learning Token Weights

Using the HISM as described so far, we can compute a high-level codebook  $\mathcal{C}^S$  for each target sign and use it to find matching signs in new images presented to the robot. The details of this detection step are described in the next section. Before, however, we note that signs often differ only by a very small fraction, such as, e.g., one digit in the two signs for `Check-in 1` and `Check-in 3`. Using the HISM to match two signs like these would result in a high matching score, although the signs are significantly different. To address this issue, we additionally extract information about which tokens in a sign are most distinctive.

The intuition we use here is the fact that distinctive tokens occur only rarely in a given set of target signs. Assuming we are given a set of target signs  $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_{m'}\}$ . Now, for each sign  $\mathcal{S}_i$  in  $\mathcal{S}$ , we match all tokens  $\mathcal{T}_j^i$  of  $\mathcal{S}_i$  with all tokens from the other signs in  $\mathcal{S}$ . As a result, we obtain a *matching score*  $s$ , defined below, for the correspondence between the tokens  $\mathcal{T}_j^i$  and  $\mathcal{T}_{j'}^{i'}$ . This score is high if the tokens match well and low otherwise. A measure  $g_j$  of how well a token  $\mathcal{T}_j^i$  matches *in general* can then be determined by summing up all matching scores:

$$g_j(\mathcal{T}_j^i) := \sum_{i'=1}^{m'} \sum_{j'=1}^{m(i')} s(\mathcal{T}_j^i, \mathcal{T}_{j'}^{i'}), \quad (4)$$

where  $m(i')$  is the number of tokens encountered in sign  $\mathcal{S}_{i'}$ . Using this definition, the value of  $g_j$  for each token  $\mathcal{T}_j^i$  characterizes its discriminative factor and is used at a later stage as a voting weight.

#### IV. SIGN DETECTION

Whenever a user shows a target sign  $\mathcal{S}$  to the robot, its task is to navigate to the goal position where  $\mathcal{S}$  is encountered. To achieve this, it needs to match all new images with  $\mathcal{S}$  and extract a direction indicator, i.e., an arrow, to infer its direction of motion. As a first step, the sign  $\mathcal{S}$  is matched within the existing database  $\mathcal{S}$ . In case  $\mathcal{S} \notin \mathcal{S}$ , its HISM is computed and stored in  $\mathcal{S}$ . Moreover, all the token weights  $g_j$  are updated.

Then, for a given new image  $I$  presented to the robot, all the interest points  $\mathbf{x}_i^I$  and shape descriptors  $\mathbf{h}_i^I$  are computed. These are then matched with all descriptors  $\mathbf{h}_j$  found in the codebook  $\mathcal{C}^{\mathcal{T}_k}$  of  $\mathcal{S}$ . The matching is carried out using a nearest neighbor distance ratio strategy [14]. The Euclidean distance  $d(\mathbf{h}_i^I, \mathbf{h}_j)$  defines the matching score between  $\mathbf{h}_i^I$  and  $\mathbf{h}_j$ . Let the first and second best matching descriptors be  $\mathbf{h}_{j_1}$  and  $\mathbf{h}_{j_2}$ . A matching pair  $(\mathbf{h}_{i_1}^I, \mathbf{h}_{j_1})$  is detected, if

$$d(\mathbf{h}_{i_1}^I, \mathbf{h}_{j_1}) \leq \vartheta_m d(\mathbf{h}_{i_1}^I, \mathbf{h}_{j_2}), \quad (5)$$

where  $\vartheta_m$  is a distance ratio.

Each descriptor  $\mathbf{h}_i^I$  that matches a descriptor  $\mathbf{h}_j$  in a codebook  $\mathcal{C}^{\mathcal{T}_k}$  casts a *vote* for an occurrence of the token  $\mathcal{T}_k$  at the position

$$\begin{aligned} \mathbf{p}_{ij} &= \mathbf{x}_i - \mathbf{v}_j \delta_{ij} \\ \delta_{ij} &= \frac{\gamma_i}{\gamma_j}, \end{aligned} \quad (6)$$

where  $\mathbf{x}_i$  and  $\gamma_i$  are the interest point and scale related to  $\mathbf{h}_i^I$ ,  $\mathbf{v}_j$  and  $\gamma_j$  are the stored displacement vector and scale related to  $\mathbf{h}_j$ , and  $\delta_{ij}$  is the scale of the vote. Moreover, each vote is weighted inversely proportionally to the matching distance:

$$w_{ij} := \frac{1}{1 + d(\mathbf{h}_i^I, \mathbf{h}_j)} \quad (7)$$

All votes  $\mathbf{q}_{ij} = (\mathbf{p}_{ij}, \delta_{ij})$  are then collected in a voting space  $\mathcal{W}$ . Occurrences of the token  $\mathcal{T}_k$  are determined by finding high density loci in  $\mathcal{W}$ . To this end, we use mean shift mode estimation [21] with a spherical uniform kernel and a scale-adaptive bandwidth. This method starts at a random point  $\mathbf{q} \in \mathcal{W}$  and iterates over computing the mean  $\bar{\mathbf{q}}$  in a local vicinity of  $\mathbf{q}$  and assigning  $\bar{\mathbf{q}}$  to  $\mathbf{q}$  until a minimal distance between  $\mathbf{q}$  and  $\bar{\mathbf{q}}$  is reached. The resulting  $\bar{\mathbf{q}}$  is a *mode*  $\mathbf{m}$  of the underlying points distribution and we call  $\mathcal{M}$  the set of points  $\mathbf{q}$  which support  $\mathbf{m}$ . The process is repeated until each point  $\mathbf{q}_{ij}$  has been assigned to a mode  $\mathbf{m}_{i'}$  =  $(\mathbf{p}_{i'}, \delta_{i'})$ . The resulting modes yield *hypotheses* for the location of the token  $\mathcal{T}_k$ . The mode  $\mathbf{m}_{i'}$  defines the token  $\mathcal{T}_{i'}$  with a matching score:

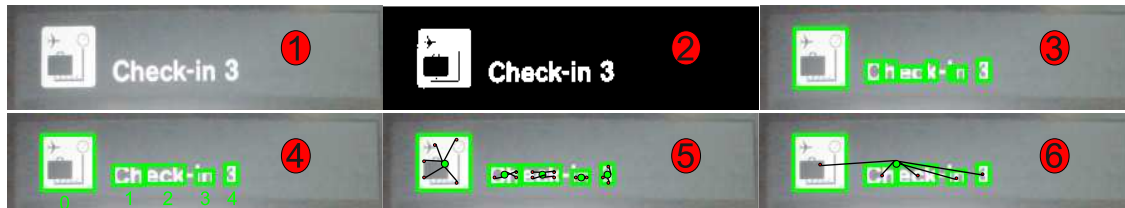


Fig. 1. Template processing. The input image 1 is binarized into image 2 with Otsu’s segmentation method. A region growing algorithm finds clusters of coherent colors in image 2 and outputs them in image 3. An agglomerative clustering algorithm groups the regions of image 3 into tokens in image 4. Image 5 shows the codebooks for each token of image 4. Image 6 shows the codebook for the entire sign from the tokens codebooks of image 5.

$$s(\mathcal{T}_{i'}, \mathcal{T}_k) := \sum_{\forall i,j} w_{ij} \quad (8)$$

Finding the entire sign in  $I$  is then done in a similar fashion as for the tokens. A token  $\mathcal{T}_{i'}$  that matches a token  $\mathcal{T}_k$  of  $\mathcal{C}^S$  casts a vote for an occurrence of the sign  $S$  at the position:

$$\begin{aligned} \mathbf{r}_{i'k} &= \mathbf{p}_{i'} - \mathbf{w}_k \delta_{i'} \\ \zeta_{i'k} &= \delta_{i'} \end{aligned} \quad (9)$$

where  $\mathbf{p}_{i'}$  and  $\delta_{i'}$  represent the hypothesis  $\mathbf{m}_{i'}$  for the position and scale of the token  $\mathcal{T}_{i'}$ , and  $\mathbf{w}_k$  is the distance vector stored in the codebook. Furthermore, each vote has a weight defined by:

$$v_{i'k} := s(\mathcal{T}_{i'}, \mathcal{T}_k) \frac{1}{g_k} \quad (10)$$

As before, we collect the votes  $\mathbf{t}_{i'k} = (\mathbf{r}_{i'k}, \zeta_{i'k})$  that are hypotheses for possible locations of a sign  $S$  in a voting space  $\mathcal{V}$ . Again, we run mean shift mode estimation to find local density maxima in  $\mathcal{V}$ . The resulting location hypotheses  $\mathbf{m}_{j'}^S = (\mathbf{r}_{j'}, \zeta_{j'})$  supported by  $\mathcal{M}_{j'}^S$  for an occurrence of the sign  $S$  in  $I$  have a score defined by:

$$s(\mathbf{m}_{j'}^S) := \sum_{\forall i',k} \mathbf{t}_{i'k} \in \mathcal{M}_{j'}^S v_{i'k} \quad (11)$$

The final score  $s(\mathbf{m}_{j'}^S)$  can be compared to a detection threshold  $\vartheta_d$  in order to validate the presence of the sign  $S$  in  $I$  at location  $\mathbf{r}_{j'}$  and scale  $\zeta_{j'}$ . This threshold influences directly the performance of the sign detector: high values of  $\vartheta_d$  reduce the number of false positive detections and thus increase the precision of the detector. Low values however increase the recall value. A detailed analysis on this is given in Sec. VI. We finally note that our system could be used in a probabilistic framework instead of outputting a binary answer.

## V. MAPPING ACTIONS TO DETECTIONS

We described in the previous sections how HISM represents a reliable sign matching method. In this section, we go

a step further and analyze how it is possible to map actions to sign detections.

For a robot to navigate in a *sign-rich* environment like an airport, we not only need a reliable sign matching algorithm. We also have to associate the instruction related to the sign occurrence to a specific action of the robot, i.e., understand the direction arrows. We present two methods: in one we build an arrow detector, in the other we show how to exploit the full potential of HISM and infer it from the data.

### A. Geometric Approach

A direct way of mapping actions to detections is to explicitly build an arrow detector. A simple arrow detector is described by an heuristic built on a given geometrical model, composed by piecewise linear segments. In order to detect an arrow, the image is segmented with Otsu’s binarization algorithm. Regions with a certain minimal size and uniform color are extracted by using the same connected component approach as for *token* extraction. The following geometrical properties are taken into account for arrow detection:

- Aspect ratio of the bounding box computed on each extracted region.
- Filled ratio, i.e. number of pixels in the clustered region over the number of pixels of the bounding box.
- Horizontal or vertical symmetry axis of the cluster.
- Position of bounding box centroid.

We can efficiently compute these key geometrical properties by using the integral image technique introduced by [4]. The idea is to halve the bounding box by an horizontal line, and then by a vertical line. The integral image allows to easily compute the sum of the pixels in the subdivided bounding box areas, thus the symmetry ratios. We encode 8 types of arrows (up, down, left, right, and their 45-degrees rotated counterparts). We first detect the four cardinal arrows. Then we apply the same technique by rotating of 45-degree the detector for the remaining diagonal arrows. Classification is obtained by empirical thresholding on each of the geometrical property. Henceforth, this method introduces several free parameters to tune, lacks of generality, and is not adaptive.

### B. Learning Approach

By using HISM, we can easily develop a robust and elegant method for mapping a sign to a robot action. The

resulting algorithm is able to deal not only with arrows, but with any kind of geometrical shapes representing a direction, like triangles or complex direction symbols.

The idea is to collect with a robot several pictures of signs, and each time an image is taken, a label is associated. The label represents the high-level action that an operator has commanded to the robot: "turn left", "turn right", "go up", and so on. As soon as this dataset is built, all the images associated with the same action are collected into  $\mathcal{I} = (\mathcal{I}^{up}, \mathcal{I}^{left}, \mathcal{I}^{right}, \mathcal{I}^{down})$ .

The intuition is to find the most recurrent *token* by simply comparing the signs of an image action set together. Thus, we consider each set  $\mathcal{I}^i$  separately and detect all the signs in each set by running our algorithm on that. Therefore, similarly to the procedure of Section III, we produce a frequency analysis. All the *tokens* related to detected signs are discarded, therefore we count how many times each remaining *token* is repeated in  $\mathcal{I}^i$  by computing the usual *token* matching score. In order to avoid ambiguities (e.g. another arrow of another sign in the image) we produce the assumption that the repeated tokens must be in an area close to a detected sign. We then store the *token* as a part of a sign, described by its ISM, related to the direction change  $\mathcal{I}^i$ . This procedure is run in for all  $\mathcal{I}$ .

## VI. EXPERIMENTS

We show in this section the results of our implementation and particularly analyze the qualitative and quantitative performance of our algorithms. Although we do not present an experiment involving a robotic platform, we outline in the conclusion the necessary steps for an integration.

All the experiments are based on a database of images collected at Zurich Airport with a standard digital camera. The original format of the images (JPEG, 3264x2448 pixels) is shrunk to a more reasonable 816x612 pixels BMP format. A gamma correction of 2 is also applied to increase the contrast. The majority of the images contain various signs at different scales, including background scenes of the airport. Some images contain no signs.

As mentioned before, our algorithm relies on several parameters: the descriptors matching distance ratio  $\vartheta_m$ , the clustering threshold in the template image  $\theta_d$ , the minimum area of a region in the template image  $\sigma$ , and the sign matching threshold  $\vartheta_d$ . In the following experiments, we empirically fixed  $\vartheta_m$  to 0.5, and made  $\theta_d$  and  $\sigma$  scale dependent. The discriminative threshold  $\vartheta_d$  was then iterated in order to find the optimal parameter.

### A. Classification

In the first set of experiments we evaluate the general quantitative performance of our classifier in the case of target signs and arrows classification in random sample images.

1) *Signs*: As exposed above, the target signs are firstly matched against each other in order to determine the most distinctive token. In this experiment, we want to match the Check-in 3 sign and we have 6 different target signs

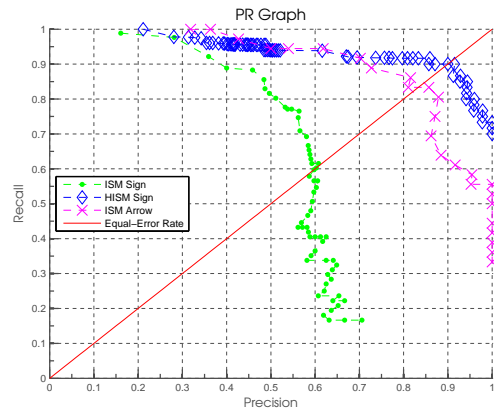


Fig. 2. Precision and Recall graph for signs and arrows classification. The plot shows that HISM outperforms ISM for signs detection, and that the arrows detection performs reliably.

(Check-in 1/2/3, Arrival 1/2, Railway). The token containing the number 3 will get the higher voting weight after the matching process.

The Check-in 3 sign is matched with all the images of the database which is manually labeled to have a ground truth. The database contains 57 images with a Check-in 3 occurrence out of 124 images.

The performance of the classification is evaluated with the Precision and Recall (PR) graph in Fig. 2, which shows the iterative discriminative threshold  $\vartheta_d$  variation. The detection reaches 90 % at the Equal Error Rate (EER). For comparison purpose, Fig. 2 also reports the results using the standard ISM approach, which gets 60.25 % at EER. In this case, a lot of false positives are introduced, since Check-in 1 and Check-in 2 signs are counted as positives. Although we do not report their PR graph, we obtained similar results with the other target signs.

As a consequence of these experiments, we can state that our algorithm is able to correctly label the images belonging to the same path. Moreover, the accuracy of the detection might come to 100 %, if we refine our system by fusing multiple hypothesis of the same spot at different scales.

2) *Arrows*: In this second experiment, we aim at extracting the high-level navigation instructions contained in the signs. In our example of Zurich Airport, these instructions are represented by 8 types of arrows (right, left, up, down, and their 45-degrees rotated counterparts).

23 images containing an arrow pointing to the right are collected. They are matched against each other in order to extract their common token. The descriptors selected in each image are then again matched against each other. The token which gets the highest summed matching score will represent this kind of arrow. The extracted token is finally matched with the entire image collection to assess the performance of the arrow classification. The database contains 35 occurrences of right arrows out of 124 images.

The results are expressed in the PR graph in Fig. 2.

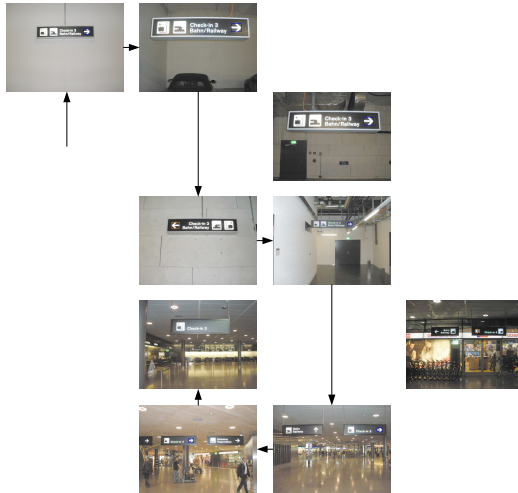


Fig. 3. Example of exploration and mapping in Zurich Airport. The robot starts from the garage and is able to reach the *Check-In 3* by following the instructions on the sign. In the end, a topological map is also built.

The detection reaches 83.33 % at EER. We believe that these results could also be improved by using temporal integration. Although we only show here the classification of right arrows, these results are generalizable to any kind of direction instructions.

For comparison purpose, we also assess the performance of our geometrical arrow detector on the same set of images. We obtain an *accuracy* of 93.54 %, which is slightly better than with the learning method. However, this approach is computationally more expensive and not as easily scalable in other environments.

### B. Exploration and Topological Map Creation

In this experiment, we show how the combination of the previous results can yield an interesting robotic application. Assuming that a robot has an arrow and a sign detector, we can for instance put it in the garage of the airport in front of a direction sign and show him an image of the *Check-in 3* destination sign it has to reach. The robot uses the sign detector to find which part in the current image is relevant to the destination. It then maps the direction instruction to a motor action of its base. The result of this action leads him to the next sign until it reaches the destination. Fig. 3 qualitatively shows the result of this experiment.

This experiment shows that our algorithm is not only suitable for finding its way in an airport. It can also be used with a robot to build a topological map of the airport.

## VII. CONCLUSION

In this paper, we introduced the concept of Hierarchical Implicit Shape Models for robustly recognizing and generalizing over different kinds of signs in unstructured environments. It consists of a two-level hierarchy: one based on image primitives and the other on subparts of signs with

weights learned from data. We showed that with HISM, it is easy to map semantics to detections by learning the meaning of the direction arrows in an unsupervised manner. Experiments have been conducted from datasets retrieved in a crowded airport terminal, that show typical guiding robot applications, with promising results.

As future work, we aim to extend this approach by using temporal integration and to develop learning and detection into a robotic mobile platform ready to be deployed in man-made environments.

### ACKNOWLEDGMENT

This work was funded within the EU Projects BACS-FP6-IST-027140 and EUROPA-FP7-231888.

### REFERENCES

- [1] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *Int. J. Comput. Vis.*, vol. 77, no. 1-3, pp. 259–289, 2008.
- [2] G. Qingji, Y. Yue, and Y. Guoqing, "Detection of public information sign in airport terminal based on multi-scales spatio-temporal vis. information," in *Proc. Int. Conf. Comput. Sci. Softw. Eng.*, 2008.
- [3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
- [4] P. Viola and M. Jones, "Robust real-time face detection," *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.
- [5] X. Baro, S. Escalera, J. Vitria, O. Pujol, and P. Radeva, "Traffic sign recognition using evolutionary adaboost detection and forest-eccoc classification," *IEEE Trans. Intell. Transport. Syst.*, vol. 10, no. 1, pp. 113–126, 2009.
- [6] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofert, and T. Koehled, "A system for traffic sign detection, tracking, and recognition using color, shape, and motion information," in *Proc. IEEE Intell. Veh. Sym.*, 2005.
- [7] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," in *Proc. IEEE Wksh. App. Comput. Vis.*, 2009.
- [8] C. Kiran, L. Prabhu, R. Abdu, and K. Rajeev, "Traffic sign detection and pattern recognition using support vector machine," in *Proc. Int. Conf. Adv. Pattern Recog.*, 2009.
- [9] H. Huang, C. Chen, Y. Jia, and S. Tang, "Automatic detection and recognition of circular road sign," in *Proc. IEEE/ASME Int. Conf. Mechatron. Embedded Sys. App.*, 2008.
- [10] D. Gavrilu and V. Philomin, "Real-time object detection for smart vehicles," in *Proc. IEEE Int. Conf. Comput. Vis.*, 1999.
- [11] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *Proc. of the IEEE Conf. on Comput. Vis. and Pattern Recog.*, 2008.
- [12] H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," in *Proc. Int. Joint Conf. Art. Intell.*, 1977.
- [13] R. B. Fisher and P. Oliver, "Multi-variate cross-correlation and image matching," in *Proc. British Machine Vis. Conf.*, 1995.
- [14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.
- [15] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 24, pp. 509–522, 2002.
- [16] K. Mikolajczyk and C. Schmid, "Scale and affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, 2004.
- [17] Economic Commission for Europe, Inland Transport Committee, "Convention on road signs and signals," 1968.
- [18] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst., Man, Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.
- [19] S. Hojjatoleslami and J. Kittler, "Region growing: A new approach," *IEEE Trans. Image Processing*, vol. 7, no. 7, pp. 1079–1084, 1998.
- [20] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*. Springer, 2006, pp. 25–71.
- [21] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, no. 5, pp. 603–619, 2002.

## Unsupervised Discovery of Repetitive Objects

Jiwon Shin, Rudolph Triebel, and Roland Siegwart  
 Autonomous Systems Lab (ASL)  
 Swiss Federal Institute of Technology, Zurich (ETHZ)  
 {jiwon.shin, rudolph.triebhel, rsiegwart}@mavt.ethz.ch

**Abstract**—We present a novel approach for unsupervised discovery of repetitive objects from 3D point clouds. Our method assumes that objects are geometrically consistent, and uses multiple occurrences of an object as the evidence for its existence. We segment input range data by superpixel segmentation, extract features for each segment, then find a set of segments that have a matching set using a joint compatibility test. The discovered objects are then verified by the Iterative Closest Point algorithm to remove false matches. The presented method was tested on real data of complex objects. The experiments demonstrate that the proposed approach is capable of finding objects that occur multiple times in a scene and distinguish apart those objects of different types.

### I. INTRODUCTION

For a robot that interacts with people, it is essential to semantically analyze its surroundings. In particular, home environments usually contain various objects, which often define the particular location at which they are encountered (e.g. furniture). The ability to detect and distinguish objects autonomously is thus a key for a robots' independence when working in a home environment. For instance, if a robot can determine that a dining room contains a set of chairs, which are multiple occurrences of the same object, and a table, which is different from chairs, then it can use such information to classify a dining room as a place with two types of objects - many chairs and one table. Then, when it encounters an unfamiliar room, it can simply search for the characteristics of the room - many instances of one object type and one instance of a different object type - and the fingerprints of the objects found in the room. When both are verified, the robot can label the room as a dining room. Such an automatic process eliminates the necessity of training a robot with every object it is likely to find in the environment. Instead, we can simply label each type of object a robot finds in the appropriate language of the household, e.g. *chair* or *Sessel*. In this work, we investigate the possibility of unsupervised discovery of objects that occur multiple times, such as chairs in a dining room, from data taken with a 3D laser scanner.

Unsupervised discovery of repetitive objects in a given scene is a challenging task because we do not know a priori the definition of an object, the number of occurrences of a certain object type, nor the number of different object types present in the scene. In addition, the method must be able to distinguish real objects - chairs and couches - from walls, ground, and ceiling as we do not pre-segment them out. The method thus should be able to hypothesize on objects

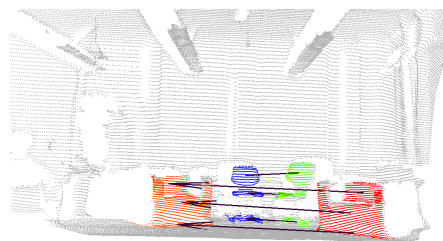


Fig. 1. An example of a scene observed with a nodding SICK laser scanner. Objects that are discovered through the algorithm are colored, where all points which belong to the same object are assigned to one color. Arrows are drawn between two segments that match.

while handling lots of clutter in the scene. As an indoor robot can easily collect more evidence to support or refute a hypothesis without any harm, it is better for a robot to claim no knowledge than have a false belief when the uncertainty is high. To minimize false discovery, we take a conservative approach and only accept the output when the uncertainty is low.

Figure 1 depicts a typical scene of interest in this paper, which is captured using a nodding SICK laser scanner. The scene contains two working chairs and two arm chairs along with some ceiling light fixtures and a plant. Of these, we are interested in discovering the two types of repeating objects - the working chairs and the arm chairs. If the process is successful, each instance of the object gets its own color, and arrows are drawn between all segments that match.

We propose an approach to discover, without supervision, objects that occur multiple times in a scene. Using 3D point clouds from a laser scanner as input, we first segment the points according to their surface property using superpixel segmentation and extract features for each segment. We use an extended joint compatibility test to discover object models and their matching objects, and verify these objects by the Iterative Closest Point algorithm to remove false matches. Through this work, we demonstrate that repetition can aid the discovery of objects and define object models.

The organization of the paper is as follows. We discuss related work in Section II. Section III explains how the input scene is segmented and how features are extracted from each segment. In Section IV, we discuss the object discovery method and the verification step. Section V presents the experimental results. The paper concludes with Section VI.

## II. RELATED WORK

Repetition detection has been well-explored in the field of image analysis. In particular, many authors have investigated methods for detecting regularly repeating patterns [1], [2]. More recently, Loy and Eklundh [3] focused on grouping of features based on symmetry, and Wenzel *et al.* [4] proposed an algorithm that uses symmetry to detect repetitive structures in facade images. They argued that symmetry is a strong clue to group features together. Likewise, we group together segments upon a discovery of a matching set, but we do not explicitly search for the symmetric plane between the two objects. In this way, our approach is similar to Zeng and van Gool [5], where the authors employ point-wise repetition to improve segmentation results. They use mutual information to determine if two segments of an initially oversegmented image are of the same group. We instead extract features for every segment and compare these features to measure the similarity between two segments.

In terms of 3D, discovery and utilization of repetition has been addressed in computer aided design and other synthetic models [6], [7], [8]. They focused on detection of symmetry or regular patterns in 3D with applications in graphics and image compression. Work of Bokeloh *et al.* [9] is more closely related to this work. The authors proposed an algorithm for detecting structural redundancy by matching symmetric constellations of feature lines. We also search for a collection of elements that repeat as a group, but we do not assume symmetry as the repetition pattern. To our knowledge, no work has dealt with discovery of objects by repetition in laser data.

In unsupervised object detection, several have proposed adaptation of text analysis methods in image analysis. For example, Liu and Chen [10] has proposed a modified probabilistic latent semantic analysis method to detect foreground objects from images. In [11], Endres *et al.* use Latent Dirichlet Allocation to detect object classes from range data without supervision. While this approach can classify objects of multiple classes, they assume that a ground plane and walls are extracted a priori and the objects are spatially disconnected. In our work, we do not make such assumptions. We consider every segment as a potential object part and test them to determine if they belong to an object.

The way we define an object is parts-based. We search for objects using the joint compatibility branch-and-bound algorithm [12]. Shin *et al.* [13] has shown that objects defined by parts can be represented by a grammar and recognized using a joint compatibility test. In our work, we do not perform a separate parts detection, nor require object parts to have physical meanings.

We employ feature-based approaches to recognize objects. Among various feature descriptors for 3D data, spin images have been shown to be successful and popular [14], [15]. Other features of interest for this work are shape distribution [16] and shape factors [17].

## III. SEGMENTATION AND FEATURE EXTRACTION

The proposed algorithm is a three-step process. First, we extract segments from the input point cloud and extract features for every segment. We apply a joint compatibility test on these segments to detect objects and then verify them using the Iterative Closest Point algorithm. In this section, we describe the segmentation method and shape descriptors.

### A. Range Data Segmentation

The goal of segmentation is to find labels  $L(\mathbf{x})$  for all data points  $\mathbf{x}$ , where points that are close to each other and similar in some predefined way, should have the same label. We use the superpixel segmentation method by Felzenszwalb and Huttenlocher [18], originally proposed for 2D images, to group together similar points. This algorithm creates a graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$  of vertices  $\mathcal{V}$  and edges  $\mathcal{E}$ , where each pixel in a given image corresponds to a vertex and the edges connect adjacent image pixels. Each edge  $\mathbf{e} = (v_i, v_j)$  has an associated weight  $w(\mathbf{e})$  representing the dissimilarity of the connected vertices  $v_i$  and  $v_j$ . In the case of an image, this can be, for example, the difference of the pixel intensities. The algorithm starts with a segmentation where each vertex is its own segment. Then, the edges are processed by increasing weights and the two segments  $C_i$  and  $C_j$  connected by a given edge  $\mathbf{e}$  are merged whenever

$$w(\mathbf{e}) \leq \min\left(d(C_i) + \frac{k}{|C_i|}, d(C_j) + \frac{k}{|C_j|}\right),$$

where  $d(C)$  is the *internal difference* function defined by the maximal edge weight of all edges in the minimum spanning tree of the segment  $C \subseteq \mathcal{V}$ , and  $k$  is a consistency parameter that influences the granularity of the segmentation: a low value of  $k$  requires segments to be more consistent and thus produces more but smaller segments. The internal difference function ensures that two segments are merged only when the difference between the two is smaller than the difference within each segment with some tolerance.

In this work, we define each point  $\mathbf{x}$  of a 3D point cloud  $\mathcal{X}$  as a vertex and form an edge between two neighboring vertices, where neighbors are determined by a triangular mesh built on the data. We use the dot product  $\mathbf{n}_i \cdot \mathbf{n}_j$  as edge weight where  $\mathbf{n}_i$  is the surface normal vector computed at point  $\mathbf{x}_i$ . Thus, regions with a smooth surface, e.g. a plane or a sphere, are segmented as one region while surfaces with sharp edges, e.g. between two sides of a box, are segmented into two regions. As a modification of the original algorithm, we do not force every point to be in a segment. This is because we cannot calculate the normal for the points with an insufficient number of neighboring points. For these isolated points, no vertices are generated in the graph, and thus no label is assigned. In addition, after termination we remove segments that contain less points than a given minimal value  $m_{size}$ . Such small segments are often caused by sensor imperfections or occlusions and do not reveal enough information for the later matching process.

### B. Shape Descriptors

As shape descriptors, we use spin images [14], shape distributions [16], and shape factors [17], and weigh them accordingly. For a given point  $\mathbf{x}$  with normal vector  $\mathbf{n}$ , a *spin image* is defined as a 2D histogram  $H^s$  oriented along the line  $l$  through  $\mathbf{x}$  with direction  $\mathbf{n}$ . Each bin of  $H^s$  counts the points with a certain distance to  $l$  and the plane through  $\mathbf{x}$  with normal vector  $\mathbf{n}$ . For the spin image descriptor of a segment  $C$ , we form vectors  $\mathbf{h}_i^s$  of stacked lines of the histograms  $H_i^s$  for all points  $\mathbf{x}_i \in C$  and compute the average  $\bar{\mathbf{h}}^s$  over all  $\mathbf{h}_i^s$ .

A *shape distribution* is defined as a histogram of values of a predefined function  $f : C^r \rightarrow \mathbb{R}$ , where  $r$  is the arity of  $\mathbf{f}$  and is usually a value between 1 and 4. In our implementation, we use two binary functions  $f_d(\mathbf{x}_i, \mathbf{x}_j)$  and  $f_a(\mathbf{x}_i, \mathbf{x}_j)$ , namely the Euclidean distance of the points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  and the scalar product of their normal vectors  $\mathbf{n}_i$  and  $\mathbf{n}_j$ . The resulting histogram vectors  $\mathbf{h}^d$  and  $\mathbf{h}^a$  are computed by evaluating  $f_d$  and  $f_a$  on all pairs of points in a segment  $C$ . To make the feature vectors invariant with respect to the sample density, we normalize the histograms  $\mathbf{h}^d$  and  $\mathbf{h}^a$  by the total number of bin entries. A normalization with respect to the maximum distance encountered in a segment is not done, as this would result in scale-invariant features, and we consider scale as a feature to be distinguished between objects.

Finally, we compute *shape factors* per segment, i.e. the normalized eigenvalues of the covariance matrix  $C_i$  of all points in segment  $C_i$ , collected in a vector  $\mathbf{h}^f$ . All individual descriptors are used to define a distance metric  $d_c$  on segments as

$$d_c(C_i, C_j) = \lambda_1 \Delta \bar{h}^s + \lambda_2 \Delta h^d + \lambda_3 \Delta h^a + \lambda_4 \Delta h^f,$$

where the  $\lambda_i$  are weight factors and  $\Delta h$  is the Euclidean distance between two feature vectors  $\mathbf{h}(C_i)$  and  $\mathbf{h}(C_j)$ .

## IV. OBJECT DISCOVERY

The challenge of unsupervised discovery of repetitive objects is that we have neither an a-priori definition of an object, nor the number of occurrences per object type. Without such information, we cannot determine for each segment if the segment is an instance of an object. To overcome this problem, we search for only those objects that occur multiple times in the scene. The multiplicity allows us to reason on the object by comparing it against another instance of the same object. In addition, we only focus on complex objects and define an object as a collection of segments. Discovering objects composed of only one segment requires us to rely entirely on the shape descriptors for matching. The minimum segment constraint allows us to use physical constraints as an additional evidence for an object. Therefore, we consider an object hypothesis valid only when it is composed of at least two segments. To reduce false matches, we verify the hypotheses for objects by finding correspondences between the point clouds of discovered objects.

---

**Algorithm 1:** JCBB The joint compatibility branch-and-bound test for discovering a pair of repetitive objects.

---

**Data:** Segments  $\mathbb{C}$  of the scene

**Input:** Current *model* hypothesis  $\mathcal{H}_M \subset \mathbb{C}$  and its matching *test* hypothesis  $\mathcal{H}_T \subset \mathbb{C}$

**Output:** A pair of hypotheses  $\mathcal{B}_M \subset \mathbb{C}$  and  $\mathcal{B}_T \subset \mathbb{C}$  that yield the highest matching score.

**Procedure:**

**if**  $|\mathcal{H}_M| \geq |\mathcal{B}_M|$  **and**  $d_c(\mathcal{H}_M, \mathcal{H}_T) > d_c(\mathcal{B}_M, \mathcal{B}_T)$  **then**

$\mathcal{B}_M \leftarrow \mathcal{H}_M$  ;

$\mathcal{B}_T \leftarrow \mathcal{H}_T$  ;

**end**

$I \leftarrow |\mathbb{C}|$  ;

**for**  $i = 1$  **to**  $I$  **do**

$C_M^i \leftarrow \text{random\_select\_from}(\mathbb{C})$  ;

$C_T^i \leftarrow \text{random\_select\_from}(\mathbb{C} \setminus \{C_M^i\})$  ;

**if** *individual\_match*( $C_M^i, C_T^i$ ) **and**

*relation\_match*( $\mathcal{H}_M \cup C_M^i, \mathcal{H}_T \cup C_T^i$ ) **then**

            JCBB ( $\mathcal{H}_M \cup C_M^i, \mathcal{H}_T \cup C_T^i, \mathbb{C} \setminus \{C_M^i, C_T^i\}$ ) ;

**end**

**end**

---

### A. Repetitive Object Discovery

To find repeating objects, we use a joint compatibility test with branch-and-bound [12], a popular solution for data association problems. Data association is a well-known problem in robotics. The joint compatibility test addresses the data association problem by finding test points that not only correspond to the model points individually but also match well as a set. The branch-and-bound aspect enables the algorithm to search efficiently by growing a hypothesis when necessary and terminating one when no appropriate part is found.

In an ordinary data association problem, the model set is predetermined, and the goal is to find the best mapping from the test set to the model set. In our framework, however, we do not have a model. Our goal is to discover a model through the detection of matching pairs of segments. We thus propose a modification to the algorithm that is capable of extracting an object model and its matching test object from the input segments.

The overall algorithm is shown in Algorithm 1. Given segments as input to the algorithm, we search for a set of segments that occur multiple times in the scene. We perform the search in two steps. In the first step, we discover an object model, i.e. a collection of segments, and its matching object. Since the only evidence we have for an object is the presence of a matching object, the process will always return two hypotheses. In the second step, the algorithm searches for the remaining occurrences of the object using as a model, the objects found in the first step.

The first step is as follows: We begin with a randomly selected segment  $C_M^0$  and look for a segment  $C_T^0$  in the scene



that match well with it. If we find  $C_T^0$ , then we begin two hypotheses,  $\mathcal{H}_M$  and  $\mathcal{H}_T$ , one for the *model* set and the other for the *test* set. The distinction of *model* and *test* hypotheses is arbitrary as one hypothesis is only valid with the existence of a matching hypothesis. Therefore, there is no definitive model hypothesis to which a test hypothesis must match. Rather, a pair of segments must be similar enough to support each other's validity.

The hypotheses  $\mathcal{H}_M$  and  $\mathcal{H}_T$  grow as we select a new segment  $C_M^1$  and search for  $C_T^1$  that is individually compatible to  $C_M^1$ , and its combination with  $\mathcal{H}_T$  is jointly compatible to  $\mathcal{H}_M \cup C_M^1$ . The hypotheses  $\mathcal{H}_M$  and  $\mathcal{H}_T$  continue to grow until the  $n$ -th model segment  $C_M^n$  no longer finds a compatible test segment  $C_T^n$ . The algorithm then starts a new pair of hypotheses with a different seed segment pairs, in search of the best pair of hypotheses  $\mathcal{B}_M$  and  $\mathcal{B}_T$ . The best pair of hypotheses contains the most number of segments with the smallest distance between the hypotheses. At the end of the process, we label  $\mathcal{B}_M$  and  $\mathcal{B}_T$  as objects  $O_1$  and  $O_2$  of type  $\mathcal{D}$ .

Upon the discovery of an object type  $\mathcal{D}$ , we begin the second step. To find the remaining instances of  $\mathcal{D}$  in the scene, we apply the algorithm again, but this time, using  $O_1$  and  $O_2$  as the model. Now the goal is to find a set of segments that matches the model best. Each time we find such a hypothesis  $\mathcal{H}_k$ , we label it as an object  $O_k$  of the type  $\mathcal{D}$ . The search for an object of type  $\mathcal{D}$  ends when we no longer find a hypothesis that matches either  $O_1$  or  $O_2$ . We repeat this two-step process of finding a pair of hypotheses and detecting other instances of the object until we no longer find a valid hypothesis.

In the presented algorithm, the individual and the joint match score play a crucial role in deciding on a match. We use the shape descriptors as described in Section III-B to evaluate a match. For an individual match, we consider a pair of segments  $C_M$  and  $C_T$  compatible if

$$d_c(C_M, C_T) < T_i,$$

where  $T_i$  is a thresholding value for individual compatibility. For the joint compatibility, in addition to calculating  $d_c(\mathcal{H}_M \cup C_M, \mathcal{H}_T \cup C_T)$ , we compute the Mahalanobis distance  $d_m(\mathcal{V}^{C, \mathcal{H}_M}, \mathcal{V}^{C, \mathcal{H}_T})$  between the new segment pairs to the segments in their corresponding hypotheses, where  $\mathcal{V}^{C, \mathcal{H}_M}$  indicates a vector from the center of the input segment  $C_M$  to a segment in the hypothesis  $\mathcal{H}_M$ . We require that for all segments in  $\mathcal{H}_M$  and  $\mathcal{H}_T$ ,

$$d_m(\mathcal{V}^{C, \mathcal{H}_M}, \mathcal{V}^{C, \mathcal{H}_T}) < T_j.$$

The physical constraints enable us to reject segments that are similar in features but are inconsistent with the hypotheses in their arrangement.

### B. Match Verification

The goal of the verification step is to minimize falsely discovered objects by confirming that the discovered objects are consistent among themselves. We achieve this by mapping all points of an object  $O_i$  to the points that belong to its matching

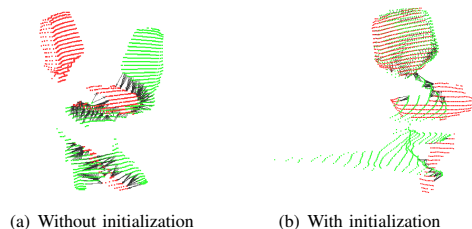


Fig. 2. Correspondences between two point clouds without the spin image initialization 2(a) and with the initialization 2(b). Without the initialization, ICP performs poorly when the two point clouds have a high rotational transformation as shown in 2(a). 2(b) shows that the same two sets match well with the initialization.

pair  $O_j$  by the Iterative Closest Point (ICP) algorithm [19]. ICP, often used in localization, finds the transformation from one point cloud to the other by minimizing the difference between the two sets. Since ICP finds a local minimum, it works well only when the initial correspondence between two point clouds is close to the global minimum. As the objects  $O_i$  and  $O_j$  can be in any orientation, the initial estimation cannot rely purely on the nearest neighbors in the Euclidean space. We instead estimate the initial transformation by computing features at various randomly selected points in  $O_i$  and finding their corresponding points from  $O_j$  in the feature space. We use spin images as the features, as presented in [14]. Figure 2 shows the effect of the initialization by the feature-space correspondence. As the figure indicates, without the initialization, the verification step performs poorly when the objects are mirrored.

The initialization is as follows: Given two objects  $O_i$  and  $O_j$ , we first center them with their respective mean values  $\bar{O}_i$  and  $\bar{O}_j$  in x- and y-direction, and randomly select a subset of points  $(\mathbf{x}_1, \dots, \mathbf{x}_n)$  in  $O_i$ . We assume that objects are in their natural vertical position and do not center the points in z-direction. This helps us eliminate wall-ceiling, wall-floor, and ceiling-floor matches. For each point  $\mathbf{x}_k$ , we calculate its spin image and search for all points  $(\mathbf{y}_1^k, \dots, \mathbf{y}_m^k)$  in  $O_j$ , whose spin image is similar to  $\mathbf{x}_k$ . These points are then used as the initial correspondence points for ICP. Once the transformation between the two point clouds is found, we count all the points in  $O_i$  that have a corresponding point in  $O_j$  and vice versa. We consider the two objects  $O_i$  and  $O_j$  matched if the total number of matched points is greater than 70 percent of the sum of points in  $O_i$  and  $O_j$ .

## V. RESULTS

In this section, we test the algorithm on scans from real world scenes. We took data using a nodding SICK laser with a width of 100 degrees and a height of 90 degrees. Each set was captured at the horizontal resolution of 0.25 degrees and the vertical resolution of 15 degrees a second. The test set was composed of 55 data sets from four different rooms. Overall, the scenes had four types of working chairs and one type of arm chairs along with trash cans, a flip chart, and a plant as background. Objects were placed up to 90 degrees

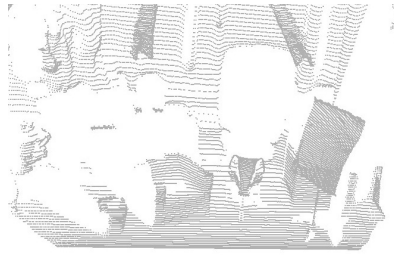


Fig. 4. A test scene with no repeating object. The algorithm discovers no object.

TABLE I  
THE EFFECT OF SEGMENTATION PARAMETERS ON THE OBJECT DISCOVERY RATE

$m_{size}$	50	75	100	120	150
$k = 6$	56%	48%	51%	45%	40%
$k = 9$	59%	48%	59%	45%	40%
$k = 12$	42%	34%	36%	36%	36%
$k = 15$	47%	47%	36%	36%	36%

of rotation from each other. Most scenes contained two or three objects of the same type, but some scenes contained two objects of two kinds. Four scenes contained no repeating objects. In total, there were 138 instances of objects that the algorithm could discover.

We evaluate the algorithm by the rate of discovery and precision. The discovery rate is the number of objects the algorithm found over the number of objects we expect it to discover. We calculate precision as the number of correctly discovered objects over the number of correctly and incorrectly discovered objects. The rates for object types are computed likewise. For example, if a scene contains three chairs of type A and one of type B, then we define the ground truth as three chairs and one object type. As mentioned earlier, the program does not detect objects of single occurrence.

Figure 3 and Figure 4 contain some of the results of the presented algorithm. All points that belong to the same object have the same color, and an arrow connects two matching segments. The arrow starts from a *model* segment and points at the corresponding *test* segment. The overall rate of object discovery is 59% and that of object types is 68%. The precision is 98% for objects and 97% for object types. The precision is high because our method eliminates every uncertain object. In a home environment, it is better for a robot to take more data when it is uncertain about the environment than to make a false assumption about its surroundings. If we set the minimum segment requirement to one, i.e. an object is composed of one or more segments, then the discovery rate goes up to 76% for objects and 84% for object types, but the precision drops to 51% for objects and 49% for object types. The drastic decrease of precision is due to the false matches among segments that belong to wall, ceiling, and floor.

Our method does not assume a perfect segmentation.

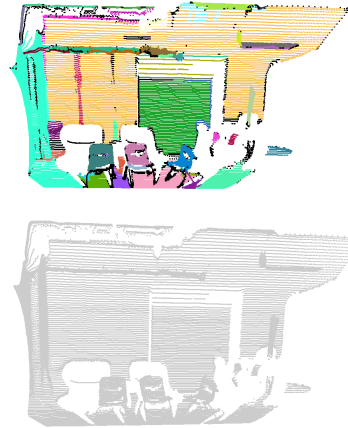


Fig. 5. No object is discovered due to sufficient number of segments. When an object is segmented as one segment (top), the program fails to discover it as an object (bottom).

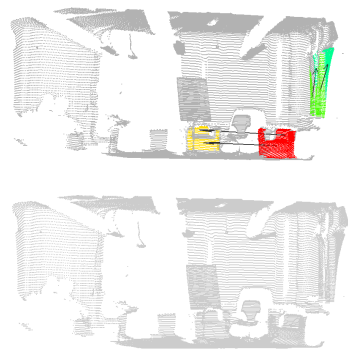


Fig. 6. No object is discovered because of a failure in the verification step. Objects that are found in the discovery phase (top) are falsely eliminated during the verification phase, yielding no detection (bottom).

However, the final outcome is affected by the quality of segmentation. Table I shows the rate of discovery against the consistency parameter  $k$  and the minimum segment size parameter  $m_{size}$ . Our experiment revealed that  $k = 9$  and  $m_{size} = 100$  yields the highest discovery rate and precision. This is partially due to our assumption that an object is composed of at least two segments. The requirement naturally favors objects that are segmented into multiple segments. Therefore, for a high discovery rate without suffering the precision, it is crucial that the segmentation is done in such a way to allow multiple parts per object while each segment being large enough to be discriminative. One major source of no discovery was the lack of sufficient object segments. When an object is segmented into a single segment, the program fails to discover the object as it is invalid according to our definition of object, as shown in Figure 5. Another source of no detection was the lack of sufficient points on

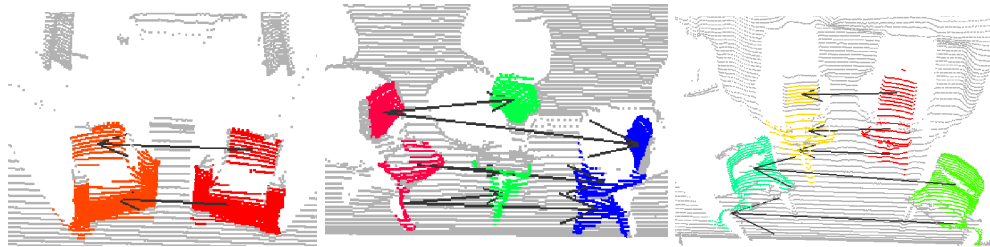


Fig. 3. Some test scenes with discovered objects in color. Points that belong to the same object have the same color

objects. In addition to occlusion and the natural sparsity of data, the incident angle limits the number of usable points in laser data. Without sufficient points, however, we cannot extract segments and features reliably. This causes a lack of object discovery. Lastly, while the verification step most often improved the quality of results, it sometimes eliminated correct hypotheses due to incorrect initialization of point clouds, shown in Figure 6.

#### VI. CONCLUSION AND OUTLOOK

We presented an approach for unsupervised discovery of repeated objects in range data without a prior knowledge on parts, location, or the number of occurrences. It determines potential object parts by applying a modified superpixel segmentation on the point cloud and extracts features on these segments using spin images, shape distributions, and shape factors. It then discovers objects by finding a set of segments that has a matching set using a joint compatibility test. The objects are verified by the Iterative Closest Point algorithm to minimize false matches. We tested the algorithm on real world data sets to demonstrate its ability to detect repeated objects. The whole process is performed without any supervision and without presegmentation of the background.

There are several avenues for improvement. Work presented in this paper has so far only been tested indoor. While outdoor also contains repetitive structure, using the current algorithm for outdoor scenes poses challenges because outdoor objects are often much bigger than indoor objects. A single scan of an outdoor scene often fails to capture multiple instances of the same object at the level of detail necessary for the algorithm. To overcome this problem, it is necessary to merge several images together to obtain more dense data. Such utilization of a robot's mobility would also improve the indoor results as some objects were undiscovered due to an insufficient number of points on the object. The ultimate goal is to enable a robot to learn the characteristics of a place, which requires to extend the approach as to find matches among several places of the same type.

#### VII. ACKNOWLEDGMENTS

This work was partially supported by the EC in the project robots@home under contract number FP6-IST-045350, and in the project BACS under contract number FP6-IST-027140.

#### REFERENCES

- [1] T. Tuytelaars, A. Turina, and L. V. Gool, "Noncombinatorial detection of regular repetitions under perspective skew," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 418–432, 2003.
- [2] Y. Liu, R. T. Collins, and Y. Tsin, "A computational model for periodic pattern perception based on frieze and wallpaper groups," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 354–371, 2004.
- [3] G. Loy and J. olof Eklundh, "Detecting symmetry and symmetric constellations of features," in *ECCV*, 2006, pp. 508–521.
- [4] S. Wenzel, M. Drauschke, and W. Förstner, "Detection of repeated structures in facade images," *Pattern Recognition and Image Analysis*, vol. 18, no. 3, pp. 406–411, 2008.
- [5] G. Zeng and L. V. Gool, "Multi-label image segmentation via point-wise repetition," in *CVPR*, Alaska, USA, June 2008.
- [6] D. Shikhare, S. Bhakar, and S. Mudur, "Compression of large 3d engineering models using automatic discovery of repeating geometric features," in *Vision, Modeling, and Visualization*, 2001.
- [7] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A planar-reflective symmetry transform for 3d shapes," in *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*. New York, NY, USA: ACM, 2006, pp. 549–559.
- [8] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas, "Discovering structural regularity in 3d geometry," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–11, 2008.
- [9] M. Bokeloh, A. Berner, M. Wand, H. Seidel, and A. Schilling, "Symmetry detection using feature lines," *Computer Graphics Forum (Proc. of Eurographics)*, vol. 28, no. 2, pp. 697–706(10), April 2009.
- [10] D. Liu and T. Chen, "Semantic-shift for unsupervised object detection," in *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*. Washington, DC, USA: IEEE Computer Society, 2006, p. 16.
- [11] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard, "Unsupervised discovery of object classes from range data using latent dirichlet allocation," in *Proc. of Robotics: Science and Systems*, 2009.
- [12] J. Neira and J. D. Tardos, "Data association in stochastic mapping using the jointcompatibility test," *IEEE Trans. Robotics and Automation*, vol. 17, no. 6, pp. 890–897, 2001.
- [13] J. Shin, S. Gächter, A. Harati, C. Pradelier, and R. Siegwart, "Object classification based on a geometric grammar with a range camera," in *IEEE Int. Conf. Robotics and Automation*, 2009.
- [14] A. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 1997.
- [15] R. Triebel, R. Schmidt, O. M. Mozos, and W. Burgard, "Instance-based amn classification for improved object recognition in 2d and 3d laser range data," in *Proc. of the Intern. Joint Conf. on Artif. Intell.*, 2007.
- [16] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.
- [17] C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz, "Geometrical diffusion measures for MRI from tensor basis analysis," in *ISMRM '97*, Vancouver Canada, April 1997, p. 1742.
- [18] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.
- [19] Z. Zhang, "Iterative point matching for registration of free-form curves

# Segmentation and Unsupervised Part-based Discovery of Repetitive Objects

Rudolph Triebel    Jiwon Shin    Roland Siegwart  
 Autonomous Systems Lab, ETH Zurich  
 Tannenstrasse 3, 8092 Zurich, Switzerland  
 email: {rudolph.triebel,jiwon.shin}@mavt.ethz.ch, rsiegwart@ethz.ch

**Abstract**— In this paper, we present an unsupervised technique to segment and detect objects in indoor environments. The main idea of this work is to identify object instances whenever there is evidence for at least one other occurrence of an object of the same kind. In contrast to former approaches, we do not assume any given segmentation of the data, but instead estimate the segmentation and the existence of object instances concurrently. We apply graph-based clustering in feature and in geometric space to presegmented input data. Each segment is treated as a potential object part, and the inter-dependence of object labels assigned to part clusters are modeled using a Conditional Random Field (CRF) named the “parts graph”. Another CRF is then applied to the scene graph to smooth the class labels using the distributions obtained from the parts graph. First results on indoor 3D laser range data are evaluated and presented.

## I. INTRODUCTION

The ability for a robot to learn and discover objects without any human guidance enhances its autonomy and makes it more independent. Such a robot requires no prior training and can more easily adapt to new, unknown environments. It is also able to autonomously draw conclusions about the structure of its environment. This functionality is useful when robots operate fully autonomously without human interaction. But also when robots live with humans, a high-level semantic analysis of the environment helps the robot to communicate with a human. As an example, a robot which can detect similarities of objects encountered in the environment, no longer requires a human to first label all occurrences of objects in a previously acquired data set. Instead, it may ask the human, “I discovered several instances of something that looks like an interesting object. What is the name of the object?” In this paper, we take a first step in this direction.

We propose an approach to segment and discover objects of multiple occurrences without supervision, where an object is defined as a constellation of object parts. We segment input point clouds and treat each segment as an instance of a potential object part. In our work, object parts are determined by grouping similar segments together using clustering in a predefined feature space. In addition, the segments are clustered in the geometric space and the number of resulting connected components is used as an upper bound on the number of potential object classes. Then, two major reasonings are used to determine a class label for each segment: First, different object parts that often occur close to each other are more likely to correspond to the same object class. For

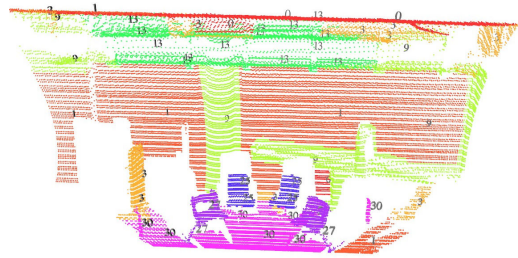


Fig. 1: 3D range scan of an indoor scene. Two different types of chairs are detected by exploiting the fact that particular constellations between back rests and seats occur more than once in the scene.

example, the fact that a back rest of a chair and a chair seat are frequently observed in close vicinity to each other rises the evidence that there is an object class for which more than one instance appears in the scene (see Fig. 1). Second, an instance of an object part may appear to correspond to some object class, but given its physical context it is more likely to be part of another class. For example, a segment may appear to be a chair leg, but if surrounded by table parts it is more likely to be a part of a table. Both ideas are implemented using probabilistic reasoning based on Conditional Random Fields (CRFs) [1].

## II. RELATED WORK

Most work on repetition detection has been in the field of image analysis. Detection of regularly repeating patterns has been the focus of many researchers [2, 3] with some recent work by Loy and Eklundh [4] on grouping of features based on symmetry, and by Wenzel *et al.* [5] on using symmetry to detect repetitive structures in facade images. Zeng and van Gool [6] employ point-wise repetition to improve segmentation results using mutual information. In 3D, discovery and utilization of repetition has been addressed in computer aided design and other synthetic models [7, 8, 9]. The work of Bokeloh *et al.* [10] is more closely related to this work. The authors proposed an algorithm for detecting structural redundancy by matching symmetric constellations of feature lines. In terms of repetition detection, the main challenge of our work lies in the lack of a repetition unit as we do not assume any regularity or symmetry of the repetition pattern.

In this work, we employ clustering to group similar segments in feature and geometric space. Clustering has received a considerable amount of attention by machine learning and pattern recognition communities. Some classic methods such as the expectation-maximization algorithm and  $k$ -means clustering assume that data can be modeled by a simple distribution, while other methods such as agglomerative clustering are sensitive to noise and outliers. To overcome these challenges, alternative approaches have been proposed. Ng, Jordan, and Weiss [11] presented a spectral clustering algorithm, which uses eigenvectors of the data matrix to group points together, and demonstrate how well the algorithm clusters even challenging data. Another work of interest is affinity propagation proposed by Frey and Dueck [12]. Affinity propagation clusters data by finding a subset of exemplars, which are cluster centers selected from data. This method avoids the pitfalls of bad initialization and does not require the number of clusters to be prespecified. In this work, we use affinity propagation to cluster segments in feature space.

Conditional Random Fields (CRFs) [1] are discriminative models, that have also been applied to object recognition problems [13, 14]. Notably, Quattoni, Collins, and Darrell [15] presented a part-based approach for object class recognition using a CRF. We also take a parts-based approach for objects. Ma and Grimson [16] proposed a coupled CRF to allow for interaction between contour and texture in image data. While our work does not explicitly use coupled CRFs, the interaction between part labels and class labels play a critical role in the success. The main idea in our work which has not been addressed previously is the use of Conditional Random Fields without any training set. Instead, we infer from clustering results the possible object labels.

In unsupervised object detection, several authors have proposed adaptation of text analysis methods in image analysis. For example, Liu and Chen [17] proposed a modified probabilistic latent semantic analysis (pLSA) method to detect foreground objects from images. Sivic *et al.* [18] compare pLSA and Latent Dirichlet Allocation (LDA) to discover object categories from sets of images. Also, Endres *et al.* [19], use LDA to discover object classes from range data without supervision. While this approach can classify objects of multiple classes, it assumes that a ground plane and walls are extracted a priori and the objects are spatially disconnected. In our work, we do not make such assumptions. We consider every segment as a potential object part and test them to determine if they belong to an object. Lastly, this work is similar to our previous work [20], which also discovers objects without supervision, but it does not explicitly label the segments as we do in this work.

### III. SEGMENTATION AND CLUSTERING

Object detection using unsupervised learning is significantly different from using supervised learning. In fact, objects can not be *detected*; that is, it is not possible to identify some part of the input data by matching it against an instance of a previously known object class as there is no such known object

#### Algorithm: SSCGF

**data** : Point Cloud  $\mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_N\}$

**input** :

- Segmentation parameters  $\kappa$  and  $\tau$
- Cluster parameters  $\vartheta_f$  and  $\vartheta_g$

**output**:

- low-level segmentation  $\mathcal{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_M\}$ ,  $\mathbf{s}_i \subset \mathcal{P}$
- feature-space clusters of segments  $\mathcal{F}_1, \dots, \mathcal{F}_C$
- geometric clusters of segments  $\mathcal{G}_1, \dots, \mathcal{G}_K$
- class label distributions  $\mathbf{d}_1, \dots, \mathbf{d}_M$ ,  $\mathbf{d}_i \in [0, 1]^K$

**procedure**:

```

 $\mathcal{S} \leftarrow \text{SuperPixelSegmentation}(\mathcal{P}, \kappa, \tau)$ 
 $\mathbf{f}_1, \dots, \mathbf{f}_M \leftarrow \text{FeatureExtraction}(\mathcal{S})$ 
 $\mathcal{F}_1, \dots, \mathcal{F}_C \leftarrow \text{ClusterInFeatureSpace}(\mathcal{S}, \vartheta_f, \{\mathbf{f}_i\})$ 
 $\mathcal{G}_1, \dots, \mathcal{G}_K \leftarrow \text{ClusterInGeometricSpace}(\mathcal{S}, \vartheta_g)$ 
 $\mathbb{P} \leftarrow \text{MakePartsGraph}(\{\mathcal{F}_i\}, \{\mathcal{G}_i\}, \mathcal{S})$ 
 $\mathbb{P} \leftarrow \text{SmoothPartsGraph}(\mathbb{P}, K)$ 
 $\mathbb{S} \leftarrow \text{MakeSceneGraph}(\mathbb{P}, \{\mathcal{G}_i\}, \mathcal{S})$ 
 $\mathbb{S} \leftarrow \text{SmoothSceneGraph}(\mathbb{S}, K)$ 
 $\mathbf{d}_1, \dots, \mathbf{d}_M \leftarrow \text{ReadFromGraphNode}(\mathbb{S})$ 

```

Alg. 1: Segmentation and smoothed clustering in geometric and in feature space (SSCGF). Note that the number of segments  $M$ , as well as the numbers  $C$  and  $K$  of clusters are computed inside the particular subroutines (see text).

class. Instead, objects can only be *discovered* by hypothesizing the existence of an object based on some sort of repetition or pattern found in the data. Thus, to discover objects, we need to find similarities in the data. We do this by extracting features (see Sec. III-B) and comparing them by a clustering algorithm in the feature space, which is described in Sec. III-C.

Another problem which arises here is that the *segmentation* of the data is unknown, i.e. we do not know where the boundaries of the objects are. The segmentation problem is tightly bound to the detection problem because a perfect segmentation would make object detection very easy – a simple comparison of the segmented object instances would suffice. To tackle this problem we perform three steps in our algorithm: first, we apply a low-level segmentation as described in Sec. III-A. Then, we obtain a coarser data segmentation by clustering in the geometric space as presented in Sec. III-C. Finally, we obtain a further improved segmentation by reasoning on parts of objects that occur in a similar constellation in different instances of the scene. The details of this are described in Sec. IV and Sec. V. An overview of the entire algorithm is shown in Alg. 1.

#### A. Low-level Segmentation

The first step in our algorithm is the segmentation of the data using the graph-based segmentation algorithm of Felzenszwalb and Huttenlocher [21], adapted to range image data. In the modified algorithm, we create a graph  $\mathbb{G} = \{\mathcal{V}, \mathcal{E}\}$  of vertices  $\mathcal{V}$  and edges  $\mathcal{E}$ , where each point  $\mathbf{p}$  in a given point cloud  $\mathcal{P}$

corresponds to a vertex and an edge connects adjacent points. Here, adjacency of two points is determined from a triangular mesh built on the point cloud. Every edge  $(\mathbf{p}_i, \mathbf{p}_j)$  has an associated weight  $w_{ij}$ , which is equal to the dissimilarity  $\Delta$  of the connected points  $\mathbf{p}_i$  and  $\mathbf{p}_j$ . In the case of a camera image, this can be the difference of the pixel intensities; in our case, we define  $\Delta(\mathbf{p}_i, \mathbf{p}_j)$  as the dot product of the normal vectors  $\mathbf{n}_i$  and  $\mathbf{n}_j$  computed at  $\mathbf{p}_i$  and  $\mathbf{p}_j$ . This yields for smooth surfaces, e.g. a plane or a sphere, being grouped as one segment, while surfaces with sharp edges, e.g. between two sides of a box, are grouped into two segments.

The algorithm begins with each vertex in its own segment. The edges are processed by increasing weights, and the two segments  $s_i$  and  $s_j$  connected by a given edge are merged whenever

$$w_{ij} \leq \min \left( d(s_i) + \frac{\kappa}{|s_i|}, d(s_j) + \frac{\kappa}{|s_j|} \right),$$

where  $d(s)$  is the *internal difference* function defined by the maximal edge weight of all edges in the minimum spanning tree of the segment  $s \subseteq \mathcal{V}$ , and  $\kappa$  is a consistency parameter that influences the granularity of the segmentation: a low value of  $\kappa$  requires segments to be more consistent and thus produces more but smaller segments.

In addition to introducing a 3D extension, we make other modifications to the original algorithm. First, as the normal at points with an insufficient number of neighboring points is ill-defined, we do not force every point to be in a segment. No vertices are generated in the graph for these points, and thus no segments are created containing them. Second, as a post-processing step, we remove segments that contain fewer points than a given minimal value  $\tau$ , which are often caused by sensor imperfections or occlusions. In our experiments, a good choice of the segmentation parameters turned out to be  $\kappa = 9$  and  $\tau = 100$ .

### B. Feature Extraction

As shape descriptors, we use spin images [22], shape distributions [23], and shape factors [24]. A *spin image* for a given point  $\mathbf{p}$  with normal vector  $\mathbf{n}$  is defined as a 2D histogram  $H$  oriented along the line  $l$  through  $\mathbf{p}$  with direction  $\mathbf{n}$ . Each bin of  $H$  counts the points with a certain distance to  $l$  and the plane through  $\mathbf{p}$  with normal vector  $\mathbf{n}$ . For the spin image descriptor of a segment  $s$ , we form vectors  $\mathbf{h}_i$  of stacked lines of the histograms  $H_i$  for all points  $\mathbf{p}_i \in s$  and compute the average  $\mathbf{h}^s$  over all  $\mathbf{h}_i$ .

A *shape distribution* is defined as a histogram of values of a predefined function  $f : \mathcal{P}^r \rightarrow \mathbb{R}$ , where  $r$  is the arity of  $f$  and is usually a value between 1 and 4. In our implementation, we use two binary functions  $f_d(\mathbf{p}_i, \mathbf{p}_j)$  and  $f_a(\mathbf{p}_i, \mathbf{p}_j)$ , namely the Euclidean distance between  $\mathbf{p}_i$  and  $\mathbf{p}_j$  and the dissimilarity  $\Delta(\mathbf{p}_i, \mathbf{p}_j)$  as defined above. The resulting histogram vectors  $\mathbf{h}^d$  and  $\mathbf{h}^a$  are computed by evaluating  $f_d$  and  $f_a$  on all pairs of points in a segment  $s$ . To make the feature vectors invariant with respect to the sample density, we normalize the histograms  $\mathbf{h}^d$  and  $\mathbf{h}^a$  by the total number of bin entries.

As we consider scale as a feature of an object, we do not perform normalization with respect to the maximum distance encountered in a segment.

Lastly, we compute *shape factors* per segment, i.e. the normalized eigenvalues of the covariance matrix  $C_i$  of all points in segment  $s_i$ , collected in a vector  $\mathbf{h}^f$ . For each individual descriptor, we compute a PCA to reduce the dimensionality, and the results are combined into a feature vector  $\mathbf{f}$ .

### C. Clustering in Feature Space

To find similar segments, we apply a clustering algorithm in the feature space. The number of existing clustering algorithms is large, and they include agglomerative clustering,  $k$ -means clustering [25], mean-shift estimation [26], spectral clustering [27, 11], and, more recently, affinity propagation (AP) [12]. We explored some of these clustering methods and decided for AP clustering because of its robustness and its ability to estimate the number of clusters implicitly. The basic principle is to determine *exemplars* out of all given points that are well-suited to explain the remaining data points. The application of the algorithm to our case is sketched as follows:

First a similarity value  $\zeta_{ij}$  is computed for all pairs  $(\mathbf{f}_i, \mathbf{f}_j)$  of feature vectors. In our implementation, we use the negative squared Euclidean distance between  $\mathbf{f}_i$  and  $\mathbf{f}_j$ . Then, in an iterative manner, two functions, namely the *responsibility*  $r(i, j)$  and the *availability*  $a(i, j)$  are computed for each vector pair, where  $r$  expresses how well-suited  $\mathbf{f}_j$  is to serve as an exemplar for  $\mathbf{f}_i$  and  $a$  expresses how appropriate it would be for  $\mathbf{f}_i$  if  $\mathbf{f}_j$  were its exemplar. These functions are defined as

$$r(i, j) = \zeta_{ij} - \max_{j' \text{ s.t. } j' \neq j} \{a(i, j') + \zeta_{ij'}\} \quad (1)$$

$$a(i, j) = \min \left\{ 0, r(j, j) + \sum_{i' \text{ s.t. } i' \notin \{i, j\}} \max \{0, r(i', j)\} \right\}, \quad (2)$$

where Eq. (2) is applied only if  $i \neq j$ . Initially,  $a(i, j)$  is set to zero for all  $i$  and  $j$ . For the special case of “self-availability” the rule

$$a(i, i) = \sum_{i' \text{ s.t. } i' \neq i} \max \{0, r(i', i)\} \quad (3)$$

is used. In each iteration, responsibilities and availabilities are computed and then, for each  $\mathbf{f}_i$ , an  $\mathbf{f}_j$  is determined so that the sum  $a(i, j) + r(i, j)$  is maximized. If the resulting  $j$  equals  $i$ , then  $\mathbf{f}_i$  is identified as an exemplar, otherwise  $\mathbf{f}_j$  serves as an exemplar for  $\mathbf{f}_i$ . The stopping criterion of the iteration is met when the assignments of points to exemplars do not change over a fixed number of iterations or a given number of maximum iterations is reached. The only parameter  $\theta_f$  of the algorithm is the *self-similarity*, which can be specified either for each data point individually or commonly for all data points. This value influences the number of resulting clusters: a high value results in more clusters, a low value in fewer clusters. In our experiments, a good value turned out to be  $-0.2$ , specified equally for all data points.

As a result, we obtain  $C$  clusters  $\mathcal{F}_1, \dots, \mathcal{F}_C$  of similar segment instances, where each cluster defines a potential object part.

#### D. Clustering in Geometric Space

From the clustering in feature space we obtain a grouping of segments into object parts. However, we also want to reason on the object level, where objects are considered to consist of several parts. To accomplish this, we also perform a clustering in the geometric space where segments are represented by their center of gravity (COG). Unfortunately, affinity propagation (AP) is not a good choice to perform the clustering in the geometric space because it produces “star-like” clusters, i.e. all points in a cluster are connected directly to the cluster exemplar. This restricts the type of objects that can be detected, as for many objects such an exemplar part can not be found. Also, in AP clustering, the distances between points are not explicitly bounded, which often results in counter-intuitive clustering results.

Therefore, we apply a different, much simpler strategy to cluster the segments. We define a distance threshold  $\vartheta_g$  and connect only those pairs of segments  $(s_i, s_j)$  with an edge, for which the COGs are closer to each other than  $\vartheta_g$ . As a result, all connected components will be farther away from each other than  $\vartheta_g$ , which rises the evidence that they correspond to different object instances. This will be of importance later on.

#### IV. SCENE GRAPH AND PARTS GRAPH

One important aspect of the work presented here is the reasoning about object *instances* solely based on the extraction of potential object *parts*, represented as segments. The challenge here is to find a proper definition of an object class as we do not know of how many parts an object consists and whether or not all of its parts are visible. In addition, the number of observed objects is unknown – we only know the number of object parts. Two intuitions and one assumption help us to reason on parts to discover objects: First, we exploit the fact that segments which occur physically close to each other are more likely to correspond to the same object. Second, segments of one type which occur often in the vicinity to segments of another type give evidence that several instances from the same object class exist. Referring back to the introductory example, we can say: if we find many backrests of a chair that are all close to chair seats, then there is probably an object class which consists of at least these two parts. Furthermore, we work under the assumption that the number of possible object classes is bounded by the number of connected components in the graph which results from clustering the geometric space using the distance threshold  $\vartheta_g$ . We call this the *scene graph* and give details in Sec. IV-A. Our assumption implicitly states that two objects are always supposed to be farther away from each other than  $\vartheta_g$ . This may seem very restrictive, but in fact, it limits the applicability of the algorithm not as much as it appears. The reason is that the number of connected components only bounds the number of object *classes*, not the number of actual object *instances*. Thus, even if two different objects are closer to each other than  $\vartheta_g$ , there are usually enough other connected components with at least two objects of the same class, so that the number of connected components exceeds the number of object classes.

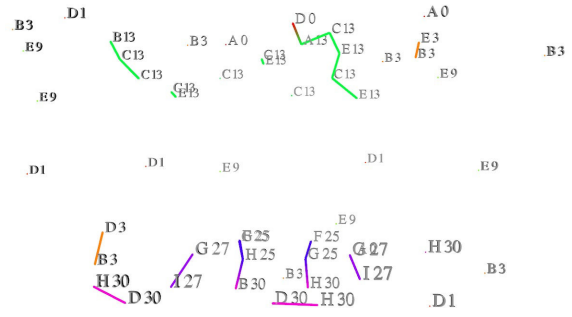


Fig. 2: Example of the scene graph obtained from the range scan shown in Fig. 1. Notice that segments with the label G are given the class labels 25 and 27, depending on their neighboring segments. The figure shows the scene graph after smoothing as described in Sec. V-B.

To analyse the multiple occurrence of segments representing the same object part in constellation with segments that represent a different part, we define another, simpler graph structure named the *parts graph*. The nodes in the parts graph correspond to clusters in feature space as described in Sec. III-C and the edges represent connections in the scene graph. Details on the parts graph are given in Sec. IV-B.

##### A. The Scene Graph

After clustering the segments in the geometric space, we obtain  $K$  subsets of  $\mathcal{S}$ , namely  $\mathcal{G}_1, \dots, \mathcal{G}_K$ . From our assumption, the number of potential object classes is bounded by  $K$ . To encode the fact that neighboring segments are more likely to correspond to the same object class, we define the scene graph  $\mathfrak{S}$ , which consists of the node set  $\mathcal{V}_{\mathfrak{S}} = \mathcal{S}$  and the edge set  $\mathcal{E}_{\mathfrak{S}} = \{(s_i, s_j) \mid \exists \mathcal{G}_i : (s_i, s_j) \subset \mathcal{G}_i\}$ . Thus,  $\mathcal{E}_{\mathfrak{S}}$  consists of all connections between segments that are closer to each other than  $\vartheta_g$ . Furthermore, to each node in scene graph we assign a *class label*  $y \in \{1, \dots, K\}$ . And finally, as each node  $s$  of  $\mathfrak{S}$  corresponds to an instance of an object part, we can associate it with a *part label*  $x \in \{1, \dots, C\}$ . At first sight,  $x$  should be fixed to the index of the feature space cluster  $\mathcal{F}$  to which  $s$  belongs. However, as we formulate our problem in a probabilistic framework, we say that this index is only the *most likely* part label for  $s$  and all others are still possible. Details of this are explained in Sec. V.

An example of a scene graph is shown in Fig. 2. Here, part labels are represented as capital letters and class labels as numbers. We can see that many of the connected components only consist of one segment, as the segments are mostly far apart from each other. Also observe that there are segments with the same part label  $x$ , but with different class labels  $y$ . This stems from the fact that they occur in different contexts.

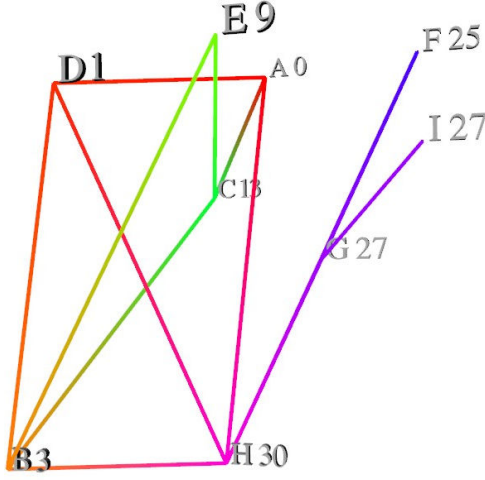


Fig. 3: Example of the parts graph obtained from the scan shown in Fig. 1. The positions of the nodes are determined at random, as only the topology of the graph is important. Again, part labels are represented as capital letters and class labels as numbers.

### B. The Parts Graph

As mentioned before, apart from the class labels of specific segment instances, we also want to reason on the interaction among different object parts in general. We do this using the parts graph  $\mathfrak{P}$ . The node set  $\mathcal{V}_p$  of  $\mathfrak{P}$  corresponds to all clusters  $\mathcal{F}_1, \dots, \mathcal{F}_C$  of the feature space, and the set of edges  $\mathcal{E}_p$  is defined as

$$\mathcal{E}_p = \{(\mathcal{F}_i, \mathcal{F}_j) \mid i \neq j, \exists (\mathbf{s}_k, \mathbf{s}_l) \in \mathcal{E}_s : \mathbf{s}_k \in \mathcal{F}_i \wedge \mathbf{s}_l \in \mathcal{F}_j\}.$$

This means, whenever there are two segments connected in the scene graph, there is a connection between the corresponding segment types in the parts graph. As in the scene graph, each node of the parts graph has an assigned part label  $x$  and a class label  $y$ , although with a slightly different interpretation: An assignment of a class label  $y$  to a node  $\mathcal{F}$  in  $\mathfrak{P}$  means that in general segments that are elements of the cluster  $\mathcal{F}$  are primarily more likely to be of class  $y$ . In contrast to the scene graph, each node of  $\mathfrak{P}$  has a unique most-likely part label  $x$ , because these are directly determined by the feature space clustering.

In Fig. 3, we see an example of a parts graph. The positions of the nodes have been defined randomly as only the topology of the graph is important. Note that this concrete example of a parts graph is non-planar, although there is an equivalent planar representation. In general however, parts graphs may not be representable as planar graphs, for example in the case of a full connectivity.

### V. SMOOTHING

So far, we described the construction of the two graph structures “scene graph” and “parts graph”, but we did not specify how these are used to determine class labels for the segments. We do this using probabilistic reasoning: the nodes in both graphs are interpreted as random variables and the edges are used to model conditional dependencies between adjacent nodes. For the scene graph, this means that the class label  $y_i$  of a given segment  $\mathbf{s}_i$  not only depends on the local evidence of the node  $i$ , i.e. the features  $\mathbf{f}_i$  extracted from  $\mathbf{s}_i$ , but also on the class labels  $y_j$  of all neighbors  $\mathbf{s}_j$  in the scene graph. Intuitively, a strong evidence for  $\mathbf{s}_i$  to belong to a certain class may “outvote” the weaker evidence for  $\mathbf{s}_j$  being of a different class. Similarly, a class label  $y_i$  for a given node  $\mathcal{F}_i$  in the parts graph may be so strong that it *propagates* to the class labels of the neighbors of  $\mathcal{F}_i$ . The reasoning here is that if a certain type of segment  $\mathcal{F}_i$  is very likely to be of a given class, then all segment types that occur frequently in the vicinity of  $\mathcal{F}_i$  are also more likely to be of the same class. Applying this strategy to a graph leads to a *smoothed* graph because it tends to remove sudden changes of class labels between adjacent graph nodes.

In our implementation, we use Conditional Random Fields (CRFs) [1] to perform the smoothing. Our CRF models the conditional distribution

$$p(\mathbf{y} \mid \mathbf{f}) = \frac{1}{Z(\mathbf{f})} \prod_{\mathcal{V}} \varphi(\mathbf{f}_i, y_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{f}_i, \mathbf{f}_j, y_i, y_j), \quad (4)$$

where  $Z(\mathbf{f}) = \sum_{\mathbf{y}'} \prod_{i=1}^N \varphi(\mathbf{f}_i, y'_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{f}_i, \mathbf{f}_j, y'_i, y'_j)$  is the *partition function*,  $\mathcal{V}$  is either  $\mathcal{V}_s$  or  $\mathcal{V}_p$  and  $\mathcal{E}$  is  $\mathcal{E}_s$  or  $\mathcal{E}_p$ . The distinction between the scene graph and the parts graph is made using different definitions for the *node potential*  $\varphi$  and the *edge potential*  $\psi$ . This will be described next.

However, before we proceed with the definition of the potentials, we mention some aspects in our formulation of the CRFs that are slightly different from others found in the literature. First, we recall that node and edge potentials are usually defined using the log-linear model so that, for the case of the node potential

$$\log \varphi(\mathbf{f}_i, y_i) = w_n \cdot f_n(\mathbf{f}_i, y_i), \quad (5)$$

where  $w_n$  is a weight vector and  $f_n$  a feature function, which is high if  $\mathbf{f}_i$  and  $y_i$  in some sense match well. For example, it can be defined as the outcome of a local classification. For our case of an unsupervised setting, we let  $f_n$  be the conditional probability  $p(y_i \mid \mathbf{f}_i)$ , i.e. a scalar value that is high if  $y_i$  matches well to  $\mathbf{f}_i$ . The same is also done for the edge potentials. As a result, the feature functions range between 0 and 1. This simplifies the weighting between node and edge potentials and turns the weight vectors  $w_n$  and  $w_e$  into scalars as well. In contrast to supervised learning with CRFs, we can not learn the node and edge weights  $w_n$  and  $w_e$ , as there is no training data available. Instead, we have to determine them manually. We do this using an appropriate evaluation measure on a validation set. This is described in Sec. VI.



### A. Smoothing the Parts Graph

After creating the parts graph, the next step in our algorithm is to run the inference step using the CRF that corresponds to  $\mathfrak{P}$ . First, we note that the node features of this CRF are not equal to the feature vectors  $\mathbf{f}_1, \dots, \mathbf{f}_M$  extracted from the segments, because a node in  $\mathfrak{P}$  actually represents a cluster in feature space and not a single segment. Instead, we define the node features in  $\mathfrak{P}$  to be the mean  $\bar{\mathbf{f}}$  of all feature vectors inside the corresponding cluster.

Furthermore, we observe that the dependence between feature vectors  $\mathbf{f}$  and labels  $y$  is only implicit as we cannot model it directly. We can, however, model conditional probabilities between segment labels  $x$  and class labels  $y$ , as well as between feature vectors  $\mathbf{f}$  and segment labels  $x$ . Following the definition of the node feature of  $\mathfrak{P}$  as a conditional probability, we have

$$p(y_i | \bar{\mathbf{f}}_i) = \sum_{x=1}^C p(y_i | x) p(x | \bar{\mathbf{f}}_i), \quad (6)$$

where we assume a conditional independence of the labels and the features given the segment types. To obtain the two terms in the sum, we proceed as follows.

The class label posterior  $p(y_i | x)$  is computed using Bayes' rule with the assumption of a uniform prior of the labels:

$$p(y_i | x) = \frac{p(x | y_i) p(y_i)}{\sum_{y'} p(x | y') p(y')} = \frac{p(x | y_i)}{\sum_{y'} p(x | y')} \quad (7)$$

The class likelihoods  $p(x | y_i)$  are determined by counting the occurrence of segments of type  $x$  inside the geometric cluster  $y_i$  and dividing by the number of all segments in  $y_i$ . The posteriors  $p(y_i | x)$  can be computed at creation time of  $\mathfrak{P}$  and collected in a  $K \times C$  matrix, as they do not depend on the node features.

For the segment type posterior  $p(x | \bar{\mathbf{f}}_i)$ , we perform a nearest-neighbor search in feature space inside a sphere of radius  $\rho$  around  $\bar{\mathbf{f}}_i$ . Denoting the number of feature vectors in the sphere with type  $x$  as  $v_x$  and the number of all elements in the sphere as  $v$ , we approximate  $p(x | \bar{\mathbf{f}}_i)$  with  $v_x/v$ .

Similar to Eqn. (6), we define the edge feature of  $\mathfrak{P}$  as

$$p(y_i, y_j | \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) = \sum_{x_i=1}^C \sum_{x_j=1}^C p(y_i, y_j | x_i, x_j) p(x_i, x_j | \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j). \quad (8)$$

As is common in literature related to CRFs, the edge features are designed to be zero whenever the labels  $y_i$  and  $y_j$  of the adjacent nodes are different ("generalized Potts model", see [28, 29]). The rationale of this is that only edges between equally labeled nodes should propagate the belief between the nodes. Thus, the first term in the sum of Eqn. (8) can be simplified to  $p(y_{ij} | x_i, x_j)$ , where  $y_{ij}$  is the common label of the nodes. We can compute this expression by counting the occurrences of edges between  $x_i$  and  $x_j$  in cluster  $y_{ij}$  and applying Bayes rule as in Eq. (7).

For the second term in the sum of Eqn. (8), we apply the formulation

$$\begin{aligned} p(x_i, x_j | \bar{\mathbf{f}}_i, \bar{\mathbf{f}}_j) &= p(x_i | \bar{\mathbf{f}}_i) p(x_j | \bar{\mathbf{f}}_j) \\ &= p(x_i | \bar{\mathbf{f}}_i) p(x_j | \bar{\mathbf{f}}_j), \end{aligned} \quad (9)$$

which results from the conditional independence assumptions on  $x_i$  and  $x_j$ , and from those for  $x_i$  and  $\bar{\mathbf{f}}_j$ , as well as  $x_j$  and  $\bar{\mathbf{f}}_i$ . The resulting terms  $p(x_i | \bar{\mathbf{f}}_i)$  and  $p(x_j | \bar{\mathbf{f}}_j)$  are again computed using nearest-neighbor.

Once the edge and node potentials are defined, we can do inference on the parts graph. We do this using max-product loopy belief propagation. This is an approximate algorithm that returns labels  $\mathbf{y}$  that maximize the conditional probability given in Eqn. (4). However, it also returns distributions over the class labels at each node. These will be used later.

### B. Smoothing the Scene Graph

From the output of the inference step run on  $\mathfrak{P}$ , we obtain at each node of  $\mathfrak{P}$  a distribution over class labels  $y$ . As the nodes of  $\mathfrak{P}$  uniquely represent the segment types  $x$ , we can use that information to read the probability  $p(y | x)$  directly from the parts graph. Thus, when creating the scene graph, we can again determine values for  $p(y | x)$ , as we did this for the parts graph, with the difference that now  $p(y | x)$  also reveals the conditional dependencies between labels of segment types  $x$ , that have been observed in a close distance. This means, that we again compute a matrix for  $p(y | x)$ , but now we simply read the class label distributions off the nodes of  $\mathfrak{P}$ , as they result from belief propagation. In accordance to the node feature of  $\mathfrak{P}$ , we define the node feature of  $\mathfrak{S}$  as the conditional probability

$$p(y_i | \mathbf{f}_i) = \sum_{x=1}^C p(y_i | x) p(x | \mathbf{f}_i). \quad (10)$$

Note that now the feature vectors  $\mathbf{f}_i$  correspond to those that are actually extracted for each segment, i.e. at each node of  $\mathfrak{S}$ . As before, the term  $p(x | \mathbf{f}_i)$  is computed with a nearest-neighbor search in feature space.

Finally, we define the edge feature of  $\mathfrak{S}$  as

$$p(y_i, y_j | \mathbf{f}_i, \mathbf{f}_j) = \sum_{x_i=1}^C \sum_{x_j=1}^C p(y_{ij} | x_i, x_j) p(x_i, x_j | \mathbf{f}_i, \mathbf{f}_j), \quad (11)$$

where we denote the common class label with  $y_{ij}$  as above and the right term in the sum is computed according to Eqn. (9). Unfortunately, the computation of  $p(y_{ij} | x_i, x_j)$  is not straightforward, as we can not simply read this value from the edges of  $\mathfrak{P}$ . Instead, we use the following strategy: The only interesting case is when the class labels of  $x_i$  and  $x_j$  are equal. Thus, we can interpret the common label  $y_{ij}$  as an *edge label* that is determined by one of the adjacent node labels. In a sense, this expresses which of the nodes was responsible for the common edge label  $y_{ij}$ . We can formulate that using a binary variable  $c$  that is true if node  $x_i$  is responsible for  $y_{ij}$  and false otherwise. Using this, we can estimate  $p(y_{ij} | x_i, x_j)$  by marginalizing out over  $c$ :

$$\begin{aligned} p(y_{ij} | x_i, x_j) &= \sum_c p(y_{ij} | x_i, x_j, c) p(c | x_i, x_j) \\ &= p(y_{ij} | x_i, x_j, c) p(c) + p(y_{ij} | x_i, x_j, -c) p(-c) \\ &= 0.5 * p(y_{ij} | x_i) + 0.5 * p(y_{ij} | x_j) \end{aligned} \quad (12)$$

Here, we assumed that  $c$  is independent on  $x_i$  and  $x_j$  and that its prior probability is 0.5. Using Eqn. (12), we can compute  $p(y_{ij} | x_i, x_j)$  by reading the values for  $p(y_{ij} | x_i)$  and  $p(y_{ij} | x_j)$  from the parts graph. As for the parts graph, we use maximum product loopy belief propagation for the inference in the scene graph.

### C. Obtaining the Class Label Distributions

Once the inference step on the scene graph is performed, we read the class label distributions  $\mathbf{d}_1, \dots, \mathbf{d}_M$  from all nodes of  $\mathcal{G}$ . This is possible, as mentioned before, because belief propagation stores these distributions for each node. For the final discovery result, we report the most likely class label at each node, but it is important to note that the distributions may be useful for later reference, for example in an online application, where several data sets are acquired subsequently and the discovery of similar objects is done across the data sets.

## VI. EXPERIMENTAL RESULTS

We tested the algorithm on data acquired from real-world scenes using a nodding SICK laser scanner with a horizontal opening angle of 100 degrees and a nodding range of 90 degrees. Each set was captured at a horizontal resolution of 0.25 degrees and a vertical resolution of 0.2 degrees. We evaluated 50 data sets from four different rooms, each room containing some number of chairs, trash cans, flip charts, plants, etc. Objects were placed up to 90 degrees of rotation from each other. Most scenes contained two or three objects of the same type, but some scenes contained up to four objects of three different kinds.

### A. Qualitative Evaluation

In addition to the result shown in Fig. 1, Fig. 4 shows some more results of our object discovery algorithm. All points that belong to the same object are depicted with the same color, and the numbers represent the class label to which each segment belongs. For instance, the scene in Fig. 1 contains four chairs of two different kinds, and they are correctly labeled as 25 (blue) and 27 (violet). Furthermore, we see that also most of the background segments, e.g. on the floor, ceiling and walls, have plausible labels. For many of them, only the local class label evidence was relevant, as they are not connected in the scene graph.

### B. Quantitative Evaluation

In contrast to supervised learning algorithms, the performance of an unsupervised object discovery method is difficult to evaluate, as there is no real ground truth. The major problem here is that humans tend to be focused and might miss similarities in the data that are irrelevant for them. However, a good way to still evaluate unsupervised object discovery methods has been recently proposed by Tuytelaars *et al.* [30]. This method uses the *conditional entropy* of the “ground truth” class labels  $\mathbf{y}^*$  given the class labels  $\mathbf{y}$  that resulted from

the discovery algorithm. Applied to our case, the conditional entropy is computed as

$$H(Y^* | Y) = \sum_{\sigma(y)=1}^{K^*} p(\sigma(y)) \sum_{y^*=1}^{K^*} p(y^* | \sigma(y)) \log \frac{1}{p(y^* | \sigma(y))},$$

where  $K^*$  is the number of ground truth classes and  $\sigma$  is an *oracle mapping* from the set of discovered classes to the set of ground truth classes, which is determined from a *tuning set* (for details see [30]). Intuitively,  $H(Y^* | Y)$  determines the number of ground truth labels a segment can have once its discovered object label is known. The smaller this number is, the better is the result of the discovery algorithm. In the best case it is zero, which means that each discovered class label directly implies a ground truth label.

We created hand labeled ground truth data consisting of the five classes “ceiling”, “floor”, “wall”, “chair”, and “other”, and evaluated the performance of our algorithm for different values of the parameters  $\theta_g, \rho$ , and the node and edge weights ( $w_n, w_e$ ). The results are shown in Fig. 5. Two conclusions can be drawn from these graphs: First, with values around 1, the results are very good compared to the results of similar algorithms described in [30]. And second, our algorithm is relatively robust against small changes in the choice of the parameters, especially for the distance threshold  $\theta_g$ , which is responsible for the clustering in geometric space.

## VII. CONCLUSION AND OUTLOOK

We presented a fully unsupervised approach to segment 3D range scan data and to discover objects of a similar type that occur more than once in the scene. Our approach uses the only assumption that the number of actually existing object classes is not higher than the number of connected components in the scene, which holds in most cases. We applied probabilistic reasoning based on Conditional Random Fields to model conditional dependencies of object part labels that are close to each other. In experiments on real data we showed that our algorithm is able to discover objects such as chairs in an indoor environment. In the future, we plan to use this technique in an online framework where the evidence of existing object classes is accumulated over time and across different data sets.

### ACKNOWLEDGMENTS

This work was partially supported by the EC in the project robots@home under contract number FP6-IST-045350, in the project BACS under contract number FP6-IST-027140, and in the project EUROPA under contract number EUROPA-FP7-231888.

### R

- [1] J. Lafferty, A. McCallum, and F. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of International Conference on Machine Learning*, 2001.
- [2] T. Tuytelaars, A. Turina, and L. van Gool, “Noncombinatorial detection of regular repetitions under perspective skew,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 4, pp. 418–432, 2003.
- [3] Y. Liu, R. T. Collins, and Y. Tsin, “A computational model for periodic pattern perception based on frieze and wallpaper groups,” *IEEE Trans. on Pattern Anal. and Machine Intell.*, vol. 26, pp. 354–371, 2004.

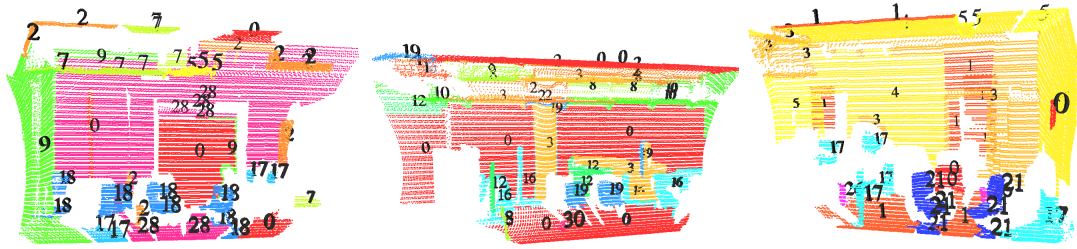


Fig. 4: Three example scenes observed with our nodding SICK laser scanner. Objects that are discovered through the algorithm are colored, where all points which belong to the same object class are assigned to one color. For visualization when viewed in grayscale, the discovered class labels are also shown.

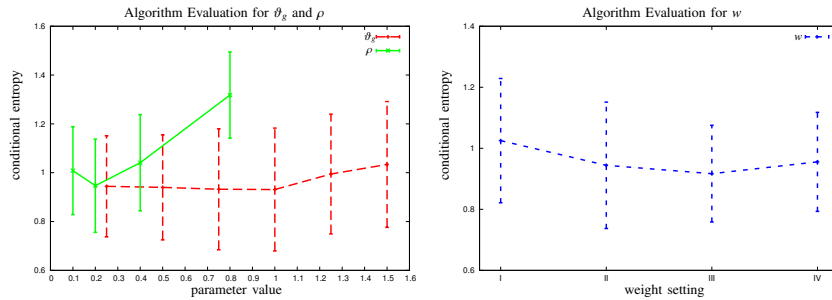


Fig. 5: Evaluation of our discovery algorithm using the conditional entropy with its standard deviation over 50 data sets. Left: Results for different values of  $\theta_g$  and  $\rho$ . As it can be seen,  $\rho$  has a stronger influence on the results as  $\theta_g$ . Right: Results for different settings of the node and edge weights ( $w_n, w_e$ ). Here, 'I' corresponds to the pair (0.5, 1.5), 'II' to (1, 1), 'III' to (1.5, 0.5) and 'IV' to (2.0, 0). The best result is obtained for  $w_n = 1.5, w_e = 0.5$ .

[4] G. Loy and J.-O. Eklundh, "Detecting symmetry and symmetric constellations of features," in *ECCV*, 2006, pp. 508–521.

[5] S. Wenzel, M. Drauschke, and W. Förstner, "Detection of repeated structures in facade images," *Pattern Recognition and Image Analysis*, vol. 18, no. 3, pp. 406–411, 2008.

[6] G. Zeng and L. van Gool, "Multi-label image segmentation via pointwise repetition," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, Alaska, USA, June 2008.

[7] D. Shikhare, S. Bhakar, and S. Mudur, "Compression of large 3d engineering models using automatic discovery of repeating geometric features," in *Vision, Modeling, and Visualization*, 2001.

[8] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser, "A planar-reflective symmetry transform for 3d shapes," in *SIGGRAPH*. New York, NY, USA: ACM, 2006, pp. 549–559.

[9] M. Pauly, N. J. Mitra, J. Wallner, H. Potmann, and L. J. Guibas, "Discovering structural regularity in 3d geometry," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 1–11, 2008.

[10] M. Bokeloh, A. Berner, M. Wand, H. Seidel, and A. Schilling, "Symmetry detection using feature lines," *Computer Graphics Forum (Proceedings of Eurographics)*, vol. 28, no. 2, pp. 697–706(10), April 2009.

[11] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Adv. in Neural Information Processing Systems*, T. Dietterich, S. Becker, and Z. Ghahramani, Eds., 2002.

[12] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, February 2007.

[13] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *IEEE Conf. on Computer Vision*, 2007.

[14] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[15] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *Adv. in Neural Inform. Proc. Systems*, L. Saul, Y. Weiss, and L. Bottou, Eds., 2005, pp. 1097–1104.

[16] X. Ma and W. E. L. Grimson, "Learning coupled conditional random field for image decomposition with application on object categorization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[17] D. Liu and T. Chen, "Semantic-shift for unsupervised object detection," in *Proc. of the Conf. on Comp. Vision and Pattern Rec. Workshop*, 2006.

[18] J. Sivic, B. Russell, A. Efros, A. Zisserman, and W. Freeman, "Discovering object categories in image collections," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005.

[19] F. Endres, C. Plagemann, C. Stachniss, and W. Burgard, "Unsupervised discovery of object classes from range data using latent dirichlet allocation," in *Proc. of Robotics: Science and Systems*, 2009.

[20] J. Shin, R. Triebel, and R. Siegwart, "Unsupervised discovery of repetitive objects," in *IEEE Int. Conf. Robotics and Automation*, 2010.

[21] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[22] A. Johnson, "Spin-images: A representation for 3-d surface matching," Ph.D. dissertation, Robotics Institute, Carnegie Mellon Univ., 1997.

[23] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Trans. on Graphics*, vol. 21, no. 4, pp. 807–832, 2002.

[24] C.-F. Westin, S. Peled, H. Gudbjartsson, R. Kikinis, and F. A. Jolesz, "Geometrical diffusion measures for MRI from tensor basis analysis," in *ISMRM '97*, Vancouver Canada, April 1997, p. 1742.

[25] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[26] D. Comaniciu, P. Meer, and S. Member, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 603–619, 2002.

[27] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[28] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d scan data," in *IEEE Conf. on Comp. Vis. and Pat. Recog. (CVPR)*, 2005, pp. 169–176.

[29] R. B. Potts, "Some generalized order-disorder transformations," *Proc. Cambridge Phil Soc.*, vol. 48, 1952.

[30] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine, "Unsupervised object discovery: A comparison," *Int. Journal of Computer Vision*, vol. 88, no. 2, pp. 284–302, 2009.

# Unsupervised 3D Object Discovery and Categorization for Mobile Robots

Jiwon Shin   Rudolph Triebel   Roland Siegwart

**Abstract** We present a method for mobile robots to learn the concept of objects and categorize them without supervision using 3D point clouds from a laser scanner as input. In particular, we address the challenges of categorizing objects discovered in different scans without knowing the number of categories. The underlying object discovery algorithm finds objects per scan and gives them locally-consistent labels. To associate these object labels across all scans, we introduce *class graph* which encodes the relationship among local object class labels. Our algorithm finds the mapping from local class labels to global category labels by inferring on this graph and uses this mapping to assign the final category label to the discovered objects. We demonstrate on real data our algorithm's ability to discover and categorize objects without supervision.

## 1 Introduction

A mobile robot that is capable of discovering and categorizing objects without human supervision has two major benefits. First, it can operate without a hand-labeled training data set, eliminating the laborious labeling process. Second, if a human-understandable labeling of objects is necessary, automatic discovery and categorization leaves the user with the far less tedious task of labeling categories rather than raw data points. Unsupervised discovery and categorization, however, require the robot to understand what an object constitutes. In this work, we address the challenges of unsupervised object discovery and categorization using 3D scans from a

---

Jiwon Shin  
Autonomous Systems Lab, ETH Zurich e-mail: jiwon.shin@mavt.ethz.ch

Rudolph Triebel  
The Oxford Mobile Robotics Group, University of Oxford e-mail: rudi@robots.ox.ac.uk

Roland Siegwart  
Autonomous Systems Lab, ETH Zurich e-mail: rsiegwart@ethz.ch

laser as input. Unlike other object discovery algorithms, our approach does not assume presegmentation of background, one-to-one mapping between input scan and label, nor a particular object symmetry. Instead, we simply assume that an entity is an object if it is composed of two or more parts and occurs more than once.

We propose a method for robots to discover and categorize objects without supervision. This work especially focuses on categorization of the discovered objects. The proposed algorithm is composed of three steps: detection of potential object parts, object discovery, and object categorization. After segmenting the input 3D point cloud, we extract salient segments to detect regions which are likely to belong to objects. After detecting these potential object parts, we cluster them in feature and geometric space to acquire parts labels and object labels. Reasoning on the relationship between object parts and object labels provides a locally-consistent object class label for each discovered object. Processing a series of scans results in a set of discovered objects, all labeled according to their local class labels. To associate these local class labels, we build a *class graph*. Class graph encodes the dependency among local class labels of similar appearance, and smoothing the graph results in a distribution of the global category labels for each local class label. Marginalizing out the local class labels gives the most likely final category label for each discovered object. We demonstrate on real data the feasibility of unsupervised discovery and categorization of objects.

Contributions of this work are two-folds. First, we improve the object discovery process by extracting potential foreground objects using saliency. Instead of relying entirely on perfect foreground extraction, our algorithm takes the foreground segments only as potential object parts and performs further processing on them before accepting them as object parts. It can thus handle imperfect foreground extraction by removing those potential object parts deemed less fit to be actual object parts. Second, we propose a novel categorization method to associate the locally-consistent object class labels to the global category labels without knowing the number of categories. Our algorithm improves the results of categorization over pure clustering and provides a basis for on-line learning. To our knowledge, no other work has addressed the problem of unsupervised object categorization from discovered objects.

The organization of the paper is as follows. After discussing related work in Sec. 2, we introduce a saliency-based foreground extraction algorithm and explain the single-scan object discovery algorithm in Sec. 3. In Sec. 4, we propose a method for associating the discovered objects for object categorization. After the experimental results in Sec. 5, the paper concludes with Sec. 6.

## 2 Related Work

Most previous work on unsupervised object discovery assume either a presegmentation of the objects, one object class per image, or a known number of objects and their classes [5, 14, 2]. In contrast, [17] proposed an unsupervised discovery algorithm that does not require such assumptions but instead utilizes regularity of

patterns in which the objects appear. This is very useful for man-made structures such as facades of buildings. [3] developed a method to detect and segment similar objects from a single image by growing and merging feature matches.

Our work builds on our previous work [18], which gives nice results for single scenes but does not address the data association problem across different scenes. Thus, the above algorithm cannot identify instances of the same object class that appear in different scenes. In contrast, this approach solves the data association problem and introduces a reasoning on the object level, instead of only assigning class labels to object parts.

An important step in our algorithm is the clustering of feature vectors extracted from image segments. Many different kinds of clustering algorithms have been proposed and their use strongly depends on the application. Some classic methods such as the Expectation-Maximization (EM) algorithm and  $k$ -means clustering assume that data can be modeled by a simple distribution, while other methods such as agglomerative clustering are sensitive to noise and outliers. To overcome these problems, alternative approaches have been proposed. [12] presented a *spectral clustering* algorithm, which uses the eigenvectors of the data matrix to group points together, with impressive results even for challenging data. Another recent clustering approach is named *affinity propagation*, proposed by [6]. It clusters data by finding a set of exemplar points, which serve as cluster centers and explain the data points assigned to it. This method avoids the pitfalls of a bad initialization and does not require the number of clusters to be prespecified. In this work, we use affinity propagation to cluster image segments in feature space.

Our object categorization method is inspired by the *bag of words* approach [4]. Outside of document analysis, the bag of words method has been applied in computer vision, e.g., for texture analysis or object categorization [11, 16]. Our work uses it to bridge the gap between reasoning on object parts and object instances.

### 3 Object Discovery

This section describes the algorithm for discovering objects from a single scan. Fig. 1 depicts the overall process of the object discovery. Our single-scan object discovery algorithm is based on our previous work [18], which treats every segment as a potential object part and accepts them as objects if after inference any nearby segment has the same class label as itself. This algorithm, however, has several disadvantages. First, because the original algorithm considers all segments as potential object parts, it makes many false neighborhood connections between foreground and background segments. This results in object candidates composed of real object parts and background parts. Second, it has relatively high false-positive rate because it cannot differentiate clutter objects from real objects. Third, it wastes computation by extracting feature descriptors on background segments. In this paper, we introduce saliency-based foreground extraction algorithm to overcome these problems.

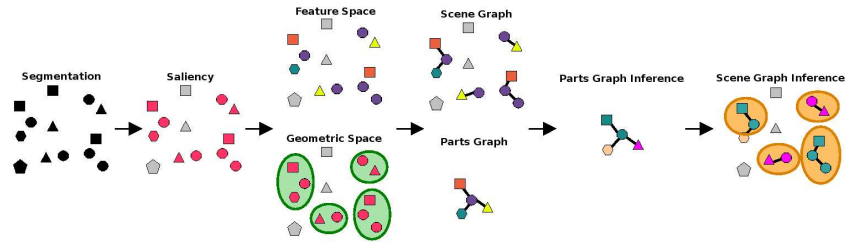


Fig. 1: Overview of the discovery process (best seen in color). After performing segmentation on input data and extracting salient segments, the algorithm clusters the salient segments in feature and geometric space. The clusters are then used to create scene graph and parts graph, which encode the relationship between object parts and objects. Running inference on the graphs result in the discovery of four objects as shown on the right.

### 3.1 Extraction of Potential Object Parts

A simple way to separate foreground from background is to fit planes into the data and remove all points that correspond to the planes. This removes all wall, ceiling, and floor parts as in, e.g., [5], but can cause at least two problems. First, it may also remove planar segments close to a wall or floor that are actually object parts and thus should not be removed. Second, it is often insufficient to define background as planar because background may be truly curved or non-planar due to sensor noise.

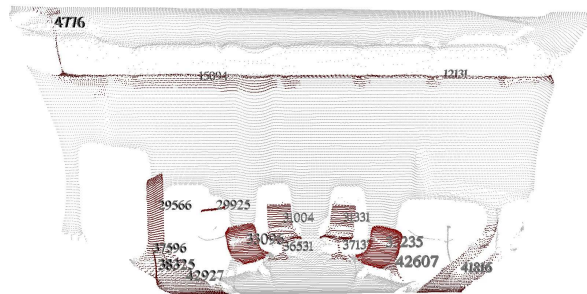


Fig. 2: An example image after saliency computation. Colored segments are considered salient and thus treated as potential object parts. Numbers indicate segment ID.

Inspired by computer vision [8], we suggest a different approach for foreground extraction using *saliency*. The idea is to classify certain parts of an image as visu-

ally more interesting or salient than others. This classification determines saliency based on difference in entropy of a region to its nearby regions. Most work on saliency has been on 3D images, but [7] uses saliency for object recognition in 3D range scans. Their technique, however, remaps depth and reflectance images as grayscale images and applies 2D saliency techniques to find salient points. This work detects salient segments in true 3D by processing depth values of range data directly.

Our saliency algorithm computes saliency at point level and segment level. Point saliency provides saliency of a point while segment saliency represents saliency of a segment. A *point saliency*  $s_p$  is composed of a *local* saliency  $s_l$  and a *global* saliency  $s_g$ . Local saliency  $s_l$  is defined as

$$s_l(\mathbf{p}) = \frac{1}{s_l^{max}} \sum_{\mathbf{p}' \in \mathcal{N}(\mathbf{p})} \mathbf{n} \cdot (\mathbf{p} - \mathbf{p}'), \quad (1)$$

where  $\mathbf{n}$  is the normal vector at a point  $\mathbf{p}$ , and  $\mathcal{N}(\mathbf{p})$  defines the set of all points in the neighborhood of  $\mathbf{p}$ . To obtain a value between 0 and 1, the local saliency is normalized by the maximum local saliency value  $s_l^{max}$ . Intuitively, local saliency measures how much the point  $\mathbf{p}$  sticks out of a plane that best fits into the local surrounding  $\mathcal{N}(\mathbf{p})$ . This resembles the plane extraction technique mentioned earlier.

Points that are closer to the sensor are more likely to belong to foreground and thus globally more salient than points that are far away from the sensor. We capture this property in global saliency. Global saliency  $s_g$  is defined as

$$s_g(\mathbf{p}) = \frac{1}{s_g^{max}} \|\mathbf{p}^{max} - \mathbf{p}\|, \quad (2)$$

where  $\mathbf{p}^{max}$  denotes the point that is farthest away from the sensor origin. As in local saliency, global saliency is normalized to range between 0 and 1.

We define segment saliency  $s_s$  for a segment  $\mathbf{s}$  as a weighted average of the local and global saliency for all points which belong to the segment and multiply it by a size penalty  $\alpha$ , i.e.,

$$s_s(\mathbf{s}) = \alpha \left( \frac{1}{|\mathbf{s}|} \sum_{\mathbf{p} \in \mathbf{s}} w s_l(\mathbf{p}) + (1 - w) s_g(\mathbf{p}) \right), \quad (3)$$

where  $\alpha = \exp(-(|\mathbf{s}| - |\mathbf{s}_{mean}|)^2)$  penalizes segments that are too big or too small as they are likely to originate from a wall or sensor noise;  $|\mathbf{s}|$  denotes the size (number of points) of the segment  $\mathbf{s}$ ; and  $w$  weighs between local and global saliency. The weight  $w$  depends on the amount of information contained in local and global saliency, measured by entropy of the corresponding distributions. Interpreting  $s_l$  and  $s_g$  as probability distributions, we can determine entropy  $h_l$  and  $h_g$  for local and global saliency by



$$h_l = - \sum_{i=1}^N s_l(\mathbf{p}_i) \log s_l(\mathbf{p}_i) \quad (4)$$

$$h_g = - \sum_{i=1}^N s_g(\mathbf{p}_i) \log s_g(\mathbf{p}_i), \quad (5)$$

where  $N = 20$  in this work. As a saliency distribution with lower entropy is more informative, we set the weight  $w$  as  $w = \frac{h_g}{h_g + h_l}$ , which is high when local saliency has low entropy and low when it has high entropy. The weight ensures that more informative entropy distribution contributes more to the final saliency.

Segment saliency  $s_s(s)$  ranges between 0 and 1. We consider a segment salient if its saliency is higher than 0.5 and accept it as a potential object part. Only these potential object parts  $\mathcal{S}$  are further processed for object discovery. Fig. 2 shows a scene after salient segments are extracted.

### 3.2 Object Discovery for a Single Scan

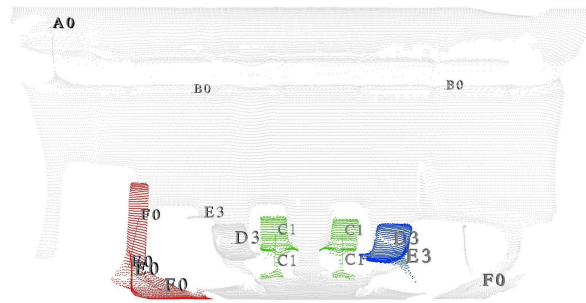


Fig. 3: Result of object discovery of the scene shown in Fig. 2. Discovered objects are colored according to their class labels. Letters indicate the parts types and numbers indicate object classes. Notice that not all potential object parts are accepted as object parts.

Once we extract potential object parts  $\mathcal{S}$ , next step is to reason on them to discover objects. The object discovery step on single scan is based on our previous work [18]. The underlying idea behind our object discovery algorithm is that object parts which belong to the same object are frequently observed together, and hence by observing which parts occur together frequently, we can deduce object class label for these parts. Using this idea, a brief summary of the algorithm is as follows. Given the potential object parts  $\mathcal{S}$ , we extract a feature vector  $\mathbf{f}_i$  for each

potential object part  $\mathbf{s}_i$ . The feature vector  $\mathbf{f}_i$  is composed of spin images [9], shape distributions [13], and shape factors [19]. To determine which set of potential object parts originate from the same *parts type*  $\mathcal{F}_i$ , we cluster these parts in feature space using affinity propagation [6]. Affinity propagation implicitly estimates the number of clusters  $C$ , resulting in clusters  $\mathcal{F}_1, \dots, \mathcal{F}_C$ . These clusters define the discovered object parts types.

Clustering in feature space provides parts types, but it does not define which parts belong to the same object *instance*. To obtain the object instances, we perform another clustering on the potential object parts  $\mathcal{S}$  but this time in geometric space. As object parts for the same object instance are physically close, clustering in geometric space enables us to group together potential object parts which belong to the same object instance. The geometric clustering algorithm connects every pair of potential objects whose centers are closer than a threshold  $\vartheta_g$ , and this results in a collection of connected components. The number of connected components  $K$  define the maximum number of object classes present in the scene, and each cluster  $\mathcal{G}_i$  of the resulting clusters  $\mathcal{G}_1, \dots, \mathcal{G}_K$  correspond to an object instance.

Given parts types  $\mathcal{F}_1, \dots, \mathcal{F}_C$  and object classes  $\mathcal{G}_1, \dots, \mathcal{G}_K$ , next step is to assign a class label  $\mathcal{G}_i$  to each potential object part  $\mathbf{s}_i$ . We determine the assignments by reasoning on the labels at two levels. First, on a more abstract level, the statistical dependency of class labels  $\mathcal{G}_1, \dots, \mathcal{G}_K$  across different parts types  $\mathcal{F}_1, \dots, \mathcal{F}_C$  is encoded in a Conditional Random Field (CRF) [10] named *parts graph*. Parts graph exploits the fact that object parts that co-occur frequently in the same object instance are more likely to belong to the same object class. For example, back rest and seat, both of which belong to a chair, are frequently found together while seat and shelf, which belong to different objects, are not. The second level of reasoning propagates parts types to object class relationship onto a finer level by combining the class labels obtained from the parts graph with the local contextual information from actual scenes. This is encoded using another CRF called *scene graph*. Performing inference on the parts graph provides the most likely object class label  $\mathcal{G}_i$  per parts type  $\mathcal{F}_i$  while inference on the scene graph leads to the object class label  $\mathcal{G}_i$  per object part  $\mathbf{s}_i$ . Once for all object instances, all their parts are labeled with the most likely object class label, we accept those object instances which contains at least two parts with the same class label as discovered objects  $\mathcal{O}_1, \dots, \mathcal{O}_N$ . Fig. 3 shows an example of the outcome of the discovery algorithm.

#### 4 Object Categorization

Object discovery algorithm of the previous section is able to find object classes for which at least two instances occur in a given scene. It uses appearance and geometry, i.e., similarity of features and structures, to find several instances of objects that are most likely to define a class in one given scene. In this paper, we go one step further and try to find object *categories*, i.e., object classes that are consistent across a sequence of input scenes. This, however, is not straightforward. As the ob-



Fig. 4: Objects found in two different scenes. Segments of the same local object label have the same color locally.

ject discovery process is entirely unsupervised, the resulting local class labels are not unique over a given number of input scans. This means that an object class might be associated with a class label  $\mathcal{G}_1$  when one scene is observed, but the same object class might have a different class label  $\mathcal{G}_2$  if observed in a different scene. An example of this is shown in Fig. 4. To identify object instances of the same class from different scenes, we need to solve the *data association problem*. Unfortunately, this problem is intractable in general as it involves a correspondence check between every pair of object classes which are found in different scenes. One simple way to address this correspondence problem is to join all scenes into one big scene and run the discovery algorithm on the big scene. This approach, however, has two major drawbacks: first, the number of connected components  $K$  in this big scene would be very large. This heavily increases the computation time of the algorithm and decreases its detection performance because it fails to sufficiently restrict the number of potential object classes. And second, it limits the possibility of running the object discovery in an online framework, which is one major goal of this work. The reason here is that the parts graph would need to be re-built every time a new scene is observed, which decreases the efficiency of the algorithm.

This work addresses the data association problem by introducing a third level of reasoning named *class graph*. The key idea behind the class graph is to find a mapping from local class labels to global category labels. Unlike the parts graph and the scene graph, the class graph models the statistical dependencies between labels of object class instances rather than object parts. Details of the class graph is explained in Sec. 4.2. Next section describes object feature vector for representation of object instances, which are the building blocks of class graph.

#### 4.1 Object Representation

Object feature vector enables a compact representation of object instances. This work employs object feature vector  $\mathbf{o}$  which captures object instance's appearance and shape. The object feature vector  $\mathbf{o}$  is composed of a histogram  $\mathbf{h}$  of visual word

occurrences and a shape vector  $\mathbf{v}$ . The histogram  $\mathbf{h}$  captures object appearance while the shape vector  $\mathbf{v}$  captures object volume. To compute the histograms, we take the *bag of words* approach and represent an object as a collection of visual words. Bag of words requires visual vocabulary to be defined, and we determine the visual vocabulary by clustering the object parts feature vector  $\mathbf{f}$  of all discovered objects. Each cluster  $\mathcal{F}_i^*$  is a word in the visual vocabulary  $\mathcal{F}_1^*, \dots, \mathcal{F}_{C^*}^*$ , and the total number of words in the vocabulary  $C^*$  is equal to the number of clusters  $C^*$ . With the visual vocabulary, representing an object as a histogram simplifies to counting the number of occurrences of each visual word in the object. In traditional bag of words approaches, every feature makes a contribution to the bin corresponding to the visual word that best represents the feature. Such approaches, however, do not take into account the uncertainty inherent in the assignment process. Hence, in our work, each object part feature vector  $\mathbf{f}$  contributes to all bins of the corresponding histogram  $\mathbf{h}$ , where the contribution to a bin is determined by the probability  $p(\mathbf{w}_i|\mathbf{f})$  of the feature vector  $\mathbf{f}$  belonging to the visual word  $\mathbf{w}_i$ . We compute this probability by nearest-neighbor.

In addition to a histogram  $\mathbf{h}$ , object feature vector  $\mathbf{o}$  contains a shape vector  $\mathbf{v}$ , which represents object's physical properties. The shape vector  $\mathbf{v}$  is composed of three elements – size in horizontal direction, size in vertical direction, and object's location in vertical direction. The horizontal and vertical spans provide the bounding volume in which the object resides. The vertical location gives an estimate on where the object is likely to be found.

## 4.2 Class Graph

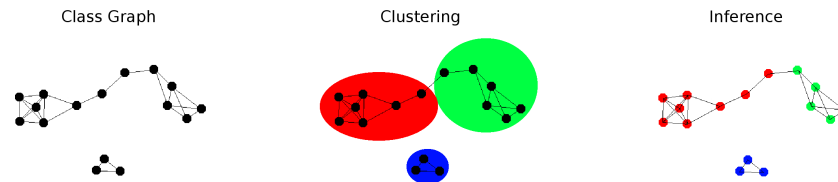


Fig. 5: Categorization by class graph. Local class labels, represented as mean histograms, are the nodes of the graph, and the links between two similar nodes form the edges. Clustering the local class labels provides the initial mapping from local class labels to global category labels. Running inference on the class graph provides a distribution of category labels for each local label. These distributions are then used to determine the category label for each discovered object.

Once the object feature vectors  $\mathbf{o}_1, \dots, \mathbf{o}_{N^*}$  are computed for all discovered objects  $\mathcal{O}_1, \dots, \mathcal{O}_{N^*}$ , we determine the mapping from local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$  to global category labels  $\mathcal{G}_1^*, \dots, \mathcal{G}_{K^*}^*$  using *class graph*  $\mathcal{C}$ . Class graph  $\mathcal{C}$  consists of the node set  $\mathcal{V}_{\bar{\mathbf{o}}} = \{\bar{\mathbf{o}}_1, \dots, \bar{\mathbf{o}}_M\}$  and the edge set  $\mathcal{E}_{\bar{\mathbf{o}}} = \{(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j) \mid D(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j) < \vartheta_{\bar{\mathbf{o}}}\}$ . The nodes are the local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$  represented as mean object feature vectors  $\bar{\mathbf{o}}_1, \dots, \bar{\mathbf{o}}_M$ , and the edges connect similar local class labels, where the similarity between two local labels is the distance between their mean object feature vectors. The threshold for object similarity  $\vartheta_{\bar{\mathbf{o}}}$  is set to 0.5.

To assign global category labels  $\mathcal{G}_1^*, \dots, \mathcal{G}_{K^*}^*$  to local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$ , we need to find the number of global categories  $K^*$ . As mentioned earlier, Affinity Propagation (AP) implicitly determines the number of clusters, and therefore, we cluster the mean object feature vectors  $\bar{\mathbf{o}}_1, \dots, \bar{\mathbf{o}}_M$  by AP clustering. The number of clusters  $K^*$  resulting from AP clustering is the maximum number of global categories, and the clusters  $\mathcal{G}_1^*, \dots, \mathcal{G}_{K^*}^*$  are the initial global category labels for the local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$ . Smoothing this initial mapping determines the final mapping from local class labels to global category labels. Fig. 5 shows the overall steps of categorization by class graph.

### 4.3 Smoothing

Class graph  $\mathcal{C}$  captures the dependency among the local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$ , but it does not assign a category label  $\mathcal{G}_i^*$  to each local label  $\mathcal{G}_i$ . To determine the category labels, we apply probabilistic reasoning. We treat the nodes of the graph as random variables and the edges between adjacent nodes as conditionally dependent. That is, the global category label  $\mathcal{G}_i^*$  of a local class label  $\mathcal{G}_i$  depends not only on the local evidence  $\bar{\mathbf{o}}_i$  but also on the class labels  $\mathcal{G}_j^*$  of all neighboring labels  $\mathcal{G}_j$ . For example, if the local class label  $\mathcal{G}_i$  is strongly of category  $\mathcal{G}_i^*$ , based on its evidence  $\bar{\mathbf{o}}_i$ , then it can propagate its category label  $\mathcal{G}_i^*$  to its neighbors  $\mathcal{G}_j$ . On the other hand, if its category label is weak, then its category label  $\mathcal{G}_i^*$  can be flipped to the category label  $\mathcal{G}_j^*$  of its neighbors. This process penalizes sudden changes of category labels, producing a smoothed graph. We perform the smoothing again using a Conditional Random Field (CRF).

Our CRF models the conditional distribution

$$p(g \mid \bar{\mathbf{o}}) = \frac{1}{Z(\bar{\mathbf{o}})} \prod_{i \in \mathcal{V}_{\bar{\mathbf{o}}}} \varphi(\bar{\mathbf{o}}_i, g_i) \prod_{(i,j) \in \mathcal{E}_{\bar{\mathbf{o}}}} \psi(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j, g_i, g_j), \quad (6)$$

where  $Z(\bar{\mathbf{o}}) = \sum_{g'} \prod_{i \in \mathcal{V}_{\bar{\mathbf{o}}}} \varphi(\bar{\mathbf{o}}_i, g'_i) \prod_{(i,j) \in \mathcal{E}_{\bar{\mathbf{o}}}} \psi(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j, g'_i, g'_j)$  is the *partition function*;  $\mathcal{V}_{\bar{\mathbf{o}}}$  are the local classes; and  $\mathcal{E}_{\bar{\mathbf{o}}}$  are the edges between the local classes. Our formulation of the CRF is slightly different from the conventional approaches in that our feature similarity function  $f_n$  of the node potential  $\log \varphi(\bar{\mathbf{o}}_i, g_i) = w_n \cdot f_n(\bar{\mathbf{o}}_i, g_i)$  is the conditional probability  $p(g_i \mid \bar{\mathbf{o}}_i)$ . Likewise, the feature similarity function  $f_e$  of the edge potential  $\log \psi(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j, g_i, g_j) = w_e \cdot f_e(\bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j, g_i, g_j)$  is also defined as a conditional

probability  $p(g_i, g_j | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j)$ . The feature functions  $f_n$  and  $f_e$  hence range between 0 and 1, simplifying the weighting between node and edge potentials to scalars. In supervised learning with CRFs, node weight  $w_n$  and edge weight  $w_e$  are learned from training data. In this unsupervised work, however, we cannot learn these values as there is no training data available. We therefore determine node weight  $w_n$  and edge weight  $w_e$  manually using an appropriate evaluation measure on a validation set. Fig. 8 in Sec. 5 shows the effect of setting different combinations of node weight  $w_n$  and edge weight  $w_e$ .

As mentioned in Sec. 4.2, the object feature vector clustering provides the total number of global object categories  $C^*$  and the initial mapping from local class labels  $\mathcal{G}_1, \dots, \mathcal{G}_M$  to global category labels  $\mathcal{G}_1^*, \dots, \mathcal{G}_{K^*}^*$ . Using the clusters, we can model the feature similarity function  $f_n = p(g_i | \bar{\mathbf{o}}_i)$  of node potential  $\varphi(\bar{\mathbf{o}}_i, g_i)$  as

$$p(g_i | \bar{\mathbf{o}}_i) = \frac{p(\bar{\mathbf{o}}_i | g_i)p(g_i)}{\sum_{g'} p(\bar{\mathbf{o}}_i | g')p(g')} \quad (7)$$

where  $p(\bar{\mathbf{o}}_i | g_i) = p(\bar{\mathbf{h}}_i | g_i^{\bar{\mathbf{h}}})p(\bar{\mathbf{v}}_i | g_i^{\bar{\mathbf{v}}}) = \exp(-\|\bar{\mathbf{h}}_i - \bar{\mathbf{h}}^{g_i}\|)\exp(-\|\bar{\mathbf{v}}_i - \bar{\mathbf{v}}^{g_i}\|)$  and  $p(g_i) = 1 - \frac{1}{|g_i|+1}$ .  $p(\bar{\mathbf{o}}_i | g_i)$  measures how well  $\bar{\mathbf{o}}_i$  fits to the cluster center  $g_i$ , and the global category prior  $p(g_i)$  reflects how likely the category exists. A cluster with more members are more likely to be a true object category than a cluster with fewer members, and hence  $p(g_i)$  is proportional to the size  $|g_i|$  of the category.

We define the edge feature as

$$p(g_i, g_j | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j) = p(g_i | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j)p(g_j | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j), \quad (8)$$

where  $p(g_i | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j) = p(g_i | \bar{\mathbf{o}}_{ij})$  and  $p(g_j | \bar{\mathbf{o}}_i, \bar{\mathbf{o}}_j) = p(g_j | \bar{\mathbf{o}}_{ij})$  are estimated by a mean object feature vector  $\bar{\mathbf{o}}_{ij}$ . The probabilities  $p(g_i | \bar{\mathbf{o}}_{ij})$  and  $p(g_j | \bar{\mathbf{o}}_{ij})$  are computed by the nearest-neighbor.

To infer the most likely labels for the nodes of the class graph  $\mathcal{C}$ , we use max-product loopy belief propagation. This approximate algorithm returns the labels  $\mathcal{G}_i^*$  which maximizes the conditional probability of Eq. 6. For the message passing, we take the generalized Potts model approach as commonly done and incorporate the edges in the inference only when  $g_i$  and  $g_j$  are equal. This results in the propagation of the belief only between equally-labeled nodes. The inference step continues until convergence and provides the distribution of global category labels  $\mathcal{G}_1^*, \dots, \mathcal{G}_{K^*}^*$  for every local class label  $\mathcal{G}_i$ .

To find the category label  $\mathcal{G}^*$  for each discovered object  $\mathcal{O}$ , we compute the category which maximizes the assignment probability

$$p(g | \mathbf{o}) = \sum_{\bar{\mathbf{o}}'} p(g | \bar{\mathbf{o}}')p(\bar{\mathbf{o}}' | \mathbf{o}). \quad (9)$$

The probability of the category for a given local label  $p(g | \bar{\mathbf{o}}')$  can be read directly from the class graph  $\mathcal{C}$ , and the probability of the local object class given an object  $p(\bar{\mathbf{o}}' | \mathbf{o}) = \exp(-\|\bar{\mathbf{o}} - \mathbf{o}\|)$  is computed as the object's similarity to the class mean. Discovered objects are accepted as objects when the probability of its most likely

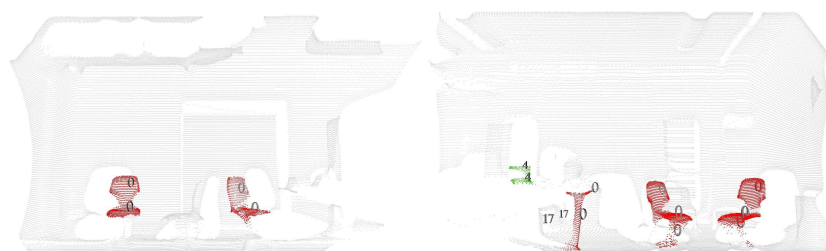


Fig. 6: Objects found in two different scenes. Segments of the same object label have the same color.

category label is greater than 0.5. Fig. 6 shows the results of categorization of the two scenes shown in Fig. 4.

## 5 Results

In this section, we present the results of running the algorithm on scans from real world scenes. The data set was collected using a nodding SICK laser with a width of 100 degrees and a height of 60 degrees. Each set was captured at the horizontal resolution of 0.25 degrees and the vertical resolution of 15 degrees a second. All scenes were static. The test set was a set of 60 scans from four offices. In total, these data sets contained 208 objects, including chairs, couches, poster boards, trash bins, and room dividers.



Fig. 7: The results of object discovery with (left) and without (right) saliency computation. All connected segments are considered objects for categorization. Objects are colored by their local class label.

We first tested the effect of including saliency in the discovery step. Fig. 7 qualitatively shows the difference in object discovery with and without saliency compu-

tation. Including saliency improves the precision<sup>1</sup> of discovery from 44% to 84% while decreasing recall from 83% to 74%. That is, while including the saliency step does eliminate some true objects, it is much more effective at eliminating none objects than the same algorithm without the saliency step.

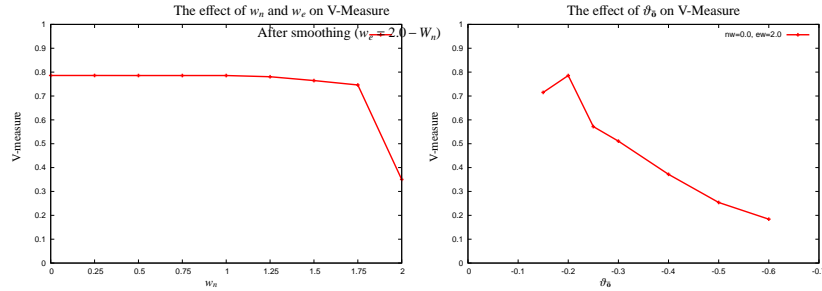


Fig. 8: Evaluation of our categorization step using V-measure. Left graph shows the effect of node and edge weights on v-measure. Right graph shows the effect of the object distance threshold on v-measure.

Quantitatively, we computed V-measure [15] of our algorithm. V-Measure is a conditional entropy-based external cluster evaluation measure which captures the cluster quality by homogeneity and completeness of clusters. It is defined as

$$V_{\beta} = \frac{(1 + \beta) * h * c}{(\beta * h) + c}, \quad (10)$$

where  $h$  captures homogeneity,  $c$  completeness, and  $\beta$  the weighting between homogeneity and completeness. A perfectly homogeneous solution has  $h = 1$ , and a perfectly complete solution has  $c = 1$ . Fig. 8 shows the quality of clustering with varying node and edge weights and the effect of object distance threshold on the quality of clustering. Left graph indicates that the results of our algorithm is robust to the change of node and edge weights, but smoothing improves the overall results over pure clustering. Right graph shows that the quality of clusters depends on the object distance threshold  $\theta_{\delta}$ , which indicates that the initial clustering result influences the final categorization quality.

Fig. 9 shows precision and recall<sup>2</sup> of the algorithm for varying object distance threshold  $\theta_{\delta}$ . Not surprisingly, precision drops and recall increases as the threshold increases. This is because higher threshold results in fewer categories, which in turn means more of the discovered objects are accepted as categorized objects.

<sup>1</sup> A discovered object is considered true positive if it originates from a real object and false positive if it is not a real object. False negative count is when a real object is not discovered.

<sup>2</sup> In computing precision and recall, we did not take into consideration the correctness of the category labels. Any real object that got categorized was considered true regardless of its label.



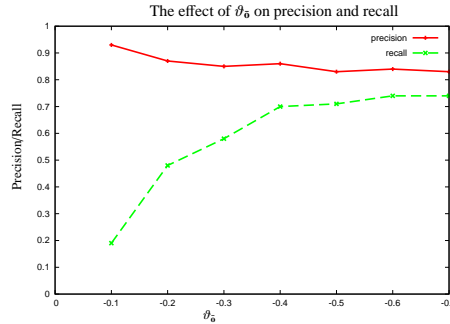


Fig. 9: Effect of the object distance threshold on precision and recall.

Fig. 10 shows qualitative results. Left images are the results of performing object discovery per each scan, and right images are the corresponding images after categorization. Discovered objects are colored according to their local class label, i.e., with respect to other objects within a single scan, while categorized objects are colored according to their global category label, i.e., with respect to all other objects of the data set. The categorization step is able to assign the same global category labels to objects with different local class labels as shown in Fig. 10b while assigning different global category labels to objects with the same local label as shown in Fig. 10d. In addition, the chairs found in different scene are correctly labeled to be the same type as shown in Fig. 10a, 10b, 10d.

## 6 Conclusion and Outlook

We presented a seamless approach to discover and categorize objects in 3D environment without supervision. The key idea is to categorize the objects discovered in various scenes without requiring a presegmented image or the number of classes. Our approach considers objects to be composed of parts and reasons on each part's membership to an object class. After objects are discovered in each scan, we associate these local object labels by building a class graph and inferring on it. We demonstrated our capability of discovering and categorizing objects on real data and performance improvement class graph smoothing brings over pure clustering.

Our approach has several avenues for future work. First, we can use the results of categorization for object recognition. Once the robot has discovered enough instances of an object category, it can use the knowledge to detect and recognize objects, much the same way many supervised algorithms work. Our algorithm simplifies creating training data to converting robotic class representation to human representation. Another direction for future work is on-line learning. While the proposed approach allows the robot to reason on knowledge gained over time, the knowledge

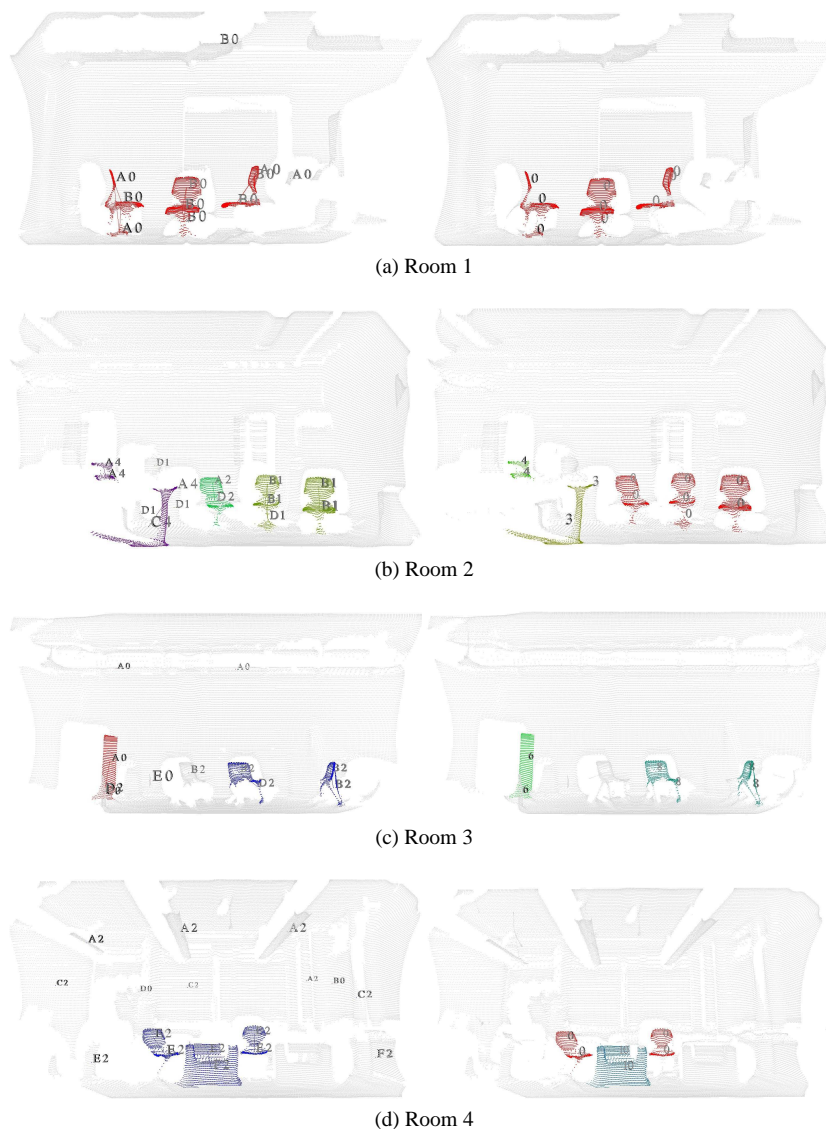


Fig. 10: Results of category discovery. Left images contain objects discovered through the object discovery process, and right images are the same objects after categorization. Objects in the left images are colored according to their local class labels while objects in the right images are colored by their global category labels. Notice that the categorization step can correct incorrect classifications of the discovery step.

is updated in batch. This limits the availability of new information until enough data is collected for the batch processing. A robot, which can process incoming data and update its knowledge on-line, can utilize the new information immediately and adapt to changing environment. Extending our work to handle categorization on-line will thus make unsupervised discovery and categorization more useful for robotics.

## References

1. Bokeloh M, Berner A, Wand M, Seidel HP, and Schilling A (2009) Symmetry Detection Using Feature Lines. *Computer Graphics Forum (Eurographics)* 28.2:697–706(10)
2. Bagon S, Brostovski O, Galun M, and Irani M (2010) Detecting and Sketching the Common. In: *IEEE Computer Vision and Pattern Recognition*
3. Cho M, Shin Y, and Lee K (2010) Unsupervised Detection and Segmentation of Identical Objects. In: *IEEE Computer Vision and Pattern Recognition*
4. Csurka G, Bray C, Dance C, and Fan L (2004) Visual categorization with bags of keypoints. In: *Workshop on Statistical Learning in Computer Vision, ECCV*
5. Endres F, Plagemann C, Stachniss C, and Burgard W (2009) Unsupervised discovery of object classes from range data using latent Dirichlet allocation. In: *Proc. of Robotics: Science and Systems*
6. Frey BJ and Dueck D (2007) Clustering by passing messages between data points. *Science* 315.5814:972–976
7. Frintrop S, Nuechter A, Surmann H, and Hertzberg J (2004) Saliency-based Object Recognition in 3D data. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*
8. Itti L, Kock C, and Niebur E (1998) A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. on Pattern Analysis and Machine Learning* 20.11:1254–1259
9. Johnson A E and Hebert M (1999) Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *IEEE Trans. on Pattern Analysis and Machine Learning* 21.5:433–449
10. Lafferty J, McCallum A, and Pereira F (2001) Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proc. of Int. Conf. on Machine Learning*
11. Leung T and Malik J (1999) Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. In: *Int. Conf. on Computer Vision*
12. Ng A, Jordan M, and Weiss Y (2002) On Spectral Clustering: Analysis and an Algorithm. In: *Adv. in Neural Information Processing Systems*
13. Osada R, Funkhouser T, Chazelle B, and Dobkin D (2002) Shape Distributions. *ACM Trans. on Graphics* 21.4:807–832
14. Ruhnke M, Steder B, Grisetti G, and Burgard W (2009) Unsupervised Learning of 3D Object Models from Partial Views. In: *IEEE Int. Conf. Robotics and Automation, Kobe, Japan*
15. Rosenberg A and Hirschberg J (2007) V-Measure: A Conditional Entropy-based External Cluster Evaluation Measure. In: *Joint Conf. on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*
16. Sivic J, Russell B, Efros A, Zisserman A, and Freeman W (2005) Discovering Object Categories in Image Collections. In: *Proc. of the Int. Conf. on Computer Vision*
17. Spinello L, Triebel R, Vasquez D, Arras K, and Siegwart R (2010) Exploiting Repetitive Object Patterns for Model Compression and Completion. In: *European Conf. on Computer Vision*
18. Triebel R, Shin J, and Siegwart R (2010) Segmentation and Unsupervised Part-based Discovery of Repetitive Objects. In: *Proc. of Robotics: Science and Systems*
19. Westin C, Peled S, Gudbjartsson H, Kikinis R, and Jolesz F (1997) Geometrical Diffusion Measures for MRI from Tensor Basis Analysis. In: *ISMRM '97*

# A Bayesian Approach to Learning 3D Representations of Dynamic Environments

Ralf Kästner, Nikolas Engelhard, Rudolph Triebel, and Roland Siegwart

**Abstract** We propose a novel probabilistic approach to learning spatial representations of dynamic environments from 3D laser range measurements. Whilst most of the previous techniques developed in robotics address this problem by computationally expensive tracking frameworks, our method performs in real-time even in the presence of large amounts of dynamic objects. The computer vision community has provided comparable methods for learning foreground activity patterns in images. However, these methods generally do not account well for the uncertainty involved in the sensing process. In this paper, we show that the problem of detecting occurrences of non-stationary objects in range readings can be solved online under the assumption of a consistent Bayesian framework. Whilst the model underlying our framework naturally scales with the complexity and the noise characteristics of the environment, all parameters involved in the detection process obey a clean probabilistic interpretation. When applied to real-world urban settings, the results produced by our approach appear promising and may directly be applied to solve map building, localization, or robot navigation problems.

## 1 Introduction and Related Work

Understanding dynamic properties of the world has become an increasingly popular research topic in mobile robotics. The motivations for this popularity are manifold. The occurrence of moving objects in the robot's sensor range may for example corrupt the localization or map building process [12, 1, 3]. On the other hand, novel

---

Ralf Kästner, Rudolph Triebel, Roland Siegwart  
Autonomous Systems Lab, ETH Zurich, Switzerland  
e-mail: [kaestner, triebel]@mavt.ethz.ch, rsiegwart@ethz.ch

Nikolas Engelhard  
University of Freiburg, 79110 Freiburg, Germany  
e-mail: nikolas.engelhard@informatik.uni-freiburg.de

planning approaches aim at navigating platforms through highly dynamic environments [11, 6]. They therefore strongly rely on robust motion parameter estimates for objects that may potentially interfere with the robot's trajectory.

A widely common group of methods addressing motion estimation is committed to tracking the displacement of entire point clusters [10]. Whilst such approaches succeed in obtaining a parametric description of the cluster motion, they usually take strong assumptions about the size or shape of objects. In the above scenarios, however, we generally do not want to constrain ourselves with a limited number of object classes. In fact, we seek to detect motion rather than to find explicit motion parameters. Consider therefore a sensor reading that has been introduced by a dynamic object. If, at any later point in time, we acquire another reading that matches the previous observation, we may not care to also answer the difficult question of identity. That is, no matter if the measurement originates from a single or two different objects, we would still want to classify it as being dynamic.

In this paper, we propose a novel approach to the problem of learning 3D representations of dynamic environments from range data. In strong analogy to background modeling in computer vision [8], this problem constitutes a binary classification task. That is, for a series of range observations we seek to estimate whether single measurements originate from a static or a dynamic object. We therefore represent correspondences between measurements and objects using Gaussian mixture distributions [14, 13].

Where standard methods for learning Gaussian mixtures fail due to the non-stationary nature of a dynamic world model [5], we propose an alternative on-line solution that does not make any assumptions about the number of Gaussians in the mixture model but efficiently scales with the complexity of the environment. Even in highly populated settings, our approach is thus capable of distinguishing dynamic from static objects in real-time.

The emphasis of this work strongly lies on the Bayesian formalization of all steps involved in the learning process [8]. In fact, following techniques used in probabilistic change detection [7] our method is strictly governed by the laws of probability, and each effective parameter comes with a clear probabilistic interpretation.

We demonstrate the practicability of our approach in simulation and by experiments involving several urban outdoor scenarios with a diversity of static structures and dynamic objects.

## 2 Probabilistic Formulation

Our algorithm to learning dynamic environment representations operates on range readings acquired with a nodding 2D laser range scanner that pitches up and down during data acquisition to produce 3D point clouds. This setup has been used frequently in the literature (see, e.g. [15]) and is usually known as the *nodding laser scanner* configuration. Throughout this paper, we define a sensor measurement  $\mathbf{z}_t$  as the tuple  $(r_t, \vartheta_t, \varphi_t)$ , where  $r_t$  is the measured range, and  $\vartheta_t$  and  $\varphi_t$  are the pitch and

yaw angles of the laser beam at time  $t$  of the data acquisition. We assume that  $r_t$  is affected by Gaussian noise, thus we have  $r_t \sim \mathcal{N}(\hat{r}_t, \sigma_r)$ , where  $\hat{r}_t$  is the true distance between the laser origin and the observed object.

Since we do not account for uncertainty in the acquisition angles  $\vartheta$  and  $\varphi$  of our nodding range scanner, we represent full sensor sweeps as *range images*. A range image is defined by an  $N \times M$  matrix of cells  $c_{i,j}$ , representing a discretization of the continuous space  $\mathcal{S} = [-\vartheta_{max}, \vartheta_{max}] \times [-\varphi_{max}, \varphi_{max}]$ , where the pitch and yaw angles  $\vartheta$  and  $\varphi$  of the laser beam range between predefined boundaries. For each  $\mathbf{z}_t$ , we can thus compute a 3D coordinate vector  $\mathbf{z}_t^{[i,j]}$ , with  $i$  and  $j$  denoting the indices of the range image cell  $c_{i,j}$  that corresponds to  $r_t$ .

To formulate our problem mathematically, we furthermore introduce a binary state variable  $\mathbf{x}_t$ , which is true if the observation  $\mathbf{z}_t$  corresponds to a dynamic object and false otherwise. Our aim is to estimate  $\mathbf{x}_t$  at any time step  $t$ , given the noisy measurement  $\mathbf{z}_t$  acquired at time  $t$ . Formally, we therefore want to find  $p(\mathbf{x}_t | \mathbf{z}_t)$ . Assuming statistical independences between all range image cells, we can then estimate the joint posterior probability of range measurements being caused by dynamic objects  $\bar{\mathbf{x}}_t = \{\mathbf{x}_t^{[i,j]}\}$  given the range image  $\bar{\mathbf{z}}_t = \{\mathbf{z}_t^{[i,j]}\}$  as

$$p(\bar{\mathbf{x}}_t | \bar{\mathbf{z}}_t) = \prod_{i,j} p(\mathbf{x}_t^{[i,j]} | \mathbf{z}_t^{[i,j]}). \quad (1)$$

To keep the following notations uncluttered, we will drop the superindices  $[i, j]$  and perform all further computations only on the cell level. Consequently, the conditional  $p(\mathbf{x}_t | \mathbf{z}_t)$  associated with each cell shall from now on be referred to as *cell posterior*.

## 2.1 Formulation using Gaussian Mixture Models

To infer the binary states  $\mathbf{x}_t$ , we use a generative approach: we assume that each observation  $\mathbf{z}_t$  was caused by the existence of one of  $K$  objects, which can be either dynamic or static. The unobserved distance of each object to the laser origin is modeled using a normal distribution  $\mathcal{N}(\mu_k, \sigma_k)$ , with mean  $\mu_k$ , variance  $\sigma_k$ , and  $k \in \{1, \dots, K\}$ . For each cell, we thus yield a set of model parameters which shall henceforth be denoted  $\Theta = \{K, \mu_1, \dots, \mu_K, \sigma_1, \dots, \sigma_K\}$ . To express the fact that  $\mathbf{z}_t$  corresponds to object  $k$ , we furthermore introduce binary correspondence variables  $g_t^k$ , where only one  $g_t^k$  can be true for any  $\mathbf{z}_t$ .

## 2.2 Decomposing the Cell Posterior

In an online process, our model changes over time. Let  $\Theta_t$  therefore represent the set of parameters at acquisition time  $t$ . Together with the states  $\mathbf{x}_t$ , the model parameters are generally unknown and need to be inferred. In other words, at any given time  $t$  we will be facing the question of how to update our latest parameter estimate  $\Theta_{t-1}$  using the most recent observation  $\mathbf{z}_t$ . We therefore want to make the  $\Theta_t$  and  $\Theta_{t-1}$  explicit in the probabilistic formulation and rewrite the cell posterior from Eqn. (1) as

$$\begin{aligned} p(\mathbf{x}_t, \Theta_t | \mathbf{z}_t, \Theta_{t-1}) &= p(\mathbf{x}_t | \mathbf{z}_t, \Theta_t, \Theta_{t-1}) p(\Theta_t | \mathbf{z}_t, \Theta_{t-1}) \\ &= p(\mathbf{x}_t | \mathbf{z}_t, \Theta_t) p(\Theta_t | \mathbf{z}_t, \Theta_{t-1}). \end{aligned} \quad (2)$$

Here, the second equality assumes our model  $\Theta_t$  to provide a complete description of the underlying process.

The above equation constitutes two conditionals that are essential to finding the cell posteriors. The first conditional describes an assignment of binary states  $\mathbf{x}_t$  to observations  $\mathbf{z}_t$  under the assumption that all model parameters  $\Theta_t$  are known. It shall therefore be coined as *dynamics likelihood*. The second conditional implies the sought model  $\Theta_t$  from an observation  $\mathbf{z}_t$  and the most recent parameter set  $\Theta_{t-1}$ . Accordingly, it will be termed the *update rule*.

## 2.3 The Dynamics Likelihood

We follow the approach presented in [8] and express the dynamics likelihood by marginalization over all correspondence variables  $g_t^k$ . We thus obtain

$$\begin{aligned} p(\mathbf{x}_t | \mathbf{z}_t, \Theta_t) &= \frac{p(\mathbf{x}_t, \mathbf{z}_t | \Theta_t)}{p(\mathbf{z}_t | \Theta_t)} \\ &= \frac{\sum_{k=1}^K p(x_t^k, \mathbf{z}_t | g_t^k, \Theta_t^k) p(g_t^k | \Theta_t^k)}{\sum_{k=1}^K p(\mathbf{z}_t | g_t^k, \Theta_t^k) p(g_t^k | \Theta_t^k)} \\ &= \frac{\sum_{k=1}^K p(x_t^k | g_t^k, \Theta_t^k) p(\mathbf{z}_t | g_t^k, \Theta_t^k) p(g_t^k | \Theta_t^k)}{\sum_{k=1}^K p(\mathbf{z}_t | g_t^k, \Theta_t^k) p(g_t^k | \Theta_t^k)}. \end{aligned} \quad (3)$$

Here, we introduce object parameters  $\Theta_t^k = \{\mu_t^k, \sigma_t^k\}$  and individual state variables  $x_t^k$  expressing if the  $k$ -th Gaussian is dynamic or static. We furthermore assume the probability of observing  $\mathbf{z}_t$  to be equal for both dynamic and static objects, given the knowledge that  $\mathbf{z}_t$  was caused by object  $k$ . This results in the conditional independence relation  $p(x_t^k, \mathbf{z}_t | g_t^k, \Theta_t^k) = p(x_t^k | g_t^k, \Theta_t^k) p(\mathbf{z}_t | g_t^k, \Theta_t^k)$ .

From Eqn. (3), three terms need to be specified further. Starting from the right, we first note that  $p(g_t^k | \Theta_t^k)$  is the *prior* probability for  $\mathbf{z}_t$  being caused by object  $k$ . Usually,  $p(g_t^k | \Theta_t^k)$  is named the *weight* of the  $k$ -th Gaussian in the mixture and denoted by the symbol  $w_t^k$ . Second, the *data likelihood*  $p(\mathbf{z}_t | g_t^k, \Theta_t^k)$  is equal to  $\mathcal{N}(\mathbf{z}_t; \mu_t^k, \sigma_t^k)$ . And, finally, the *dynamics likelihood of Gaussian  $k$*  is defined as  $p(\mathbf{x}_t^k | g_t^k, \Theta_t^k)$ .

## 2.4 The Update Rule

Just as for the dynamics likelihood, we write the update rule as a marginal over correspondences  $g_t^k$  and obtain

$$p(\Theta_t | \mathbf{z}_t, \Theta_{t-1}) = \sum_{k=1}^K p(\Theta_t^k | \mathbf{z}_t, g_t^k, \Theta_{t-1}^k) p(g_t^k | \mathbf{z}_t, \Theta_{t-1}^k) \quad (4)$$

This provides us with two additional terms. The *correspondence likelihood*  $p(g_t^k | \mathbf{z}_t, \Theta_{t-1}^k)$  constitutes a statistical law for selecting Gaussian  $k$  as an explanation for the occurrence of  $\mathbf{z}_t$ , and the *update rule of Gaussian  $k$*  is denoted by  $p(\Theta_t^k | \mathbf{z}_t, g_t^k, \Theta_{t-1}^k)$ .

## 2.5 The Mixture Weights

As stated above, we assume that each observation  $\mathbf{z}_t$  is caused by only one possible object. This corresponds to the *hard assignment* of data points to clusters known from the  $k$ -means clustering algorithm. Using this, we can say that all  $t$  data points acquired at the discrete time steps  $1, \dots, t$  are each assigned to one out of  $K$  clusters where each cluster corresponds to a Gaussian. If we denote the number of observations that correspond to cluster  $k$  at time  $t$  by  $n_t^k$ , we can estimate the prior probability of a new observation  $\mathbf{z}_t$  to be caused by object  $k$ . As this prior is equal to the weight  $w_t^k$ , we have

$$p(g_t^k | \Theta_t^k) = w_t^k = \frac{n_t^k}{t} \quad (5)$$

We note that knowing  $t$ ,  $n_t^k$  thus becomes an equivalent representation of the mixture weight  $w_t^k$ .



### 3 Online Estimation of the Mixture Parameters

In this section, we present an online implementation for estimating our model parameters. As shown above, we can compute the posterior probability of an observation  $\mathbf{z}_t$  to be caused by a dynamic object in two stages: First, we apply the update rule according to Eqn. (4) and reestimate the mixture parameters  $\Theta_t$  with respect to the latest model  $\Theta_{t-1}$ . And second, we use the dynamics likelihood from Eqn. (3) on the updated mixture models in order to infer new state variables  $\mathbf{x}_t$ .

#### 3.1 Sequential Parameter Updates

A key contribution of this paper pertains to the central question of how to integrate new observations with an existing cell mixture. In contrast to the approach presented in [14], we seek to strictly govern the sequential update of our mixture models by statistical densities and the laws of probability.

In mathematical terms, finding the optimal parameter assignment for  $\Theta_t$  given  $\Theta_{t-1}$  and an observation  $\mathbf{z}_t$  is equivalent to maximizing the update probability  $p(\Theta_t | \mathbf{z}_t, \Theta_{t-1})$ . We therefore want to reconsider the probabilistic update rule for our cell mixtures defined in Eqn. (4). It suggests that in order to maximize  $p(\Theta_t | \mathbf{z}_t, \Theta_{t-1})$ , we first need to recover an optimal assignment for the correspondence variables  $g_t^k$ .

We recall that following our above assumptions, each observation  $\mathbf{z}_t$  may only be caused by one possible object. Put differently, we are interested in inferring whether a sensor response  $\mathbf{z}_t$  originates from an object  $k \in \{1, \dots, K\}$  that is already represented by our mixture or not. Following the proposal in [7], we therefore introduce a joint probability and estimate correspondences with respect to two distinct cases.

**Explained** An observation  $\mathbf{z}_t$  can be explained by the  $k$ -th Gaussian in the current mixture model. Consider therefore the range reading  $r_t$  associated with  $\mathbf{z}_t$  along with the expected measurement noise  $\sigma_r$ . Furthermore, let  $\Theta_{t-1}^k$  be the parameters of the Gaussian  $k$  in the mixture distribution that best explains  $\mathbf{z}_t$ . Then our observation model gives rise to the assumption that  $p(\mathbf{z}_t | g_t^k) \sim \mathcal{N}(\mu_{t-1}^k, \sigma_{t-1}^k + \sigma_r)$ . Note that by summing up the variances of the measurement  $\sigma_r$  and the object representation  $\sigma_{t-1}^k$ , we account for the noise in both models.

**Unexplained** An observation cannot be explained by any of the  $K$  Gaussians in the current mixture model. Without making any specific assumption on how unobserved objects occur within the sensor range, we assume a uniform distribution over the entire beam length. Hence, we define  $p(\mathbf{z}_t | g^{new}) \sim \mathcal{U}(0, r_{max})$  where  $\mathcal{U}$  denotes the uniform distribution with support in  $[0, r_{max}]$ , and  $g^{new}$  is a new Gaussian explaining  $\mathbf{z}_t$ .

Given the above cases, we are able to arrive at a posterior for unexplained observations. We define  $p_{new} = p(g^{new})$  and apply Bayes rule to relate  $p(g^{new} | \mathbf{z}_t)$  to the likelihood at which observations are generated by unrepresented objects:

$$p(g^{new} | \mathbf{z}_t) = \frac{p(\mathbf{z}_t | g^{new}) \cdot p_{new}}{p(\mathbf{z}_t | g^{new}) \cdot p_{new} + p(\mathbf{z}_t | g_t^k) \cdot (1 - p_{new})} \quad (6)$$

Then, by exploiting the assumption that  $p(\mathbf{z}_t | g^{new}) \cdot p_{new}$  is small we can approximate the logarithm of this expression

$$\begin{aligned} \log p(g^{new} | \mathbf{z}_t) &\approx \log \frac{p(\mathbf{z}_t | g^{new}) \cdot p_{new}}{p(\mathbf{z}_t | g_t^k) \cdot (1 - p_{new})} \\ &= \log p(\mathbf{z}_t | g^{new}) + \log p_{new} - \\ &\quad \log p(\mathbf{z}_t | g_t^k) - \log(1 - p_{new}) \end{aligned} \quad (7)$$

Plugging in our model assumptions with respect to the described cases consequently yields

$$\begin{aligned} \log p(g^{new} | \mathbf{z}_t) &\approx -\log r_{max} + \log p_{new} - \log(1 - p_{new}) + \\ &\quad \frac{1}{2} \log 2\pi(\sigma_{t-1}^k + \sigma_r) + \frac{1}{2} \frac{(r_t - \mu_{t-1}^k)^2}{(\sigma_{t-1}^k + \sigma_r)} \end{aligned} \quad (8)$$

It appears beneficial to combine all expressions depending on the range measurement  $r_t$  in the above equation. This leaves us with a simple quadratic distance

$$d_k(r_t) = (r_t - \mu_{t-1}^k)^T (\sigma_{t-1}^k + \sigma_r)^{-1} (r_t - \mu_{t-1}^k) \quad (9)$$

Exploiting the assumption that an unexplained observation with a probability of  $p(g^{new} | \mathbf{z}_t) > 0.5$  is significant, this distance may then be compared to the following constant threshold

$$\begin{aligned} d_k^{min} &= 2 \log 0.5 + 2 \log r_{max} - 2 \log p_{new} + \\ &\quad 2 \log(1 - p_{new}) - \log 2\pi(\sigma_{t-1}^k + \sigma_r) \end{aligned} \quad (10)$$

We conclude that if  $d_k(r) < d_k^{min}$ , the  $k$ -th Gaussian is a possible explanation for the occurrence of observation  $\mathbf{z}_t$ .

The reader may have noticed that the method presented so far only allows for selecting a set of candidate Gaussians from the mixture. We therefore propose to proceed as follows: If we find any Gaussian explaining  $\mathbf{z}_t$ , we will arrange a hard assignment of  $\mathbf{z}_t$  to the candidate Gaussian  $k$  with the lowest distance  $d_k(r_t)$ . Note that this is equivalent to selecting the object with the highest correspondence likelihood  $p(g_t^k | \mathbf{z}_t) = 1 - p(g^{new} | \mathbf{z}_t)$ . We then account for maximizing Eqn. (4) by computing the *maximum likelihood* estimate for the  $k$ -th Gaussian. A sequential approach exists to finding the maximum likelihood solution for the parameters of a

**Algorithm 1:** updateMixture( $\mathcal{M}_{t-1}, r_t$ )

---

**Input:** Mixture of Gaussians  $\mathcal{M}_{t-1} = \{\langle n_{t-1}^1, \theta_{t-1}^1 \rangle, \dots, \langle n_{t-1}^K, \theta_{t-1}^K \rangle\}$   
**Input:** Laser reading  $r_t$   
**Output:** Updated Mixture of Gaussians  $\mathcal{M}_t$

$\mathcal{M}_{cand} \leftarrow \emptyset$   
**foreach**  $\langle n_{t-1}^k, \theta_{t-1}^k \rangle \in \mathcal{M}_{t-1}$  **do**  
    Evaluate  $d_k(r_t)$  and  $d_k^{min}$  according to Eqs. 9 and 10  
    **if**  $d_k(r_t) < d_k^{min}$  **then**  
        |  $\mathcal{M}_{cand} \leftarrow \mathcal{M}_{cand} \cup \{\langle n_{t-1}^k, \theta_{t-1}^k \rangle\}$   
    **end**  
**end**  
**if**  $\mathcal{M}_{cand} \neq \emptyset$  **then**  
     $\langle n_{t-1}^k, \theta_{t-1}^k \rangle \leftarrow \underset{\langle n_{t-1}^l, \theta_{t-1}^l \rangle \in \mathcal{M}_{cand}}{\operatorname{argmin}} d_l(r_t)$   
     $\mathcal{M}_t \leftarrow \mathcal{M}_{t-1} \setminus \{\langle n_{t-1}^k, \theta_{t-1}^k \rangle\}$   
     $n_t^k \leftarrow n_{t-1}^k + 1$   
     $\theta_t^k \leftarrow \operatorname{updateGaussian}(\theta_{t-1}^k, r_t)$   
     $\mathcal{M}_t \leftarrow \mathcal{M}_t \cup \{\langle n_t^k, \theta_t^k \rangle\}$   
**else**  
     $n_t^{K+1} \leftarrow 1$   
     $\theta_t^{K+1} \leftarrow \{\mu_t^{K+1} \leftarrow r_t, \sigma_t^{K+1} \leftarrow \sigma_r\}$   
     $\mathcal{M}_t \leftarrow \mathcal{M}_{t-1} \cup \{\langle n_t^{K+1}, \theta_t^{K+1} \rangle\}$   
**end**  
collapseMixture( $\mathcal{M}_t$ )

---

Gaussian distribution that allows new observations  $\mathbf{z}_t$  to be processed one at a time. For a detailed discussion of this approach, the interested reader may refer to [2].

In cases where no candidate Gaussian exists, we represent the observed object by introducing a new Gaussian  $K + 1$  into the mixture. This Gaussian is then initialized with mean  $r_t$  and variance  $\sigma_r$ .

We are now ready to state Alg. 1 for sequential parameter updates. It takes the range measurement  $r_t$  associated with a new observation  $\mathbf{z}_t$  and the corresponding cell mixture distribution  $\mathcal{M}_{t-1}$  as input arguments and in return outputs the updated density  $\mathcal{M}_t$ .

Our algorithm makes use of two auxiliary functions. As the name suggests, `updateGaussian` sequentially reestimates mean and variance of the best candidate Gaussian with respect to  $r_t$ . The second function of concern is `collapseMixture`. It provides an abstract mechanism for joining Gaussians that share a significant fraction of the state space. To see why this is necessary, the reader may consider the very nature of the mixture density  $p(\mathbf{z}_t | \theta_t)$ . This density actually constitutes a noise model of the environment with the sensor uncertainty displayed by newly added Gaussians solely acting as a prior. In fact, the variance of each Gaussian may often grow beyond this initial uncertainty, e.g. in order for it to represent a highly scattered surface.

Although alternative approaches exist to collapsing Gaussian mixture models (see e.g. [9], p. 185 ff.), we propose to use a method that widely resembles our

sequential update algorithm. Just as for associating noisy observations with the most likely mixture candidates, we have implemented a very similar algorithm to perform a pair-wise identification of Gaussians obeying the distance threshold from Eq. 10. The advantage of such an implementation is clear at hand: Instead of introducing a new parameter into the update step, we may reuse the prior probability  $p_{new}$  for the occurrence of unexplained objects. Hence, the total number  $K$  of Gaussians contained in a cell mixture is bound by our model assumptions and does not require artificial clamping.

### 3.2 Computing the Dynamics Likelihood

Above, we have demonstrated how to update the parameters of a Gaussian mixture distribution that is capable of representing observations in statistically independent range image cells. However, our model does not yet disambiguate between observations caused by dynamic and observations caused by static objects. Or formally speaking, we have not yet provided an estimate for the binary state variables  $x_t^k$  associated with each of the Gaussians.

From the dynamics likelihood stated in Eqn. (3), we know that the conditional density  $p(x_t^k | g_t^k, \theta_t^k)$  defines a state labeling strategy and that, in its most general form, this strategy constitutes a weighting of the  $k$ -th Gaussian in the mixture distribution.

To compute the dynamics likelihood of Gaussian  $k$ , we use the method described in [14]. It is based on the observation that the majority of range readings usually originates from static objects. This means in turn that the objects, to which the most observed data points correspond, are more likely to be static. As we model each hypothetical occurrence of an object with a Gaussian  $k$ , and as according to Eqn. (5) the number of observations caused by the  $k$ -th object is encoded in the mixture weight  $w_k$ , we can estimate  $p(x_t^k | g_t^k, \theta_t^k)$  as follows: First, we sort all weights  $w_k$  in descending order. Then, we compute the number  $K_S$  of Gaussians that most probably correspond to static objects as

$$K_S = \operatorname{argmin}_l \left\{ \sum_{k=1}^l w_{s(k)} > \rho \right\}. \quad (11)$$

Here,  $s(k)$  is the index of the weight  $w_k$  after sorting and  $\rho \in [0, 1]$  is an environment-dependent parameter that represents a measure for the minimum portion of observations that should be accounted for by static hypotheses. The last  $K - K_S$  Gaussians in the sorted mixture are consequently labeled as dynamic. Thus, the lower the value of  $\rho$ , the more Gaussians are considered to be dynamic. For  $\rho = 0$ , only the most evident Gaussian, i.e. the one with the highest count of observations, remains static. Using Eqn. (11), the dynamics likelihood is approximated as

$$p(\mathbf{x}_t^k | g_t^k, \Theta_t^k) = \begin{cases} 1 & \text{if } s(k) > K_S \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

### 3.3 Computational Complexity

We briefly want to discuss the expected computational costs of the proposed method. We therefore analyze the steps taken in order to update the model parameters  $\Theta_t$  given a new observation  $\mathbf{z}_t$  and for recalculating the state variables  $\mathbf{x}_t$ .

Reestimating the cell mixture is largely dominated by the search for the best candidate distribution and by the pair-wise collapsing of Gaussians. As the average number of Gaussians  $\hat{K}$  contained in the cells varies with the choice of  $p_{new}$ , complexity strongly depends on our prior expectation about the occurrence of world dynamicity. We may however state that the update costs are bound by a recursive collapse of the entire mixture distribution into a single Gaussian density. And hence, Alg. 1 runs in worst-case  $O(\hat{K}^2 \log \hat{K})$ .

Estimating the states requires an additional sorting of the updated cell mixture which takes additional effort in  $O(\hat{K} \log \hat{K})$ .

## 4 Evaluation

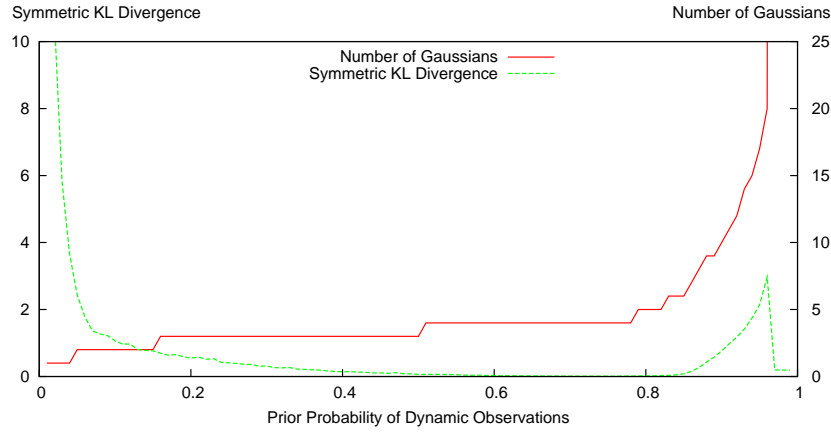
In order to evaluate the approach proposed in this paper, we have conducted several experiments based on both simulated and real-world data. In this section, we present our results and some major insights originating from the analysis of these results.

First, we want to consider the influence of the parameter  $p_{new}$  on the quality of our on-line mixtures estimates. We will then discuss the performance of our method as applied to range observations from different outdoor settings.

### 4.1 Simulations and the Influence of $p_{new}$

To assess the theoretical soundness of our approach, we have initially generated various sets of sample observations from a predetermined ground truth. This ground truth was composed of Gaussian mixture distributions of which the parameters were known. Drawing samples from such mixtures is straightforward, and these samples may directly serve as simulated range measurements to the estimation process.

As a similarity measure between the simulated and the estimated mixture distributions, we have chosen to adapt a variational approximation to the Kullback-Leibler divergence for Gaussian mixture models. This approximation was first proposed in [4] and provides a fairly accurate, closed-form distance function.



**Fig. 1** The influence of the prior probability  $p_{new}$  on the size and the quality of the learned mixture distributions. The graphs display the number of estimated Gaussians and the approximated distance to the ground truth.

Consider two Gaussian mixtures  $\mathcal{M}_a$  and  $\mathcal{M}_b$  with weights  $w_a^k$  and  $w_b^l$ , and densities  $\mathcal{N}_a^k$  and  $\mathcal{N}_b^l$ , respectively. Then, the variational approximation of the distance  $d_{var}(\mathcal{M}_a||\mathcal{M}_b)$  between  $\mathcal{M}_a$  and  $\mathcal{M}_b$  is given by

$$d_{var}(\mathcal{M}_a||\mathcal{M}_b) = \sum_k w_a^k \log \frac{\sum_{k'} w_a^{k'} e^{-d_{KL}(\mathcal{N}_a^k||\mathcal{N}_a^{k'})}}{\sum_l w_b^l e^{-d_{KL}(\mathcal{N}_a^k||\mathcal{N}_b^l)}} \quad (13)$$

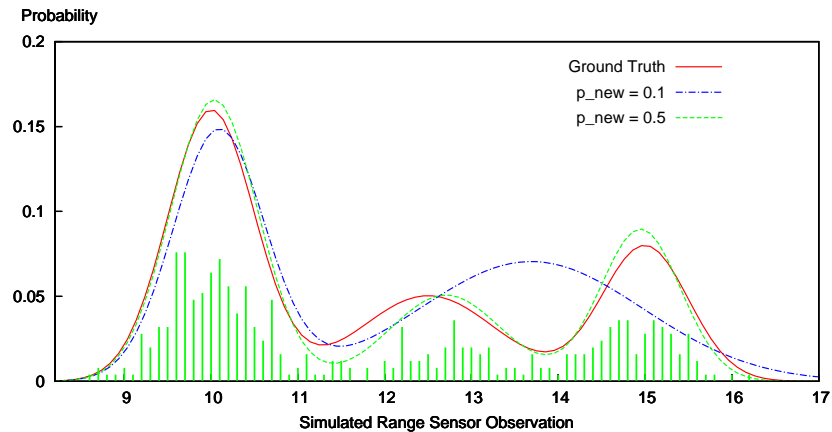
where  $d_{KL}(\mathcal{N}_k||\mathcal{N}_l)$  denotes the Kullback-Leibler divergence between normal densities  $\mathcal{N}_k$  and  $\mathcal{N}_l$ .

The symmetric form of the approximated distance shall then be defined as

$$d_{sym}(\mathcal{M}_a||\mathcal{M}_b) = \frac{d_{var}(\mathcal{M}_a||\mathcal{M}_b) + d_{var}(\mathcal{M}_b||\mathcal{M}_a)}{2} \quad (14)$$

One of the major objectives of this evaluation was to examine the influence of the prior probability  $p_{new}$  on the resulting estimates. In the course of our analysis, we have therefore repeatedly sampled 1000 data points from a ground truth consisting of 4 Gaussians with varying mean and variance parameters. Each of those sample sets was then used to infer a new mixture under the assumption of different values for  $p_{new}$ . The number of Gaussians in the resulting mixtures and the approximated distance between the ground truth and the estimates are illustrated in Fig. 1.

A deeper investigation of the graphs in this figure reveals that it can be divided into four major parts: The Gaussians in the estimates tend to associate easily for very small values of  $p_{new}$ . Under these conditions, the number of independent state hy-



**Fig. 2** The influence of  $p_{new}$  by example. A small prior probability causes estimates to generalize over the ground truth. But for a wide parameter range, a fairly accurate regression of the sample distribution is achieved.

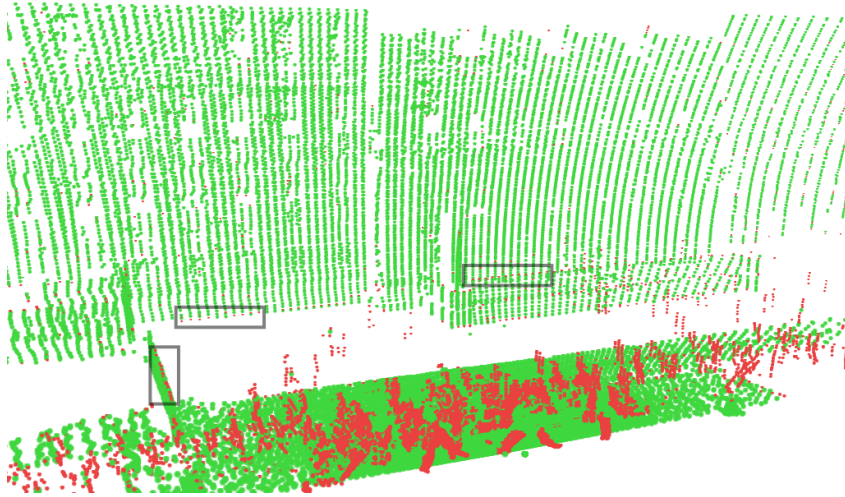
potheses is underestimated, but the model might excellently generalize over ground truth distribution with many individual densities. An example of this behavior is depicted in Fig. 2.

For slightly higher values of  $p_{new}$  up to a probability of about 0.8, the number of Gaussians then remains near constant, whereas the resulting mixture evidently achieves a fairly accurate regression of the sample distribution. This insight is also supported by Fig. 2 where we witness a tight fit between ground truth and estimate.

Assuming values above 0.8,  $p_{new}$  allows Gaussians to only associate if they are very close with respect to our probabilistic distance. This consequently leads to an overfitting behavior, resulting in a higher number of individual densities and decreasing similarity.

If  $p_{new}$  converges towards a probability of 1.0, individual Gaussians cease to associate. The scenario culminates in a trivial situation where each sample is represented by a single Gaussian. In those cases, the distance becomes negligible, but the mixture model completely explodes in complexity.

Our analysis on the influence of  $p_{new}$  on learning mixtures from simulated data advocates important insights pertaining to real-world processes. In order for the estimator to disambiguate between static and dynamic objects, it is important to adjust  $p_{new}$  according to the following criterion: If we expect very little motion in the environment, we will choose small priors such that the model generalizes well even for noisy readings of still surfaces. For widely dynamic environments, we will assume higher probabilities  $p_{new}$  to allow for larger numbers of individual and strongly discriminating hypotheses.



**Fig. 3** Estimation results from an urban scene with pedestrians moving in front of a building. The approach robustly segments dynamic objects (red) from static background surfaces (green). The gray boxes mark problematic edges that cannot be explained by the proposed model.

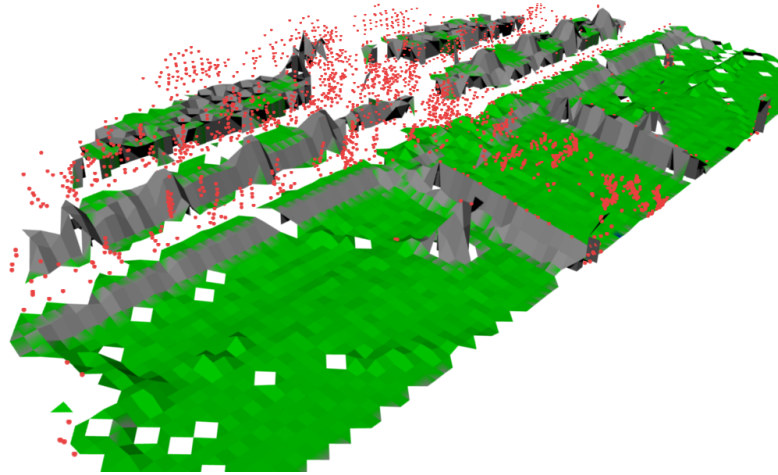
## 4.2 Outdoor Experiments

To evaluate its real-world performance, we have furthermore applied our method to several data sets of outdoor scenarios with varying dynamic properties. Such scenarios involved different background structures as well as a selection of dynamic objects that usually move within urban settings. Amongst these objects, we found pedestrians, cars, trams, and cyclists. We have fixed our nodding range sensor with a typical pitch range of about 45 degrees in positions overlooking extensive areas. The pitch frequency was usually adjusted to about 0.5 Hz. Hence, moving objects appear slightly distorted.

Fig. 3 shows an exemplary outdoor scene and the learned environment representation. The data set used in order to produce these results is composed of continuous scans over a time period of several minutes. For the purpose of visualization, we have decided to obtain a pointcloud representation of the cell mixtures. We therefore depict the mean of a static Gaussian by a green point. Accordingly, dynamic hypotheses are marked in red.

By qualitative visual investigation, we have found that the depicted outdoor results display a robust segmentation of static and dynamic objects. Unfortunately, providing a labeled ground truth explaining our data sets is a difficult challenge that limits the feasibility of a quantitative analysis. Instead, we show an application of our approach to create a 3D grid-based Multi-level Surface (MLS) map [16] in Fig. 4. As we can see, our approach to detect and remove dynamic objects reduces the number of obstacles in the map, represented as non-traversable map cells.





**Fig. 4** Multi-level Surface map created from a different urban scene. The dynamic objects have been removed using the approach presented in this paper. Green cells are classified as traversable. The laser readings that have been classified as dynamic are overlaid to the map as red points.

The attentive observer may have noticed that some of the static regions of our estimates contain small numbers of outliers. The regions of concern are specifically characterized by edges lying in the measurement plane of the range sensor. For those edges, the laser beam has an equally distributed chance of observing a foreground or a background surface. Problematic edges hence constitute a model discontinuity which cannot be explained by our current noise assumptions. Addressing such discontinuities, we therefore propose an alternative implementation of  $p(\mathbf{x}_t | \mathbf{z}_t, \theta_t)$  that takes into account the spatial neighborhood between cell models [13].

## 5 Conclusions

We have presented a novel approach to the difficult problem of detecting dynamic objects from range measurements. As opposed to previous work in the field, our method takes very few assumptions about the structure of the environment. Nevertheless, our estimation algorithm is strictly governed by statistical models and the laws of probability. The outdoor results produced within the scope of this paper appear promising and may directly serve as input to a variety of high-level approaches, such as map building or object tracking.

## 6 Acknowledgements

This work was funded within the EU Projects EUROPA-FP7-231888 and BACS-FP6-IST-027140.

## References

1. P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. In *Proc. of Robotics: Science and Systems (RSS)*, 2005.
2. C. Bishop et al. *Pattern Recognition and Machine Learning*. Springer New York:, 2006. pages 94-97.
3. W. Burgard, C. Stachniss, and D. Hahnel. Mobile robot map learning from range data in dynamic environments. *Springer Tracts in Advanced Robotics*, 35, 2007.
4. J. Hershey and P. Olsen. Approximating the Kullback Leibler divergence between Gaussian mixture models. In *Proc. of The International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, pages 317–320, 2007.
5. S. Hou and A. Galata. Robust estimation of Gaussian mixtures from noisy input data. In *Proc. of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.
6. B. Jensen, R. Philippsen, and R. Siegwart. Motion detection and path planning in dynamic environments. In *Workshop Proceedings Reasoning with Uncertainty in Robotics, International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.
7. R. Kaestner, S. Thrun, M. Montemerlo, and M. Whalley. A non-rigid approach to scan alignment and change detection using range sensor data. In *Field and Service Robotics, Springer Tracts in Advanced Robotics*, volume 25/2006, pages 179–194. Springer Press, 2006.
8. D. Lee, J. Hull, and B. Erol. A Bayesian framework for Gaussian mixture background modeling. In *Proc. of The IEEE International Conference on Image Processing*, volume 3, pages 973–976, 2003.
9. U. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford University, 2002.
10. M. Luber, K. Arras, C. Plagemann, and W. Burgard. Classifying dynamic objects: An unsupervised learning approach. *Robotics: Science and Systems IV*, page 270, 2009.
11. N. Roy, W. Burgard, D. Fox, and S. Thrun. Coastal navigation: Mobile robot navigation with uncertainty in dynamic environments. In *IEEE International Conference on Robotics and Automation*, pages 35–40. Citeseer, 1999.
12. D. Schulz and W. Burgard. Probabilistic state estimation of dynamic objects with a moving mobile robot. *Robotics and Autonomous Systems*, 34(2-3):107–115, 2001.
13. Y. Sheikh and M. Shah. Bayesian object detection in dynamic scenes. In *Proc. of The IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, page 74, 2005.
14. C. Stauffer and W. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.
15. H. Surmann, A. Nüchter, and J. Hertzberg. An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments. *Journal of Robotics and Autonomous Systems (JRAS)*, 45(3-4), 2003.
16. R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

## Bayesian On-line Learning of Driving Behaviors

Jérôme Maye\*, Rudolph Triebel\*, Luciano Spinello†, and Roland Siegwart\*

\* Autonomous Systems Lab, ETH Zurich, Switzerland

email: {jerome.maye, rudolph.triebel, roland.siegwart}@mavt.ethz.ch

† Social Robotics Lab, University of Freiburg, Germany

email: spinello@informatik.uni-freiburg.de

**Abstract**—This paper presents a novel self-supervised on-line learning method to discover driving behaviors from data acquired with an inertial measurement unit (IMU) and a camera. Both sensors were mounted in a car that was driven by a human through a typical city environment with intersections, pedestrian crossings and traffic lights. The presented system extracts motion segments from the IMU data and relates them to visual cues obtained from camera data. It employs a Bayesian on-line estimation method to discover the motion segments based on change-point detection and uses a Dirichlet Compound Multinomial (DCM) model to represent the visual features extracted from the camera images. By incorporating these visual cues into the on-line estimation process, labels are computed that are equal for similar motion segments. As a result, typical traffic situations such as braking maneuvers in front of a red light can be identified automatically. Furthermore, appropriate actions in form of observed motion changes are associated to the discovered traffic situations. The approach is evaluated on a real data set acquired in the center of Zurich.

### I. INTRODUCTION

The development of intelligent driver assistant systems has become a very active research field in the last years. The large spectrum of potential applications for such systems ranges from automatic warning systems that detect obstacles and dynamic objects over automated parking systems to fully autonomous cars that are able to navigate in busy city environments. One aspect that is of major importance in all these systems is the perception part of the vehicle, i.e., the data acquisition and semantic interpretation of the environment. The major challenges here include the required accuracy of the detection system, the time constraints given by the speed of the vehicle and its implied temporal restrictions on the decision process, as well as the large variability in which potential objects and the environment itself may appear. Especially this latter point poses a significant challenge on the perception task, because standard learning techniques that most often rely on supervised off-line classification algorithms tend to give poor results when the test environment largely differs from the acquired training data. Furthermore, such systems are not capable of adapting to new, unseen situations, which reduces their applicability for long-term use cases.

In this paper, we present a self-supervised on-line learning algorithm that recognizes driving behaviors and predicts appropriate actions accordingly. A driving behavior in our context is defined as a short sequence of actuation commands to the vehicle that typically occur in certain traffic situations.



Fig. 1. Example of a traffic light scenario (label 10) detected by our algorithm. The suggested action is a braking maneuver.

An example is the braking maneuver in front of a red traffic light. In our system, the driving behaviors are observed using an inertial measurement unit (IMU) and a camera while a human is driving the vehicle. Using our approach, the system is able to detect and classify new traffic scenarios and predict appropriate actions based on the driving behaviors learned in earlier stages of the data acquisition process. The principle idea is to first segment the data stream from the IMU into consistent sequences using *change-point detection*, and then relate these motion sequences to visual features observed in the camera data during the corresponding motion. To find the change-points in the motion data, we use an efficient Bayesian approach based on a Rao-Blackwellized particle filter. The visual features are represented in a bag-of-words approach using a Dirichlet Compound Multinomial (DCM) model. The detected motion segments are grouped on-line and without human intervention, according to their similarities in their corresponding visual features. This enables the system to predict new motion commands according to the traffic situation it detects from new camera data. Thus, it predicts a braking maneuver when it encounters enough evidence for a red light in the camera data. Fig. 1 shows a typical output of our algorithm.

The paper is structured as follows. Section II summarizes the previous works related to ours. Section III introduces our Bayesian framework. Section IV describes our motion segmentation method. Section V shows how we model a traffic situation. Section VI demonstrates our action model. Section VII presents experimental results. Section VIII outlines our conclusions and provides some insights for future work.

## II. RELATED WORK

Existing driving behavior models in psychology are largely subjective and based on self-report scales [1]. They are difficult to quantify, because they include many psychological aspects like motivation, or risk assessment. Many works in the intelligent vehicle literature [2], [3], [4], [5] focus on modeling the driver behavior via their steering behavior or road tracking information or desired driver's path as source of behavior's information. Other works recognize driver's intentions via Bayesian reasoning on a complex input including the driver's current control actions and the traffic environment surrounding them [6], [7]. In a previous work [8], we were able to infer an action from a direction sign in an indoor environment with a semi-supervised approach using vision and prerecorded robot actions. We extend this idea to outdoor, remove any supervision, and predict vehicle actions in an on-line fashion. Meyer *et al.* [9] predicted traffic situations using Hidden Markov Models (HMM). They however restricted their situations space by modeling states with respect to surrounding vehicles (distance, speed, bearing) and manually segmented image sequences for initial estimates. In this paper, we exclude any manual intervention in the process and use a more complete set of variables for predicting states. Other works [10], [11] make use of supervised off-line classification methods for learning the relation between driving actions and visual features. The actions are manually annotated and discretized in the training phase. To our knowledge, there has been few research works that combine traffic scenario recognition and action prediction in an on-line and unsupervised fashion.

## III. PROBLEM FORMULATION

Given a vehicle equipped with an Inertial Measurement Unit (IMU) and a monocular camera, we seek to learn the relation between motion and visual data in an on-line and unsupervised manner. We shall follow an entirely probabilistic approach and formulate the problem as the estimation of the joint filtering distribution

$$p(r_t, l_t, \mathbf{a}_t | \mathbf{z}_{1:t}, \mathbf{c}_{1:t}), \quad (1)$$

where  $r_t$  represents the motion segment length at time  $t$ ,  $l_t$  the image label at time  $t$ ,  $\mathbf{a}_t$  the predicted action at time  $t$ ,  $\mathbf{z}_{1:t}$  the IMU measurements up to time  $t$ , and  $\mathbf{c}_{1:t}$  the camera measurements up to time  $t$ .

Assuming  $r_t$  is conditionally independent of  $\mathbf{c}_{1:t}$  given  $\mathbf{z}_{1:t}$ ,  $l_t$  of  $\mathbf{z}_{1:t}$  given  $\mathbf{c}_{1:t}$ , and  $\mathbf{a}_t$  of  $\mathbf{c}_{1:t}$  given  $\mathbf{z}_{1:t}$ , we can decompose (1) into

$$p(r_t, l_t, \mathbf{a}_t | \mathbf{z}_{1:t}, \mathbf{c}_{1:t}) = p(r_t | \mathbf{z}_{1:t})p(l_t | r_t, \mathbf{c}_{1:t})p(\mathbf{a}_t | r_t, l_t, \mathbf{z}_{1:t}). \quad (2)$$

$p(r_t | \mathbf{z}_{1:t})$  corresponds to the motion segmentation of Section IV,  $p(l_t | r_t, \mathbf{c}_{1:t})$  to the traffic situation modeling of Section V, and  $p(\mathbf{a}_t | r_t, l_t, \mathbf{z}_{1:t})$  to the action prediction model of Section VI.

## IV. BAYESIAN ON-LINE SEGMENTATION OF MOTION DATA

Our motion segmentation algorithm is based on *change-point detection*. A change-point is an abrupt variation in the generative parameters of sequential data. An efficient Bayesian on-line method for detecting change-points has been independently proposed by Adams and MacKay [12] and by Fearnhead and Liu [13]. In the following, we first present this method in general and then show how we apply it to the problem of segmenting motion data.

### A. Change-Point Detection

Suppose we are given a time-dependent sequence of observations  $\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_T$ , where the  $\mathbf{z}_t$  can be scalars or vectors. Our goal is to find segments  $s_1, s_2, \dots, s_N$  with  $s_n = [\mathbf{z}_{b_n}, \dots, \mathbf{z}_{e_n}]$ , where  $e_n > b_n$  and  $b_n = e_{n-1} + 1$  for  $n = 1, \dots, N$ . We assume that all data points  $\mathbf{z}_{b_n}, \dots, \mathbf{z}_{e_n}$  of a segment  $s_n$  are independently and identically distributed (i.i.d.) according to a parameterized statistical model  $p(\mathbf{z} | \boldsymbol{\eta}_n)$ . The parameter vectors  $\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_N$  are also assumed to be i.i.d. The computation of the segments is done on-line, i.e., at each time step  $t$  a decision is made whether  $\mathbf{z}_t$  is added to the current segment  $s_n = [\mathbf{z}_{b_n}, \dots, \mathbf{z}_{t-1}]$  or a new segment is started. As shown above, we denote the length of the current segment as  $r_t$ . Thus, after deciding on  $\mathbf{z}_t$ , we have either  $r_t = r_{t-1} + 1$  or  $r_t = 0$  in case we start a new segment.

To determine whether time step  $t$  is a change-point, we analyze the posterior distribution of the segment length conditioned on the data observed so far, i.e.  $p(r_t | \mathbf{z}_{1:t})$ . Using the product rule, this filtering distribution can be written as

$$p(r_t | \mathbf{z}_{1:t}) \propto p(r_t, \mathbf{z}_{1:t}). \quad (3)$$

The joint distribution in (3) can be further expressed as

$$\begin{aligned} p(r_t, \mathbf{z}_{1:t}) &= \sum_{r_{t-1}} p(r_t, r_{t-1}, \mathbf{z}_{1:t}) \\ &= \sum_{r_{t-1}} p(r_t, \mathbf{z}_t | r_{t-1}, \mathbf{z}_{1:t-1}) p(r_{t-1}, \mathbf{z}_{1:t-1}) \\ &= \sum_{r_{t-1}} p(r_t | r_{t-1}) p(\mathbf{z}_t | r_{t-1}, \mathbf{z}_{1:t-1}) p(r_{t-1}, \mathbf{z}_{1:t-1}). \end{aligned} \quad (4)$$

The right-hand side of (4) consists of three terms: the *transition probability*  $p(r_t | r_{t-1})$  of the Markov chain formed by  $r_1, r_2, \dots, r_t$ , the *predictive distribution*  $p(\mathbf{z}_t | r_{t-1}, \mathbf{z}_{1:t-1})$ , and the posterior  $p(r_{t-1}, \mathbf{z}_{1:t-1})$  from the previous time step. We have exploited Markov assumption for the simplifications in (4).

As there are only two possible successor states for  $r_t$ , namely  $r_{t-1} + 1$  or 0, we can model the transition probability using statistical survival analysis, i.e., the segment length can either "survive" or "die". To do this, we define a *survival function*  $S(t)$  as the probability that the current segment is still alive after time step  $t$ . The complement of  $S$  is usually named the *lifetime distribution function*  $F(t) = 1 - S(t)$  and its temporal derivative  $f(t)$  is denoted the *event rate*. Finally, the *hazard function*  $h(t)$  is defined as the event rate conditioned on the survival of the segment at time  $t$ , i.e.

$$h(t) = \frac{f(t)}{S(t)} = \frac{f(t)}{1 - F(t)}. \quad (5)$$

Intuitively,  $h(t)$  represents the probability that the segment dies exactly at the current time instant  $t$ . We can use  $h(t)$  to model the transition probability as

$$p(r_t | r_{t-1}) = \begin{cases} h(r_{t-1} + 1) & \text{if } r_t = 0 \\ 1 - h(r_{t-1} + 1) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

A common approach is to model  $S(t)$  as an exponential function  $S(t) = \exp(-\lambda t)$  with some given rate parameter  $\lambda$ . Then, the hazard function turns into

$$h(t) = \frac{\lambda \exp(-\lambda t)}{\exp(-\lambda t)} = \lambda. \quad (7)$$

Thus, the hazard rate is constant and the process is “memoryless”.

For the computation of the predictive distribution, we can make it dependent only on the last data point  $\mathbf{z}_{t-1}$  since we are doing a sequential update of the parameters. Thus, it can be expressed as  $p(\mathbf{z}_t | r_{t-1}, \mathbf{z}_{t-1})$ . We finally introduce the model parameters  $\boldsymbol{\eta}^{r-1}$  that are learned on the current segment and compute the predictive distribution by marginalizing them out, i.e.

$$p(\mathbf{z}_t | r_{t-1}, \mathbf{z}_{t-1}, \boldsymbol{\psi}^{r-1}) = \int_{\boldsymbol{\eta}^{r-1}} p(\mathbf{z}_t | \boldsymbol{\eta}^{r-1}) p(\boldsymbol{\eta}^{r-1} | r_{t-1}, \mathbf{z}_{t-1}, \boldsymbol{\psi}^{r-1}) d\boldsymbol{\eta}^{r-1}. \quad (8)$$

Here, we have added the prior hyperparameters  $\boldsymbol{\psi}^{r-1}$  for completeness. The integral in (8) can be solved analytically if we model the prior of the parameter vector  $\boldsymbol{\eta}^{r-1}$  as a conjugate to the probability density function  $p(\mathbf{z}_t | \boldsymbol{\eta}^{r-1})$ . Otherwise, this leads to expensive numerical computations. When the terms inside the integral are conjugate models, the marginal distribution is usually a function of the hyperparameters  $\boldsymbol{\psi}^{r-1}$  which can be updated iteratively as data arrives.

### B. Complexity and Approximate Inference

In order to exactly infer the positions of all change-points until time  $t$ , we need to compute and store  $p(r_t | \mathbf{z}_{1:t})$  for  $t$  and all previous time steps. We can then get the Maximum A Posteriori (MAP) estimate of the sequence of segment lengths using the on-line Viterbi algorithm of [13].

Regarding complexity, if we have processed  $n$  data points, the storage of the full posterior distribution has a memory cost of  $O(n^2)$  and  $O(n)$  computational cost. This might be prohibitive for huge datasets. For this reason, the distribution has to be approximated. A simple way sketched in [12] is to discard values where the distribution is significantly low, i.e., lower than a given threshold. However, as we want to accurately estimate our distribution and control the computational costs, we use a *particle filter*. The state-space of  $r_t$  being discrete and the number of successor states being small, we can evaluate all the possible descendants of  $r_t$ .

Indeed, if  $r_t$  takes  $k$  possible values,  $r_{t+1}$  will take  $k + 1$  possible values. At each time step  $t$ , we approximate the posterior distribution with a set  $\{r_t^{(i)}, \boldsymbol{\psi}^{r_t^{(i)}}\}_{i=1}^M$  of  $M$  particles weighted by  $\{w_t^{(i)}\}_{i=1}^M$  with

$$w_t^{(i)} \propto p(\mathbf{z}_t | r_{t-1}^{(i)}, \mathbf{z}_{t-1}, \boldsymbol{\psi}^{r_{t-1}^{(i)}}). \quad (9)$$

In order to limit the number of particles at each time step, we use the Stratified Optimal Re-sampling (SOR) presented in [13], whenever  $M$  gets bigger than our particles number limit  $P$ .

Using this method reduces the memory costs to  $O(n)$  and the computational costs to  $O(1)$ , i.e. constant run-time. We also notice that this particle filter is Rao-Blackwellized [14] and has thus a lower variance since the sampling space of the state is reduced to  $r_t$  and the rest is marginalized out.

### C. Application to Motion Data Segmentation

In our particular case, data comes from an IMU and we consider accelerations in the  $x, y$  axes and the *yaw* rate, with  $x$  pointing forward,  $y$  on the left, and  $z$  upward. We can safely assume that an IMU measurement  $\mathbf{z}_t$  arises from a multivariate normal distribution with mean  $\boldsymbol{\mu}_n$  and covariance matrix  $\boldsymbol{\Sigma}_n$  for segment  $s_n$ . The parameter vector for segment  $r_t$  is thus  $\boldsymbol{\eta}^{r_t} = \{\boldsymbol{\mu}^{r_t}, \boldsymbol{\Sigma}^{r_t}\}$ . In order to solve the integral in (8) analytically, we model the parameter prior as a normal-Wishart distribution which is conjugate to the multivariate Gaussian. This distribution has four hyperparameters  $\boldsymbol{\psi}^{r_t} = \{\kappa^{r_t}, \boldsymbol{\rho}^{r_t}, \nu^{r_t}, \boldsymbol{\Lambda}^{r_t}\}$  that can be updated iteratively as a new data point  $\mathbf{z}_t$  arrives with:

$$\begin{aligned} \kappa^{r_t} &= \kappa^{r_{t-1}} + 1 \\ \boldsymbol{\rho}^{r_t} &= \frac{\kappa^{r_{t-1}} \boldsymbol{\rho}^{r_{t-1}} + \mathbf{z}_t}{\kappa^{r_{t-1}} + 1} \\ \nu^{r_t} &= \nu^{r_{t-1}} + 1 \\ \boldsymbol{\Lambda}^{r_t} &= \boldsymbol{\Lambda}^{r_{t-1}} + \frac{\kappa^{\nu^{r_{t-1}}}}{\kappa^{r_{t-1}} + 1} (\mathbf{z}_t - \boldsymbol{\rho}^{r_{t-1}})(\mathbf{z}_t - \boldsymbol{\rho}^{r_{t-1}})^\top. \end{aligned} \quad (10)$$

In case we start a new segment and  $r_t = 0$ , the hyperparameters are fixed to some prior values  $\boldsymbol{\psi}_0 = \{\kappa_0, \boldsymbol{\rho}_0, \nu_0, \boldsymbol{\Lambda}_0\}$ .

From (10), we can express the parameters of the resulting multivariate normal distribution in (8) as

$$\begin{aligned} \boldsymbol{\mu}^{r_t} &= \boldsymbol{\rho}^{r_t} \\ \boldsymbol{\Sigma}^{r_t} &= (\boldsymbol{\Lambda}^{r_t})^{-1} / \kappa^{r_t}. \end{aligned} \quad (11)$$

Finally, for the computation of the predictive distribution in (8), we approximate the multivariate normal distribution with a Student’s  $t$ -distribution which is known to be more robust to outliers in case of few data points. This distribution converges to the Gaussian when its degrees of freedom go to infinity. We use the number of processed points as the degrees of freedom for the distribution, so as to have a bigger variance at the beginning.

## V. LABELING OF TRAFFIC SITUATIONS

Our aim is to find a label for each segmented motion pattern. This label represents a traffic situation, e.g., a stop or turn condition. Moreover, we are interested in associating two different motion segments to the same label whenever they depict the same traffic situation. In the following, we show how we can integrate this labeling into the on-line framework of Section IV.

## A. Traffic Situation Model

As shown above, we denote the label of a segment  $r_t$  as  $l_t$ . This label can take values in  $\{1, 2, \dots, N\}$  corresponding to  $N$  parametric models  $M_1, M_2, \dots, M_N$ . Each of the  $M_i$  is a generative model  $p(\mathbf{c}_t | \boldsymbol{\eta}_i)$  for a particular traffic situation with parameter vector  $\boldsymbol{\eta}_i$ . At time  $t$ , we estimate the distribution over the known models conditioned on the data seen so far and the segment we are in with Bayes law as

$$\begin{aligned} p(l_t | r_t, \mathbf{c}_{1:t}) &\propto p(\mathbf{c}_t | l_t, r_t, \mathbf{c}_{1:t-1})p(l_t | r_t, \mathbf{c}_{1:t-1}) \\ &= p(\mathbf{c}_t | l_t, r_t, \mathbf{c}_{t-1})p(l_t | r_t, \mathbf{c}_{1:t-1}). \end{aligned} \quad (12)$$

For the prior part in (12), we use the posterior of the previous time step, that is  $p(l_t | r_t, \mathbf{c}_{1:t-1}) = p(l_{t-1} | r_{t-1}, \mathbf{c}_{1:t-1})$ . If we are in a new segment with  $r_t = 0$ , we set the prior to a uniform distribution over the known models, i.e.,  $p(l_t = 1 : N | r_t, \mathbf{c}_{1:t-1}) = \frac{1}{N}$ . The likelihood part in (12) is computed with the model probability density function  $p(\mathbf{c}_t | \boldsymbol{\eta}_i)$  in the same fashion as in (8), i.e., using a conjugate prior with hyperparameters  $\boldsymbol{\psi}_i$  as will be detailed below. Furthermore, we have only kept the dependency on the last data point  $\mathbf{c}_{t-1}$  since we update the parameters iteratively.

As we want to be able to discover new traffic situations on-line, we have to state if the current data  $\mathbf{c}_t$  is unlikely to come from any of the  $N$  known models so far. We use Bayesian hypothesis testing and compute the *Bayes factor* [15] for all the models  $M_i$  against an alternative model:

$$B = \frac{p(\mathbf{c}_t | l_t = i, \boldsymbol{\psi}_i)}{p(\mathbf{c}_t | l_t, r_t, \mathbf{c}_{t-1}, \boldsymbol{\psi}^{r_{t-1}})}, \quad (13)$$

where  $\boldsymbol{\psi}^{r_{t-1}}$  are the hyperparameters learned over the current segment  $r_{t-1}$ .

The value  $B$  in (13) indicates our confidence in the model  $M_i$  and we compare it to a threshold  $\xi$  for the decision. In case all models are rejected, we create a new instance  $M_{N+1}$  with hyperparameter vector  $\boldsymbol{\psi}^{r_{t-1}}$ , set  $p(l_t = N + 1 | r_t, \mathbf{c}_{1:t}) = p_{new}$ , and  $p(l_t = 1 : N | r_t, \mathbf{c}_{1:t}) = (1 - p_{new})/N$ . We finally update the hyperparameters  $\boldsymbol{\psi}_i$  of model  $M_i$ , such that  $i = \arg \max_{j=1:N} p(l_t = j | r_t, \mathbf{c}_{1:t})$ , with  $\mathbf{c}_t$ .

From an implementation point of view, we attach the distribution (12), the hyperparameters  $\boldsymbol{\psi}^{r_{t-1}}$ , and the incremental set of known models  $M_i$  to each particle. Thus, our system is able to learn new traffic situations on-line and refine its knowledge over previously visited scenes.

## B. Measurements Representation

We represent images using the widely adopted *bag-of-words* model [16]. In the document modeling formulation, text documents are represented as histograms of word counts from a given dictionary. This model can be easily applied to computer vision tasks, words being replaced by features and text documents by images.

We use Scale-Invariant Feature Transform (SIFT) [17] descriptors computed at Difference of Gaussians (DoG) keypoints. SIFT descriptors have been shown to be highly discriminative for object recognition. Although some authors claim that they obtain significantly better results with dense grid representations [18], DoG interest points are more suitable for our purpose. Indeed, we are not interested in capturing uniform regions such as sky, but rather focused on objects.  $N$  images are randomly selected from the entire dataset to build a *codebook* or dictionary of features using K-means clustering. Each feature of an image is then assigned to the nearest *codeword* of the dictionary and we can therefore build a convenient histogram representation.

The link between *bag-of-features* models in computer vision and *bag-of-words* models in text document modeling is intuitive. We can therefore use the generative model of [19] to represent an image in a probabilistic manner as was already proposed in [20]. Image histograms are modeled with a Dirichlet Compound Multinomial (DCM), also known as multivariate Polya distribution. The DCM combines a multinomial model and a Dirichlet prior, and provides an analytical solution to the marginalization of the multinomial parameters [21]. The multinomial distribution  $p(\mathbf{c}_t | \boldsymbol{\theta})$  has parameters  $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_K]$ , corresponding to  $\boldsymbol{\eta}$  above. The Dirichlet prior  $p(\boldsymbol{\theta} | \boldsymbol{\alpha})$  has hyperparameters  $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \dots, \alpha_K]$ , corresponding to  $\boldsymbol{\psi}$  above. The likelihood part of (12) can now be formulated as

$$\begin{aligned} p(\mathbf{c}_t | l_t, r_t, \mathbf{c}_{t-1}, \boldsymbol{\alpha}^{r_{t-1}}) &= \\ \int_{\boldsymbol{\theta}^{r_{t-1}}} p(\mathbf{c}_t | \boldsymbol{\theta}^{r_{t-1}})p(\boldsymbol{\theta}^{r_{t-1}} | l_t, r_t, \mathbf{c}_{t-1}, \boldsymbol{\alpha}^{r_{t-1}})d\boldsymbol{\theta}^{r_{t-1}} &= \\ \frac{n!}{\prod_{k=1}^K n_k!} \frac{\Gamma(\boldsymbol{\alpha}^{r_{t-1}})}{\Gamma(n + \boldsymbol{\alpha}^{r_{t-1}})} \prod_{k=1}^K \frac{\Gamma(n_k + \alpha_k^{r_{t-1}})}{\Gamma(\alpha_k^{r_{t-1}})}, \end{aligned} \quad (14)$$

where  $\Gamma(\cdot)$  is the Gamma function,  $n_k = \mathbf{c}_t(k)$ ,  $n = \sum_{k=1}^K n_k$ ,  $\boldsymbol{\alpha}^{r_{t-1}} = \sum_{k=1}^K \alpha_k^{r_{t-1}}$ , and we have added the hyperparameters  $\boldsymbol{\alpha}^{r_{t-1}}$  in the conditional.

For the iterative update of the hyperparameters  $\boldsymbol{\alpha}^{r_t}$ , we can use the simple rule

$$\boldsymbol{\alpha}^{r_t} = \boldsymbol{\alpha}^{r_{t-1}} + \mathbf{c}_t. \quad (15)$$

In case we start a new segment and  $r_t = 0$ , the hyperparameters are fixed to some prior values  $\boldsymbol{\psi}_0 = \{\alpha_0\}$ .

## VI. ACTION MODEL

We want to estimate the posterior probability distribution over actions conditioned on the current traffic situation and segment. To this end, we closely follow the strategy of

Section V. To each of the traffic situation model  $M_i$  is associated an action model  $A_i$ , which we fit with a Gaussian Mixture Model (GMM). For the same traffic situation  $M_i$ , we are able to model several possible behaviors corresponding to the different Gaussian components. For instance, when we reach a traffic light, we might brake when the light is red and continue when it is green. Moreover, a driver does not always brake or accelerate exactly the same manner every time. Finally, our system can adapt to new drivers. We can thus formulate the following model that we estimate and update at each time step:

$$p(\mathbf{a}_t | r_t, l_t, \mathbf{z}_{1:t}, \boldsymbol{\psi}^{\mathbf{x}_t}) = \sum_{\mathbf{x}_t} p(\mathbf{x}_t) p(\mathbf{a}_t | \mathbf{x}_t, r_t, l_t, \mathbf{z}_{1:t}, \boldsymbol{\psi}^{\mathbf{x}_t}), \quad (16)$$

where  $\mathbf{x}_t$  is a  $K$ -dimensional vector with a single one at the position  $k$  encoding for the  $k$ -th Gaussian and zeros elsewhere,  $p(\mathbf{x}_t)$  is the prior for selecting a particular Gaussian component  $\mathbf{x}_t$ , and  $p(\mathbf{a}_t | \mathbf{x}_t, r_t, l_t, \mathbf{z}_{1:t}, \boldsymbol{\psi}^{\mathbf{x}_t})$  is a Gaussian distribution with hyperparameters  $\boldsymbol{\psi}^{\mathbf{x}_t}$ . In a similar fashion as in Section IV, we have marginalized out the parameters of the Gaussian and are thus able to iteratively update the hyperparameters. The prior distribution  $p(\mathbf{x}_t)$  is defined as

$$p(\mathbf{x}_t(i) = 1) \propto n_t^i, \quad (17)$$

where  $n_t^i$  is the sum of the points assigned to Gaussian component  $i$ .

Upon reception of a new data point  $\mathbf{z}_t$ , we compute the Bayes factor for all the Gaussian components  $\mathbf{x}_{t-1}$  of the model  $l_t$  and compare it to  $\varepsilon$ . If all the components are rejected, a new Gaussian is created with hyperparameters  $\boldsymbol{\psi}_0$ . We update the hyperparameters of the most likely Gaussian component with the rule from (10) and increment the corresponding  $n_t^i$ .

From an implementation point of view, the distribution (16) and the learned GMM  $A_i$  are attached to the particle filter of Section IV.

## VII. EXPERIMENTS

In order to evaluate the approach proposed in this paper, we have collected a dataset with a car in an urban setting. Our car is equipped with a Sony XCD-SX910 camera recording 1280x960 images at 3.75 frames per second and an XSens MTi IMU running at 100 Hz with  $x$  pointing forward,  $y$  to the right, and  $z$  upward. The sequence contains 8218 images and lasts around 40 minutes. We have encountered different scenes comprising of traffic lights, crosswalks, or changes of speed limit. We have driven in a loop so as to come several times in the same situation and thus have an estimation of the quality of our solution.

### A. Simulation

Since it is easier to have a ground truth on simulated data and hence validate our approach, we first display an experiment of the whole algorithm on synthetic data. For visualization purposes, we have simulated IMU data with an

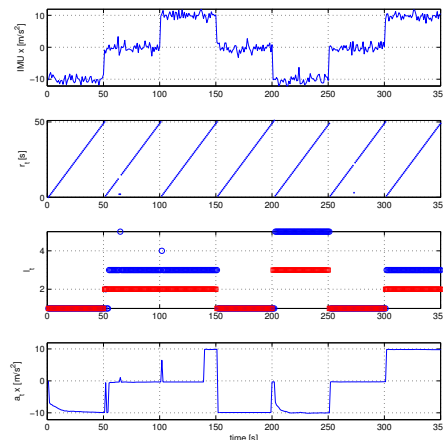


Fig. 2. Simulation results of the algorithm. From top to bottom, the plots display the simulated IMU data  $\mathbf{z}_t$ , the inferred segment lengths  $r_t$ , the inferred labels  $l_t$  (blue circle) with ground truth (red square), and the MAP estimate for the action  $\mathbf{a}_t$ .

univariate normal distribution and have introduced change-points every 50 data points. We have randomly generated 3 different  $\alpha_i$  with  $K = 256$  coding for the traffic situations. Although the algorithm starts with no prior knowledge, we could also start with previously learned models  $M_i$  and  $A_i$ .

Fig. 2 depicts the output of the simulation and demonstrates the pertinence of our method. We display the MAP solution for (16) on the bottom plot and thus the prediction reflects the Gaussian with the maximum number of data points. At time step 100, a new Gaussian with mean 10 is created for label 2. It becomes the MAP only at time step 300 after accumulating enough evidence. Even though the label numbers  $l_t$  differ from the ground truth, they are actually correctly estimated since the induced partition is equivalent.

### B. Motion Segmentation

We have estimated the quality of our motion segmentation algorithm from Section IV on real-world data and performed inference on the final posterior distribution (4) to get the optimal sequence of segment lengths which represents our motion segments. We set the hazard rate to  $\lambda = 1/10$ , the number of particles to  $P = 100$ , and the prior hyperparameters of the normal-Wishart to  $\kappa_0 = 1, \rho_0 = \mathbf{0}, \nu_0 = 3, \Lambda_0 = \mathbf{I}$ . We only considered IMU data at 10 Hz.

Fig. 3 shows the extracted motion segments along with the corresponding IMU data. Our algorithm identified 165 segments which are validated by visual inspection of the IMU data. Furthermore, the segmentation has been compared to a manual annotation of our image sequence and exhibited an accuracy of approximately 92%. For the labeling of the change-points, we have watched the video and noted down where we would expect a change of driving behavior. The parameter  $\lambda$  controls the false positives/negatives rates.

### C. Traffic Situation Labeling and Recognition

We have evaluated the technique presented in Section V and performed inference on (12) to obtain the most likely

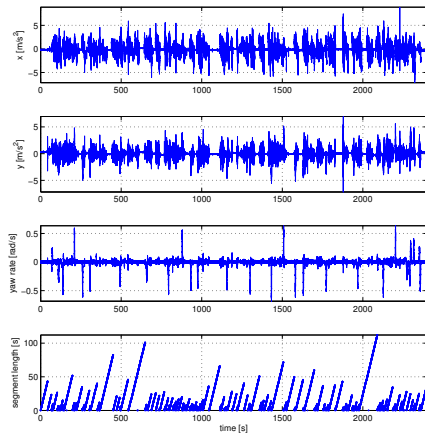


Fig. 3. Optimal motion segmentation from IMU data. The three top plots are the IMU raw values over time. The bottom plot depicts the motion segments discovered by our algorithm.

label for a scene. In a first phase, we have collected a subset of images from traffic lights, yield signs, and pedestrian crossings. Models  $M_i$  were learned on these images using (15) and frozen during the evaluation. In a second phase, we have started the algorithm with no prior models. The dictionary was created from a set of  $N = 400$  randomly picked images and the SIFT features quantized into  $K = 256$  visual words. The prior hyperparameters of the Dirichlet distribution were set to  $\alpha_0 = \mathbf{1}$ .

In the supervised case, we have manually annotated the image sequence and compared the resulting labeling to the ground truth. We obtained an accuracy of 93% for traffic light scenes, 99% for yield scenes, and 91% for pedestrian crossings scenes. We except these results to drop slightly in a previously unseen environment. In the unsupervised case,  $\xi$  acts as a concentration parameter, i.e., it controls the tendency to create new classes. The final labeling is challenging to evaluate. Two traffic lights scenes might for instance get different labels without interfering into the final action prediction. With  $\xi = 200$ , our algorithm discovered 15 different traffic situations and was able to re-associate correctly to the same labels in the different runs of our driving loop.

#### D. Action Prediction

The strategy presented in Section VI is relatively straightforward to evaluate, since predictions can be compared to incoming IMU data. We set the threshold for creating a new Gaussian to  $\epsilon = 5$  and inferred on (16). Our algorithm performed accurately in predicting the driving actions.

### VIII. CONCLUSION

In this paper, we have presented a novel approach for on-line learning of driving behaviors in an unsupervised fashion. To this end, we have developed an entire Bayesian framework that is able to learn and adapt to new traffic situations and drivers. Visual traffic situations models have been modeled probabilistically from image streams and associated

to motion segments from IMU data. Potential actions related to a particular traffic scene are jointly learned, providing predictions in unseen environments. Our system is suitable for lifelong learning since it is able to continuously update its models. We quantified the usefulness and the performance of this approach on a challenging urban dataset.

In a further work, we aim at improving our image representation with a more sophisticated model in order to determine which object in the scene induces an action. Action modeling at a higher level could be represented with a Hidden Markov Model (HMM).

#### ACKNOWLEDGMENT

This work has partly been supported by the EC under FP7-231888-EUROPA and by the DFG under SFB/TR-8.

#### REFERENCES

- [1] T. A. Ranney, "Models of driving behavior: A review of their evolution," *Accident Analysis & Prevention*, vol. 26, no. 6, pp. 733–750, Dec. 1994.
- [2] E. Donges, "A two-level model of driver steering behavior," *J. Human Factors and Ergonomics Soc.*, vol. 20, no. 6, pp. 691–707, Dec. 1978.
- [3] D. T. McRuer, "Human dynamics in man-machine systems," *Automatica*, vol. 16, no. 3, pp. 237–253, May 1980.
- [4] R. A. Hess and A. Modjtahedzadeh, "A control theoretic model of driver steering behavior," *IEEE Control. Syst. Mag.*, vol. 10, no. 5, pp. 3–8, 1990.
- [5] C. C. MacAdam, "Application of an optimal preview control for simulation of closed-loop automobile driving," *IEEE Trans. Syst. Man Cybern.*, vol. 11, no. 6, pp. 393–399, Jun. 1981.
- [6] N. Oliver and A. P. Pentland, "Graphical models for driver behavior recognition in a smart car," in *Proc. IEEE Intel. Veh. Symp.*, 2000.
- [7] A. Liu and D. Salvucci, "Modeling and prediction of human driver behavior," in *Proc. 9th Int. Conf. Human-Comput. Interaction*, 2001.
- [8] J. Maye, L. Spinello, R. Triebel, and R. Siegwart, "Inferring the semantics of direction signs in public places," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2010.
- [9] D. Meyer-Delius, C. Plagemann, and W. Burgard, "Probabilistic situation recognition for vehicular traffic scenarios," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009.
- [10] M. Heracles, F. Martinelli, and J. Fritsch, "Vision-based behavior prediction in urban traffic environments by scene categorization," in *Proc. Brit. Mach. Vis. Conf.*, 2010.
- [11] N. Pugeault and R. Bowden, "Learning pre-attentive driving behaviour from holistic visual features," in *Proc. Europ. Conf. Comput. Vis.*, 2010.
- [12] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," University of Cambridge, Cambridge, UK, Tech. Rep., 2007.
- [13] P. Fearnhead and Z. Liu, "On-line inference for multiple changepoint problems," *J. Roy. Stat. Soc. Series B*, vol. 69, no. 4, pp. 589–605, Apr. 2007.
- [14] G. Casella and C. P. Robert, "Rao-Blackwellisation of sampling schemes," *Biometrika*, vol. 83, no. 1, pp. 81–94, Jan. 1996.
- [15] R. E. Kass and A. E. Raftery, "Bayes factors," *J. Americ. Stat. Assoc.*, 1995.
- [16] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2003.
- [17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.
- [18] L. Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pat. Recog.*, 2005.
- [19] R. E. Madsen, D. Kauchak, and C. Elkan, "Modeling word burstiness using the Dirichlet distribution," in *Proc. Int. Conf. Mach. Learn.*, 2005.
- [20] A. Ranganathan and F. Dellaert, "Bayesian surprise and landmark detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009.
- [21] T. P. Minka, "Estimating a Dirichlet distribution," Microsoft Research, Tech. Rep., 2003.



## Parsing Outdoor Scenes from Streamed 3D Laser Data Using Online Clustering and Incremental Belief Updates

Rudolph Triebel<sup>a</sup>   Rohan Paul<sup>a</sup>   Daniela Rus<sup>b</sup>   Paul Newman<sup>a</sup>

<sup>a</sup> Mobile Robotics Group, Oxford University, UK  
{rudi, rohanp, pnewman}@robots.ox.ac.uk

<sup>b</sup> Computer Science and Artificial Intelligence Laboratory  
Massachusetts Institute of Technology, USA  
rus@csail.mit.edu

### Abstract

In this paper, we address the problem of continually parsing a stream of 3D point cloud data acquired from a laser sensor mounted on a road vehicle. We leverage an online star clustering algorithm coupled with an incremental belief update in an evolving undirected graphical model. The fusion of these techniques allows the robot to parse streamed data and to continually improve its understanding of the world. The core competency produced is an ability to infer object classes from similarities based on appearance and shape features, and to concurrently combine that with a spatial smoothing algorithm incorporating geometric consistency. This formulation of feature-space star clustering modulating the potentials of a spatial graphical model is entirely novel. In our method, the two sources of information: *feature similarity* and *geometrical consistency* are fed continually into the system, improving the belief over the class distributions as new data arrives. The algorithm obviates the need for hand-labeled training data and makes no a-priori assumptions on the number or characteristics of object categories. Rather, they are learnt incrementally over time from streamed input data. In experiments performed on real 3D laser data from an outdoor scene, we show that our approach is capable of obtaining an ever-improving unsupervised scene categorization.

### Introduction

Obtaining semantic knowledge about the environment from a stream of data is a key component in any mobile robotic system. Despite the availability of many useful and efficient methods aiming to solve the robot perception task, at least two main challenges still remain: to relieve the requirement of vast amounts of human-labeled training data and to build a system that performs the learning task in an ever ongoing way instead of once before system deployment. The latter is often referred to as *life-long learning*, and the former is known as *unsupervised learning*. In this paper, we present a solution to both problems by means of an algorithm that continuously interprets a stream of 3D point cloud data acquired from a laser sensor that is mounted on a mobile robot platform. The two major components of our system are an on-line clustering algorithm and a spatial smoothing algorithm

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

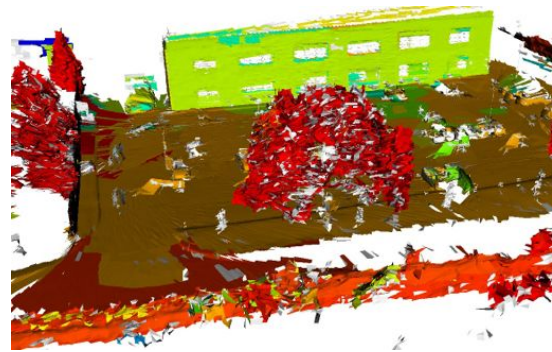


Figure 1: Example result of our scene parsing algorithm. Colours represent object categories discovered by the algorithm in a 3D laser scan scene of a car park with a building, trees, a hedge in the front, and ground plane. The algorithm started with a comparably poor categorization using a single point cloud and improved its performance incrementally (result after 17 point clouds is shown).

based on an ever growing undirected graphical model: while the former groups observed parts of the environment according to their similarity and refines that grouping as new data is observed, the latter enforces geometric consistency by probabilistically reasoning on cluster memberships of parts that are physically close to each other. Both algorithms are online in the sense that their internal representations grow and their results are refined with every new data input obtained from the sensors, and these representations are not rebuilt at every time step. Although this is substantially different from the claim that the system runs in real-time – which we explicitly do not make here, the concept of an unsupervised online learning perception algorithm is a novel contribution in the field of life-long learning for robot perception. In our experiments we show that the core computation can be done with comparably few update operations while still obtaining good performance in terms of semantic interpretation of the observed environment.

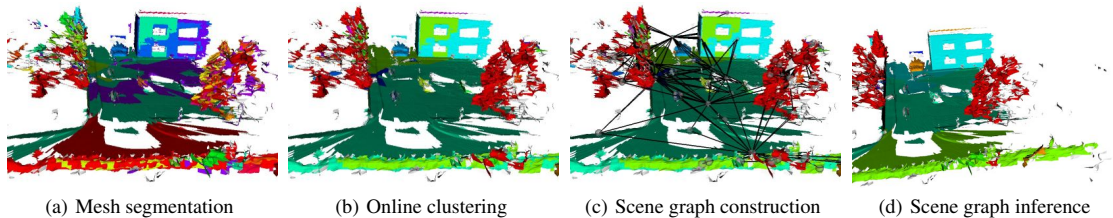


Figure 2: Processing pipeline key steps. (a) Result after segmenting the triangle mesh. Each color represents a different segment. (b) Result after online clustering in feature space. Each color represents a different feature cluster. (c) Construction of the scene graph. Nodes are centers of oriented bounding boxes (OBBs) around each segment. Edges connect segments with overlapping OBBs. (d) Result after inference in the scene graph. The class label distribution is smoother compared to (b), as can be seen, e.g., in the upper left corner of the building.

### Related work

Some approaches for unsupervised object discovery have been presented earlier [Endres, Plagemann, and Stachniss, 2009; Ruhnke et al., 2009; Bagon et al., 2010]. However, these techniques either assume a pre-segmentation of the objects, one object class per image, or a known number of objects and their classes. In contrast, Spinello et al. [2010] proposed an unsupervised discovery algorithm that does not require such assumptions, but instead utilizes the regularity of patterns in which the objects appear. However, in general regular patterns such as the locations of windows on a facade are not available, which is why this technique is not appropriate in our case. Cho, Shin, and Lee [2010] developed a method to detect and segment similar objects from a single image by growing and merging feature matches. Triebel, Shin, and Siegwart [2010] presented a method to discover objects in indoor scenes without hand-labeled training data. Similar to that approach, we also use clustering and probabilistic reasoning, but our approach is conceptually an online learner, where both the clustering and the reasoning part are performed using incremental update steps rather than batch processing at every point in time. Furthermore, Maye et al. [2011] use online unsupervised change-detection and Bayes filtering to discover driving behaviours from streamed IMU and camera data.

### Algorithm Overview

Given a sequence of 3D range scans, our task is to automatically label the scenes without prior training and with a model representation that is refined and improved during operation. We note that in our unsupervised learning framework, instances of classes cannot be *detected*, because no class model is given explicitly. The existence and type of an instance must be *discovered* or *inferred* by accumulating evidence via appearance similarity and spatial coherence patterns from data. Therefore, we propose a framework that combines online feature-space clustering with an incremental version of a spatial smoothing algorithm to obtain and improve geometric consistency as new data is observed. At each time step, when a new 3D range scan is available, our system repeats the following major three stages (see also Fig. 2) which are then described in detail in the next section.

- First, the obtained point cloud is converted into a triangle mesh, and a low-level segmentation is applied to the mesh. The resulting segments contain more information than single scan points or triangles.
- Next, each segment is described by a set of features such as shape and orientation, and the feature vectors are fed into an online clustering algorithm which accumulates information about the segments' similarities over time by refining and extending the current clustering based on the new observations.
- An undirected graphical model named the *scene graph* is refined and extended with the new observations. The scene graph poses geometrical constraints on the discovered class labels and reduces inconsistencies caused by different labelings for overlapping segments.

### Online Preprocessing Steps

The processing pipeline begins by creating a triangular mesh for the incoming 3D point cloud according to the scan manifold order, followed by a segmentation using a variant of the algorithm of Felzenszwalb and Huttenlocher [2004], where the dot product of the normal vectors corresponding to two adjacent triangles is used as a dissimilarity measure. This results in segments with a consistent distribution of normal vectors, representing for example consistently flat or round-shaped surface patches. Then, for each resulting mesh segment, a number of feature vectors based on samples on the surface is computed and then stacked together into one long feature vector. In particular, we compute *spin images* [Johnson, 1997] per segment and use the mean spin image as one feature vector. Furthermore, we compute three kinds of *shape distributions* [Osada et al., 2002], i.e. histograms over unary or binary functions applied to the samples on the mesh surface. For the first kind, we use the Euclidean distance between the two samples, for the second we use the dot product of the corresponding normal vectors, and for the last we use the unary function of the elevation angle of the normal vector. Finally, we compute *shape factors* [Westin et al., 1997] for each mesh segment, i.e. the fractions  $\frac{e_1}{e_2}$ ,  $\frac{e_1}{e_3}$ , and  $\frac{e_2}{e_3}$  of the three eigenvalues  $e_1, e_2, e_3$  of the scatter matrix computed for the segment. As mentioned, all feature vectors use samples on the surface of the mesh segment. To obtain invariance

to the sensor's variable sample density, we re-sample points uniformly on the mesh surface and use them for the feature extraction.

In addition to the feature vectors, we compute an Oriented Bounding Box (OBB)  $B_i$  around each mesh segment. The three main axes of  $B_i$  are determined by the eigen vectors of the segment's scatter matrix, and the dimensions of the box are chosen such that the segment fits tightly into it. The OBBs will be used later to find mesh segments that are close to each other. We do this by defining a distance measure based on the amount of overlap between the two corresponding OBBs. An efficient way to compute this overlap is to draw uniform samples in one OBB and determine the fraction of samples that are contained in the other OBB.

### Star Clustering and Online Organization

We use the star clustering algorithm [Aslam, Pelekhev, and Rus, 2004] to cluster segments obtained from the low-level segmentation based on the shape and appearance features. This algorithm organizes a data corpus into star-shaped clusters based on a given similarity metric. Using the cosine distance metric between feature vectors, the star clustering algorithm guarantees a minimum similarity between any pair of points associated with a cluster. Unlike the  $k$ -means algorithm and many other clustering methods, the star clustering algorithm does not require the number  $k$  of final clusters as an input. Instead, it discovers this number depending on the desired minimum similarity between the elements in the cluster. The star clustering algorithm is computationally very efficient and can be run online. The ability to cluster incrementally makes it especially suitable for our problem setting, where data collection is incremental in nature. This allows the feature space clustering to improve continually as more information about new or existing object categories is encountered by an exploring robot.

Formally, the data corpus is represented as a *similarity graph*,  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{w})$  where the vertices  $\mathcal{V}$  correspond to feature vectors  $\mathbf{f}$  from laser segments, and weights  $\mathbf{w}$  assigned to the edges  $\mathcal{E}$  represent feature similarity. Normalized cosine distance  $d(\mathbf{f}_i, \mathbf{f}_j) = \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\| \|\mathbf{f}_j\|}$  measures the similarity between features  $\mathbf{f}_i$  and  $\mathbf{f}_j$ . The similarity graph  $\mathcal{G}$  can be studied at various pair-wise similarity thresholds  $\sigma$ . The *thresholded graph*  $\mathcal{G}_\sigma$  is obtained from  $\mathcal{G}$  by removing edges with pairwise similarity less than  $\sigma$ , Fig. 3(a). A star-shaped subgraph on  $m + 1$  vertices consists of a star center and  $m$  satellite vertices, where edges exist between the star center and each of the satellite vertices, Fig. 3(b).

The clustering algorithm covers the thresholded graph  $\mathcal{G}_\sigma$  with a minimal cover of maximal star-shaped subgraphs, Fig. 3(c). The number of clusters is naturally induced by the dense cover. The expected size of the star cover on  $n$  vertices is  $O(\log(n))$ . In the star cover obtained, each vertex is adjacent to at least one center of equal or larger degree and no two centers can be adjacent, Fig. 3(c). Satellite segments similar to multiple categories can be associated with multiple star clusters. Each node maps to a vector space with a cosine similarity metric. By examining the geometry of the star-subgraphs in the implied vector space, Fig. 3(b) the ex-

pected similarity between satellite vertices can be obtained as Eq. (1). Here, the center-satellite similarities for any two satellites in the star are represented by  $\cos\alpha_1$  and  $\cos\alpha_2$  and  $\cos\gamma$  represents the expected satellite-satellite similarity. The expected pairwise similarities are high and implying dense accurate clustering in feature space.

$$\cos\gamma \geq \cos\alpha_1 \cos\alpha_2 + \frac{\sigma}{\sigma + 1} \sin\alpha_1 \sin\alpha_2 \quad (1)$$

The algorithm is asymptotically *linear* in the size of the input graph and can be obtained incrementally by re-arranging star centers in the presence of new data points and maintaining the correct star cover, Fig. 4. The number of re-arrangement operations required is usually small, which we verified experimentally. Further, we used an optimized version of the algorithm that saves operations by predicting the future status of a satellite vertex or other star-satellite changes induced by the inserted vertex.

The clustering thus obtained represents initial evidence for object categories based on feature space similarity. Further, since the clusters evolve incrementally with each new input scan, the object categorization improves continually and incrementally with acquired data. Note that categorization obtained till this stage is based on shape and appearance similarity only. Next, we describe a probabilistic graphical model that incorporates the geometric context information and refines online the object categories obtained through feature space star clustering.

### Graph-based Smoothing

As we will show in the experiments, the online star clustering method presented in the previous section yields a mesh segmentation that is fairly good in comparison with a human labelling. However, as it is based on features only, it fails where objects can have different appearances, for example due to occlusions. Therefore, we additionally leverage information obtained from geometric constraints by constructing a simplified Conditional Random Field (CRF), where each node corresponds to a mesh segment and each edge connects segments that are sufficiently close to each other in a geometric sense. The reasoning behind this is that segments that are physically close to each other are more likely to have the same label. Mathematically, for the given set of feature vectors  $\mathbf{f} = \mathbf{f}_1, \dots, \mathbf{f}_N$  we aim to find a set of segment labels  $\mathbf{l} = l_1, \dots, l_N$  that maximise the conditional probability:

$$p(\mathbf{l} | \mathbf{f}) = \frac{1}{Z(\mathbf{f})} \prod_{\mathcal{V}} \varphi(\mathbf{f}_i, l_i) \prod_{(i,j) \in \mathcal{E}} \psi(\mathbf{f}_i, \mathbf{f}_j, l_i, l_j), \quad (2)$$

where  $Z(\mathbf{f})$  is the *partition function*, and the node and edge potentials are defined as:

$$\log \varphi(\mathbf{f}_i, l_i, w_n) = w_n \cdot f_n(\mathbf{f}_i, l_i) \quad (3)$$

$$\log \psi(\mathbf{f}_i, \mathbf{f}_j, l_i, l_j, w_e) = w_e \cdot f_e(\mathbf{f}_i, \mathbf{f}_j, l_i, l_j). \quad (4)$$

Here,  $w_n$  and  $w_e$  are the node and edge weights. The CRF we use is simplified in that the edge feature function does not depend on the node labels and both feature functions are scalars between 0 and 1. As node feature function  $f_n$  we use:

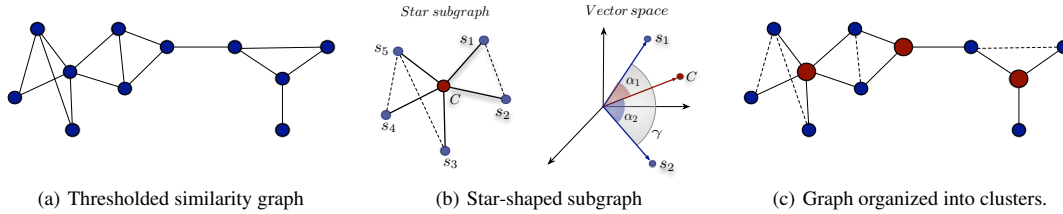


Figure 3: Star Clustering. (a) Similarity graph  $G_\sigma$  where each feature vector  $\mathbf{f}_i$  is a node and edges indicate similarities exceeding threshold  $\sigma$ . (b) Star-shaped subgraph with center  $C$  (red) and five satellite vertices (blue). Each node maps to a vector space with a cosine similarity metric. (c) The graph organized into clusters via a minimal covering with maximal star sub-graphs.



Figure 4: (a) A new data point may introduce additional links in the similarity graph (green) affecting adjacency and hence the validity of the current minimal star cover. (b) Inconsistent stars are re-arranged (green circles). The number of broken stars largely determine the running time. On real graphs, the avg. number of stars broken is usually small (experimentally verified) yielding an efficient incremental approach.

$$f_n(\mathbf{f}_i, l_i) = \begin{cases} 1 & \text{if } \mathbf{f}_i = \mathbf{c}_l \\ \frac{d(\mathbf{f}_i, \mathbf{c}_l)}{\sum_{l: \mathbf{f}_i \leftrightarrow \mathbf{c}_l} d(\mathbf{f}_i, \mathbf{c}_l)} & \text{if } \mathbf{f}_i \leftrightarrow \mathbf{c}_l \\ 0 & \text{else,} \end{cases} \quad (5)$$

where  $\mathbf{c}_k$  denotes the cluster center of cluster  $k$  in  $\mathcal{G}$  and  $\leftrightarrow$  represents a connection by an edge in  $\mathcal{G}$ . The advantage of this node feature function is that in most cases, namely when  $\mathbf{f}_i$  is only connected to one cluster center, a change of cluster membership only affects one node potential. This is important for an efficient online belief update. The simplified edge feature function is defined as  $f_e(\mathbf{f}_i, \mathbf{f}_j) = d_o(B_i, B_j)$ , where  $d_o$  is an estimate of the overlap between the bounding boxes  $B_i$  and  $B_j$  around the mesh segments corresponding to  $\mathbf{f}_i$  and  $\mathbf{f}_j$ .

Usually, the node and edge weights  $w_n$  and  $w_e$  are obtained by maximising Eq.(2) for a given training data set with ground-truth labels  $\mathbf{I}^*$  for each feature vector  $\mathbf{f}_i$ . However, our approach is totally unsupervised, thus a hand-labeled training set is not available. Instead, we fix  $w_n$  to 1 and determine  $w_e$  empirically using an evaluation set. This is possible because the feature functions are particularly simple and only the ratio of  $w_e$  and  $w_n$  is important. In the experimental section, we give more details on choosing  $w_e$ .

### Inference

To perform the inference step in the CRF, we use loopy belief propagation (LBP), [Yedidia, Freeman, and Weiss, 2005]. In general, LBP iteratively computes *messages* defined as label distributions between the nodes in the CRF. First, each message  $m_{ij}$  from node  $i$  to node  $j$  is initialised with the uniform distribution. Then, in each iteration  $\eta$ , the

messages are recomputed based on the node and edge potentials and the messages from the previous iteration  $\eta - 1$ . In our case, we are only interested in the maximum likelihood labelling, and we consider messages to be in log-space for numerical stability. Therefore, we use the *max-sum* rule to compute the messages:

$$m_{ij}^{(\eta)}(l_j) \leftarrow \max_{l_i} \log \varphi_i + \log \psi_{ij} + \sum_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{(\eta-1)}(l_i). \quad (6)$$

Here, we used a short-hand notation for the potentials and  $\mathcal{N}(i)$  denotes all the nodes connected to node  $i$ . Eq. (6) is repeatedly computed until a convergence criterion is met. A good choice is to compute the amount of change of the message and stop iterating when a minimal change  $\xi$  is reached. Then, the belief  $b_i$  at each node is computed as

$$b_i(l_i) \leftarrow \nu(\log \varphi_i + \sum_{j \in \mathcal{N}(i)} m_{ji}(l_i)), \quad (7)$$

where  $\nu$  normalizes the belief so that it is a valid distribution.

### Online Belief Update

Using standard LBP for the inference requires a re-initialization of all messages every time a new scan is observed. Thus, the number of message updates grows at least linearly with the number of totally observed mesh segments. To avoid this, we perform the message update *online*, i.e. we only update messages that got affected by a change in the cluster graph  $\mathcal{G}$  and the messages that depend on them. First, we note that in the CRF, nodes are never removed, and a change in  $\mathcal{G}$  can affect nodes from earlier points in time.

Thus, we need to provide two kinds of update operations: inserting a new node into the CRF, and changing the feature function of an existing node. In the first one, new messages are added, in the second, existing messages need to be re-computed, which is essentially the same as removing the old message and adding a new one. The major problem here is however, that a newly inserted and initialised message has maximal entropy and can not propagate the same amount of information as the existing messages obtained after LBP convergence earlier. This leads to an "over-voting" of the potentials of the new nodes from the existing nodes.

To overcome this problem, we store all messages computed in each LBP iteration in a *message history*  $\mathbf{m}_{ij} = m_{ij}^{(1)}, m_{ij}^{(2)}, \dots$ . Then, before computing (6), we determine the minimal history length  $\mu$  of all message histories  $\mathbf{m}_{ki}$  where  $k \in \mathcal{N}(i) \setminus j$ , and the max-sum-rule turns into

$$m_{ij}^{(\mu+1)}(l_j) \leftarrow \max_i \log \varphi_i + \log \psi_{ij} + \sum_{k \in \mathcal{N}(i) \setminus j} m_{ki}^{(\mu)}(l_i). \quad (8)$$

Some care has to be taken here: to avoid inconsistencies, all messages in the history  $\mathbf{m}_{ij}$  later than  $\mu$  need to be removed. Also, all message histories that depend on  $\mathbf{m}_{ij}$  need to be updated as well. However, the amount of change caused by these updates decreases with every set of successor messages to be updated. To avoid an entire update of all message histories, we determine a threshold  $\epsilon$  and stop updating message histories when the change drops below  $\epsilon$ . Note that this is different from the convergence criterion using  $\xi$ : while  $\epsilon$  determines the number of messages updated after an online update – and thus the performance difference between online and offline processing,  $\xi$  influences the amount of smoothing. By changing  $\epsilon$  gradually towards 0, the online LBP algorithm turns into its standard offline version. A discussion on  $\epsilon$  is provided in the experimental section.

### Cluster Assignment

To be able to perform the online belief update, we need to find all nodes in the CRF, for which the potential  $\varphi_i$  changes after inserting new nodes into  $\mathcal{G}$ . As  $\varphi$  directly depends on the cluster membership of a node, we need to solve the data association between the previous clustering  $C_{t-1} = C_1^{t-1}, \dots, C_n^{t-1}$  and the current clustering  $C_t = C_1^t, \dots, C_n^t$  at every time step and find the elements that changed cluster. Here, we need to consider only those nodes which have been removed from a cluster  $C_i^{t-1}$ , because the others have either been removed themselves from another cluster  $C_j^{t-1}$  or they were added newly to  $C_i^{t-1}$  while growing the cluster graph  $\mathcal{G}$  (in the latter case, no message histories exist, and the update is done as in regular LBP). To assign previous clusters  $C_i^{t-1}$  to current clusters  $C_j^t$ , we therefore define a cost function  $c$  based on the number of removed cluster elements:

$$c(C_i^{t-1}, C_j^t) = L(\zeta(C_i^{t-1}), \zeta(C_j^t)) - I(\zeta(C_i^{t-1}), \zeta(C_j^t)). \quad (9)$$

Here,  $\zeta$  sorts the elements of a cluster with respect to their global element indices,  $L$  is the Levenshtein (edit) distance of two sequences, i.e. the minimal number of deletions, replacements and insertions  $I$  required to change the first se-

quence into the second. Thus,  $c$  computes the minimal number of deletions and replacements of elements in  $C_i^{t-1}$ . The data association between  $C_{t-1}$  and  $C_t$  is then done by minimizing the total association cost between all cluster pairs using an algorithm by Edmonds and Karp [1972].

Then, once a cluster assignment is obtained, all messages that are sent from a node in the CRF, for which the feature vector  $\mathbf{f}_i$  has changed cluster membership, are removed, and new message histories are computed as in Eq. (8).

## Results

To evaluate our approach, we ran experiments on streamed 3D laser range data acquired with an autonomous car. The sensor consists of three SICK LMS-151 laser scanners mounted vertically on a turn table. The rotation frequency was set to 0.1Hz. We drove the car slowly ( $\approx 15\text{km/h}$ ) around our research site. The obtained point cloud data is comparably dense: each point cloud consists of 100,000 to 160,000 points, resulting in a data rate of 10,000 to 16,000 points per second. For evaluation we use qualitative and quantitative measures. The qualitative evaluation is done by visualizing the discovery results with different colors for each category, as already shown in Fig. 1. The quantitative measures are: number of resulting categories, number of update steps, and the entropy-based *v-measure* [Rosenberg and Hirschberg, 2007], which is defined as the harmonic mean of *homogeneity* and *completeness* of the obtained labelling compared to a human-labeled ground-truth.

### Qualitative Evaluation

Fig. 5 shows the results of our scene parsing algorithm over a sequence of time. In the figure, time evolves from the top image row to the bottom row. Each row of images shows the result as it was obtained at a particular time step. For an improved visibility, we visualize the results in two ways: First, we show each clustering result as a colored mesh representation with each color corresponding to a different cluster in the left image of each row. In addition to that, we show meshes for each obtained cluster where the particular cluster is highlighted in red (remaining images of each row). In the example, we used a similarity threshold  $\sigma$  of 0.7 for clustering. This results in a smaller number of clusters and in a slightly worse overall performance of the algorithm compared to the result shown in Fig. 1 (see next section for details). However, it also gives the opportunity to highlight the algorithm's ability to improve its performance over time: As we can see, the number of obtained clusters is very low when the first couple of point clouds are processed. As a result, the labelling is comparably poor, assigning for example the trees and the building to the same category. However, as the algorithm obtains more information about its environment, it increases the number of categories and improves its scene parsing performance: in the bottom image row, a clear distinction between the ground plane, the building and the trees can be seen. To visualize the effect of the graph-based smoothing we show two examples of labelings before and after smoothing in Fig. 9. We can see that the labeling after smoothing is clearly more consistent, visible for example in the hedge (left images) and the building (right images).

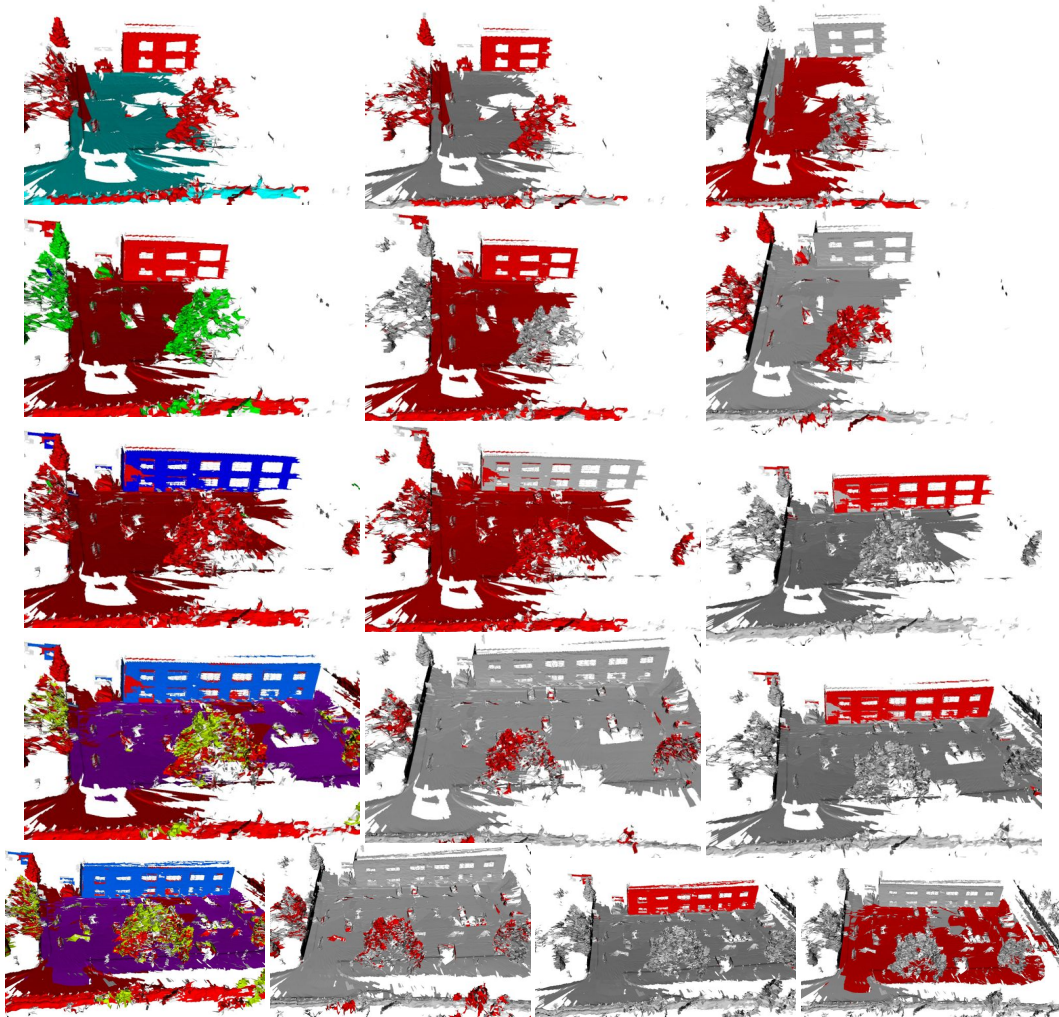


Figure 5: Scene parsing results (best viewed in color). Each row shows the result after processing a different number of point clouds: from the top to the bottom, results are shown after 2, 4, 6, 10, and 17 point clouds. The left image in each row visualizes the obtained scene parsing result with one color for each discovered category. The other images in each row highlight each of the categories with the most elements in red. Note that initially, only two categories are discovered, and the categorization is incorrect (e.g. the tree and the building are assigned the same label). However, as the algorithm evolves over time, the categorization improves, and the number of classes is increased.

Table 1: Statistics for online star clustering.

Threshold, $\sigma$	Data set A			Data set B		
	0.7	0.8	0.9	0.7	0.8	0.9
Num. of clusters	107	580	2699	8	14	84
Graph edges ( $\times 10^5$ )	220.48	98.08	23.40	19.32	13.29	4.02
Insertion/iter (msec)	122.14	85.72	19.98	19.53	31.59	4.54
Insertion/scan (sec)	4.15	2.91	0.67	1.09	1.76	0.25
Insertion time (sec)	1450.85	1018.28	237.44	44.72	72.35	10.41
Stars broken/iter	0.23	0.60	0.55	0.02	0.08	0.21
Stars broken/scan	7.68	20.40	18.60	1.07	4.48	11.75
Total stars broken	2688	7141	6511	44	184	482

### Quantitative Evaluation

To evaluate online star clustering quantitatively, we used two different data sets: data set A consisted of 350 point clouds and resulted in a total of 11879 segments. It was collected on roads surrounding the test site with a vehicle speed  $v$  of about  $40\text{km/h}$  and a scanner rotation frequency of  $1\text{Hz}$ . Data set B is the one mentioned earlier with  $v \approx 15\text{km/h}$  and  $f_r = 0.1\text{Hz}$ , consisting of 41 scans.

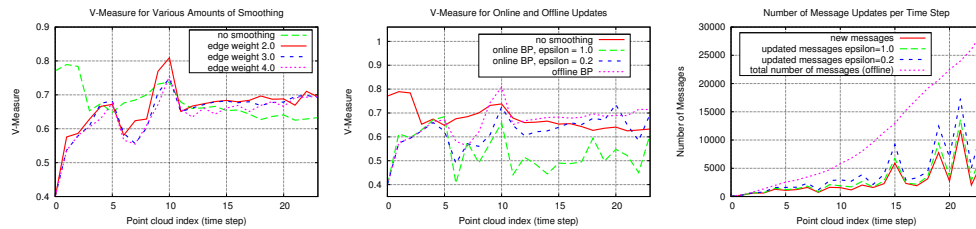


Figure 8: Left: V-Measure compared to ground truth for each time step with different values of  $w_c$  (all offline). In the beginning, smoothing makes the result worse, as the number of clusters is reduced too much. Later, smoothing improves the result. The amount of smoothing has not a strong influence. Middle: Comparison between online and offline LBP. With decreasing value of  $\epsilon$ , online performance approaches the offline quality, with some random effects. Right: Number of messages updated in online and offline LBP. The red line shows the number of new messages introduced at each time step, which is the minimum number of necessary updates. A smaller  $\epsilon$  leads to more message updates.



Figure 9: Effect of smoothing (best viewed in color) on two examples. Left image for each case shows the result using only online star clustering, the right image is the result after applying graph-based smoothing. As can be seen, class labels are clearer and distributed more precisely within each object.

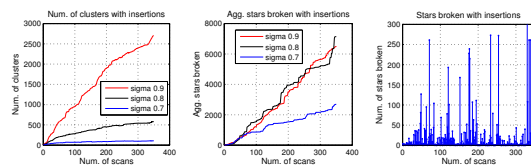


Figure 6: Left: cluster growth. Middle: number of aggregate stars broken for data set A with varying thresholds. This number grows linearly with iterations. The growth rate is small compared to the number of nodes inserted in the graph. Right: number of stars broken in each scan for  $\sigma = 0.8$ . On average this number is low and larger peaks are rare.

Table 1 shows efficiency results for the star clustering algorithm for  $\sigma \in \{0.7, 0.8, 0.9\}$ . Higher values of  $\sigma$  reduce the number of edges in the graph, resulting in an increase of the number of clusters  $N$ . For data set A, the value of  $N$  varied between 107 and 2699, while for data set B it was between 8 and 84. The average number of stars broken during insertion indicates the work done to re-arrange the existing graph. Note that this number is very small. As an example, a total of 2688 stars were broken while incrementally clustering 11897 segments (0.23 stars broken per insertion). The average insertion time per scan ranged from 0.67sec to 4.15sec for data set A and from 0.25sec to 1.76sec for data set B. The insertion time is a function of the graph size, number of stars broken per iteration and the underlying similarity

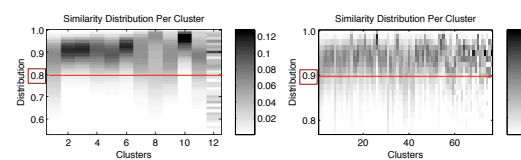


Figure 7: Probability histograms (plotted vertically) for all pair-wise similarities between satellite vertices for each cluster obtained at  $\sigma = 0.8$  (left) and  $\sigma = 0.9$  (right) for Data set B. Red line indicates threshold. The expected similarity values were found higher than or close to  $\sigma$  indicating that star clusters are reasonably dense.

distribution for the data set. The cosine similarity computation time grows linearly with the number of vertices and was 123.34 sec for data set A and 4.46sec for data set B.

Fig. 6 (left) plots the cluster count after each insertion for data set A (results were similar for data set B). Overall, the number of clusters  $N$  increases over time as new segments are added. As the robot explores new environment,  $N$  grows rapidly with newly acquired information. Later, the clusters become increasingly representative of the environment, stabilize, and hence the growth rate shows a decline. For smaller values of  $\sigma$  the saturation effect is more prominent and lies always below the graph with a higher  $\sigma$ . Fig. 6(middle) plots the aggregate number of stars broken during insertion, showing an approximately linear growth

over time with a small growth rate compared to the number of vertices in the graph. It also shows instants when many stars are broken (when the graph re-structures), more commonly observed for the run with the higher value of  $\sigma = 0.8$ , where  $N$  is high compared to lower values of  $\sigma$ . Fig. 6(right) plots the number of stars broken for each scan. On average this number is low and larger peaks are less common.

Fig. 7 illustrates the clustering quality at a specified threshold  $\sigma$  for data set B. For each cluster, the similarity distribution for all pair-wise satellite vertices was plotted along y-axis (bin size 0.0125). Probability histograms were smoothened to account for variable cluster sizes and sampling error as suggested by Cussens [1993]. Clustering at threshold  $\sigma$  ensures that the center-satellite similarities within the star-subgraph are at least  $\sigma$  (by construction). Using Eq. (1) we obtain the expected satellite-satellite similarity as  $\sigma$ , plotted as a horizontal line in Fig. 7. The figure shows that the expected similarity values for clusters was found to be above or close to  $\sigma$ . This indicates that star clusters are reasonably dense and yield clusters with high expected pair-wise satellite similarities. The results were similar for data set A and not included in the interest of space.

Fig. 8 shows a performance comparison with respect to different edge weight parameters  $w_e$  and online message update thresholds  $\epsilon$ . The left and middle figure show the V-measure compared to a hand-labeled ground truth over time. We can see that the performance increases over time and that the online LBP version for  $\epsilon = 0.2$  is only slightly worse than the offline version. However, as shown in Fig. 8(right), there is a significant reduction in the number of updated messages compared to the offline LBP. A smaller  $\epsilon$  improves the V-measure performance, but it also increases the message passing horizon causing more message updates and thus a longer computation time.

### Conclusions

In this paper, we presented an unsupervised scene parsing algorithm that improves its performance during operation time as more information becomes available. We achieve this by combining an online clustering algorithm with an undirected graphical model that grows continually over time. As a result, for each new data frame our algorithm refines its internal representation with only a few update steps as opposed to a complete recomputation required by an offline learner. Additionally, its quantitative performance quickly approaches that of the offline counterpart as new data arrives. We believe that this competency, applied in outdoor environments, constitutes an important step towards life-long autonomy.

### Acknowledgements

This work was partly funded by the EU project EUROPA-FP7-231888. Paul Newman was supported by an EPSRC Leadership Fellowship, EPSRC Grant EP/I005021/1. Daniela Rus was supported for this work in parts by the MAST Project under ARL Grant W911NF-08-2-0004 and ONR MURI Grants N00014-09-1-1051 and N00014-09-1-1031. We thank Benjamin Davis for maintaining the platform used for this research.

### References

- Aslam, J.; Pelekhev, E.; and Rus, D. 2004. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications* 8(1):95–129.
- Bagon, S.; Brostovski, O.; Galun, M.; and Irani, M. 2010. Detecting and sketching the common. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.
- Cho, M.; Shin, Y. M.; and Lee, K. M. 2010. Unsupervised detection and segmentation of identical objects. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.
- Cussens, J. 1993. Bayes and pseudo-Bayes estimates of conditional probabilities and their reliability. In *Proc. of the European Conf. on Machine Learning*, 136–152. Springer.
- Edmonds, J., and Karp, R. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* 19(2):248–264.
- Endres, F.; Plagemann, C.; and Stachniss, C. 2009. Unsupervised discovery of object classes from range data using latent Dirichlet allocation. In *Proc. of Robotics: Science and Systems*.
- Felzenszwalb, P. F., and Huttenlocher, D. P. 2004. Efficient graph-based image segmentation. *Int. J. Comput. Vision* 59(2):167–181.
- Johnson, A. 1997. *Spin-Images: A Representation for 3-D Surface Matching*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon Univ.
- Maye, J.; Triebel, R.; Spinello, L.; and Siegwart, R. 2011. Bayesian on-line learning of driving behaviors. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*.
- Osada, R.; Funkhouser, T.; Chazelle, B.; and Dobkin, D. 2002. Shape distributions. *ACM Trans. on Graphics*.
- Rosenberg, A., and Hirschberg, J. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 410–420.
- Ruhnke, M.; Steder, B.; Grisetti, G.; and Burgard, W. 2009. Unsupervised learning of 3d object models from partial views. In *IEEE Int. Conf. Robotics and Automation (ICRA)*.
- Spinello, L.; Triebel, R.; Vasquez, D.; Arras, K.; and Siegwart, R. 2010. Exploiting repetitive object patterns for model compression and completion. In *European Conf. on Computer Vision (ECCV)*.
- Triebel, R.; Shin, J.; and Siegwart, R. 2010. Segmentation and unsupervised part-based discovery of repetitive objects. *Proc. of Robotics: Science and Systems*.
- Westin, C.-F.; Peled, S.; Gudbjartsson, H.; Kikinis, R.; and Jolesz, F. A. 1997. Geometrical diffusion measures for MRI from tensor basis analysis. In *ISMRM '97*, 1742.
- Yedidia, J.; Freeman, W.; and Weiss, Y. 2005. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on* 51(7):2282–2312.



# Driven Learning for Driving: How Introspection Improves Semantic Mapping

Rudolph Triebel, Hugo Grimmer, Rohan Paul, and Ingmar Posner

**Abstract** This paper explores the suitability of commonly employed classification methods to action-selection tasks in robotics, and argues that a classifier's *introspective* capacity is a vital but as yet largely under-appreciated attribute. As illustration we propose an active learning framework for semantic mapping in mobile robotics and demonstrate it in the context of autonomous driving. In this framework, data are selected for label disambiguation by a human supervisor using uncertainty sampling. Intuitively, an introspective classification framework – i.e. one which moderates its predictions by an estimate of how well it is placed to make a call in a particular situation – is particularly well suited to this task. To achieve an efficient implementation we extend the notion of introspection to a particular sparse Gaussian Process Classifier, the Informative Vector Machine (IVM). Furthermore, we leverage the information-theoretic nature of the IVM to formulate a principled mechanism for forgetting stale data, thereby bounding memory use and resulting in a truly life-long learning system. Our evaluation on a publicly available dataset shows that an introspective active learner asks more informative questions compared to a more traditional non-introspective approach like a Support Vector Machine (SVM) and in so doing, outperforms the SVM in terms of learning rate while retaining efficiency for practical use.

## 1 Introduction

In answering the question ‘where am I?’ roboticists have gone to great lengths to model, manage and, indeed, exploit uncertainty. This, however, is not as yet the case when it comes to asking ‘what is this?’. As we aspire to robust, long-term

---

Rudolph Triebel  
Mobile Robotics Group, Oxford University, UK, e-mail: rudolph.triebel@in.tum.de,  
who has since moved to Computer Vision Group, Technical University of Munich, Germany

Hugo Grimmer, Rohan Paul, and Ingmar Posner  
Mobile Robotics Group, Oxford University, UK, e-mail: {hugo, rohanp, hip}@robots.ox.ac.uk

autonomous operation our systems have to contend with vast amounts of continually evolving, non-i.i.d. data from which information needs to be assimilated. This presents a challenge and an opportunity particularly to the robotics community as here the real cost of failure can be significant. We believe that *realistic* estimates of uncertainty are pivotal to achieving robust and efficient decision making in robotics. In particular, classification as a precursor to *action-selection* seems to be largely disregarded by the community.

We frame our argument in the context of offline semantic mapping. Significant progress in autonomous driving in recent years has inspired a view that successful autonomous operation in complex, dynamic environments critically depends on *a-priori* available semantic maps representing ostensibly permanent aspects of the environment such as lane markings, traffic light positions and road sign information (see, for example, [3, 22]). Owing to their safety-critical nature, these maps are typically created manually for particular routes [5]. This is, of course, an expensive process which scales badly with the number of routes for which autonomous operation is to be provided. Much, therefore, can be gained by reducing human involvement in this process and thus providing a robust and scalable solution.

A prominent approach to tackling such a challenge is that of *active learning*, where classification results are iteratively refined by asking a human supervisor for ground-truth labels in ambiguous cases and incorporating the added information into classifier training. To the best of our knowledge this paper is the first in robotics to present an efficient and scalable active learning framework for the task of offline semantic mapping. Crucially, however, our work is also set apart from the vast majority of the related works in active learning by the unusual stance we take with regards to uncertainty estimates in the system. Commonly, active learning relies on selecting data for human labelling using a variant of *uncertainty sampling*, by which data are selected according to how confident a classifier is in individual predictions (see, for example, [17]).

However, Grimmett *et al.* [7] show that several of the classification frameworks commonly used in robotics are unrealistically overconfident in their assessment of class membership. To characterise this attribute, the authors introduce the notion of the *introspective capacity* of a classification framework: the ability to estimate a classification confidence which realistically reflects how qualified the classifier is to make a particular class decision in each individual test instance. In this paper we show that *introspective classification* harbours significant benefits for active learning as compared to more traditional, non-introspective approaches. In particular, our contributions are

- the application of an active learning framework to semantic mapping in robotics,
- the application of the notion of introspection to the Informative Vector Machine (IVM) [10] as an efficient extension to [7],
- the application of the IVM specifically to achieve *introspective* active learning, which is demonstrated to lead to more effective information extraction over more traditional approaches, and
- the introduction of a principled mechanism for the IVM to *forget* less important data to provide for scalable, life-long active learning on a mobile robot.

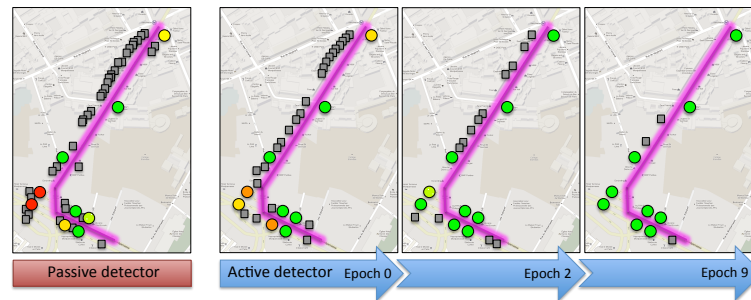


Fig. 1: Active learning in a semantic mapping context. This figure shows semantic maps indicating the positions of traffic lights along a street in Paris. Circles denote the locations of ground-truth traffic lights. The shading encodes the correctness of the classification output as provided by a probabilistic classifier: red denotes a recall of 0 (no detections), and green denotes a recall of 1 for that particular traffic light (all views of that object correctly detected). False positives are shown as grey squares. From left to right, we first see a typical passive detector, followed by our active-learning framework at epochs 0, 2, and 9 respectively. Note that in the active learning setting the shading of the circles progresses from red to green as a greater proportion of traffic lights are correctly detected with increasing confidence. Similarly the number of false positives reduces dramatically. By epoch 2 the active learning framework already outperforms the passive detector. In this paper we show that our formulation of an *introspective* active learning approach provides for more efficient information extraction – and thus a higher learning rate – over conventional active learning approaches. (This figure is best viewed in colour.)

The work presented here first appeared as a workshop paper by the same authors [21]. However, here we offer a more detailed treatment as well as the following significant extensions:

- the introspective capacity of the IVM is established, including the effects of varying the sparsity factor,
- qualitative results are included of when the IVM is confident (correctly and incorrectly) in its classifications, and
- timing information is provided regarding the training of an IVM.

We apply our framework to the detection of traffic lights in a real, third-party vision dataset and demonstrate iteratively improved semantic mapping, which makes efficient use of available label information. A typical qualitative example of our system output is shown in Fig. 1.

## 2 Related Works

Active learning is an established and vibrant field of research spanning a significant number of application domains. Consequently, a variety of methods have been proposed for selecting informative measurements for labelling and/or for incrementally training a learning algorithm. For example, Freund *et al.* [6] propose disagreement among a committee of classifiers as a criterion for active data selection. McCullum and Nigam [12] apply this to text classification using high label inconsistency

as a query criterion coupled with expectation maximisation (EM) for online learning. More recently, Joshi *et al.* [8] address multi-class image classification using SVMs and propose criteria based on entropy and best-versus-second-best (BvSB) measures based on the hyperplane-margin for determining uncertain points. Tong and Koller [19] pick unlabelled data for query based on minimising the version space within a margin-based SVM formulation. Kapoor *et al.* [9] propose an active learning system for object categorization using a GP classifier where data points possessing large uncertainty (using posterior mean and variance) are queried for labels and used to improve classification.

Within the robotics community, active learning and directed information acquisition has received attention in recognition, planning and mapping tasks. For example, Dima *et al.* [4] present unlabelled data filtering for outdoor terrain classification tasks with the aim of reducing the amount of training data to be human-labelled. The approach relies on kernel density estimation over unlabelled data and estimating a “surprise” score for image patches, hence only querying the least likely samples given the density estimate for human labelling. In [14] the authors present a learning approach for continually improving place recognition performance by actively learning an appearance model of a robot’s operating environment. The method uses probabilistic topic models and a measure of perplexity to identify least explained images which further drives retrieval of thematically linked samples leading to an improved workspace representation. Recent work by Tellex *et al.* [18] explores active information gathering for human-robot dialog. The authors formulate an information-theoretic strategy for asking clarifying questions to disambiguate the robot’s belief over the mapping between phrases and aspects of the workspace.

While, to the best of our knowledge, this is the first work in robotics applying active learning to a semantic mapping task, our work is also set apart significantly from prior art in active learning in that we introduce and demonstrate the benefits of efficient *introspective* active learning. In this respect, the work most closely related to ours is that of [9] above, in which an inherently introspective classifier is used but its use is not motivated by its introspective qualities.

### 3 Introspective Classification

The introspective capacity of a classifier characterises its ability to *realistically* estimate the uncertainty in its predictions. Grimmett *et al.* [7] define the introspective capacity as a classifier’s ability to moderate its output by an appropriate measure as to how ‘qualified’ it is to make a call given its own prior experience, usually in the form of training data. The intuition is that test data, which are in some form ‘similar’ to that seen in training, are classified with higher certainty than data which are more dissimilar. This points towards non-parametric approaches potentially being more introspective than parametric ones, as all the training data are available for inference in the former, whereas inference in the latter is based on parametric models learned from the data. Grimmett *et al.* [7] investigated several commonly used

classification frameworks providing probabilistic output and found that a Gaussian Process classifier (GPC) [16] indeed is significantly more introspective than, for example, the more commonly used Support Vector Machine (see, for example, [1]) with a probabilistic calibration (such as, for example, provided by Platt *et al.* [15]).

In [7], this quality is attributed to a GPC's Bayesian treatment of predictive variance. Consider a set of training data  $\{X, \mathbf{y}\}$ , where  $X = \{\mathbf{x}_1, \dots, \mathbf{x}_{|X|}\}$  denotes the set of feature vectors and  $\mathbf{y}$  denotes the set of corresponding class labels. Probabilistic predictions for a test point,  $\mathbf{x}_*$ , are obtained in two steps. First, the distribution over the latent variable corresponding to the test input is obtained by

$$p(f_* | X, \mathbf{y}, \mathbf{x}_*) = \int p(f_* | X, \mathbf{x}_*, f) p(f | X, \mathbf{y}) df, \quad (1)$$

where  $p(f | X, \mathbf{y})$  is the posterior distribution over latent variables. This is followed by applying a sigmoid function  $\sigma(\cdot)$ , which in our implementation is the cumulative Gaussian, and *marginalising* over the latent  $f_*$  to yield the class likelihood  $p(y_* | X, \mathbf{y}, \mathbf{x}_*)$  as

$$p(y_* | X, \mathbf{y}, \mathbf{x}_*) = \int \sigma(f_*) p(f_* | X, \mathbf{y}, \mathbf{x}_*) df_*. \quad (2)$$

It is this *marginalisation* over all models induced by the training set, as opposed to relying on a single *minimisation*-based estimate, which accounts for a more accurate estimate of the inherent uncertainty in class distribution, and therefore endows GP classification with a high introspective capacity.

### 3.1 Efficiency by Sparsification

A key drawback of a GPC is its significant computational demand in terms of memory and run time. This is due to the fact that the GPC maintains a mean  $\boldsymbol{\mu}$ , as well as a covariance matrix  $\boldsymbol{\Sigma}$ , which is computed from a kernel function and has size  $|\mathbf{y}|^2$ . A number of sparsification methods have been proposed in order to mitigate this computational burden. For efficiency, in this work we adopt one such sparsification method: the Informative Vector Machine (IVM) [10]. The main idea of this algorithm is to only use a subset of the training points denoted the *active set*,  $\mathcal{I}$ , from which an approximation  $q(f | X, \mathbf{y}) = \mathcal{N}(f | \boldsymbol{\mu}, \boldsymbol{\Sigma})$  of the posterior distribution  $p(f | X, \mathbf{y})$  is computed. The IVM algorithm computes  $\boldsymbol{\mu}$  and  $\boldsymbol{\Sigma}$  incrementally, and at every iteration  $j$  selects the training point  $(\mathbf{x}_k, y_k)$  which maximizes the entropy difference  $\Delta H_{jk}$  between  $q_{j-1}$  and  $q_j$  for inclusion into the active set. Because  $q$  is Gaussian,  $\Delta H_{jk}$  can be computed by

$$\Delta H_{jk} = -\frac{1}{2} \log |\boldsymbol{\Sigma}_{jk}| + \frac{1}{2} \log |\boldsymbol{\Sigma}_{j-1}|. \quad (3)$$

The details of the implementation can be found in Lawrence *et al.* [11]. The algorithm stops when the active set has reached a desired size. In our implementation, we choose this size to be a fixed fraction  $\gamma$  of the training set  $q$ .

To find the kernel hyper-parameters  $\theta$  of an IVM, two steps are iterated a given number of times: the estimation of  $\mathcal{I}$  given  $\theta$ , and minimising the marginal likelihood  $q(\mathbf{y} | X)$  given  $\mathcal{I}$ . Although there are no convergence guarantees, in practice already a small number of iterations are sufficient to find good kernel hyper-parameters.

Importantly for our work, since inference with the IVM is similar to that with a GPC, the IVM retains the model averaging described in Eq. (2). We argue therefore, that the IVM provides a significant and well-established improvement in processing speed over a GPC while maintaining its introspective properties (see Sec. 5 and 5.4 for details).

## 4 Scalable Active Learning: Drive, Ask, Improve

The power of an active learning framework lies in its ability to select a suitable training set in an application-oriented way. It thus inherently allows the system to adapt naturally to the non-stationarity of the data often encountered in long-term robotics applications. The active learning framework considered here is a supervised learning process by which a human operator provides class labels for machine-selected test data, which are then fed back into classifier training to improve the classification result of the next round. We examine performance over successive *epochs*, which each consist of (re-)training, classification, and user-feedback. The implementation of a scalable active learning framework requires two problems to be addressed: firstly, a subset of test data has to be selected for re-training such that classification performance increases in the next epoch. Secondly, measures have to be taken that guarantee that the training set is bounded in size, since otherwise the algorithm will sooner or later exhaust the resources of a finite-memory, real robotic system. We compare this active learning approach with a more conventional “passive” alternative, that is, training a classifier once without any subsequent human-feedback improvement.

We now outline the specific active learning algorithm employed in this work, before providing details of both our data selection strategy and our approach to forgetting (bounding the training set size).

### 4.1 The Active Learning Algorithm

Algorithm 1 describes our active learning framework which, for reasons given in Sec. 3, uses an IVM as the underlying classifier. It requires five different input parameters: the initial hyper-parameters  $\theta_0$  used for training the IVM, the fraction  $\gamma$  of training points that are used for sparsification, the batch size  $b$ , the normalised entropy (NE) threshold  $\vartheta$  that a test point needs to exceed to be considered for re-training, and the maximum number of questions  $r$  that the algorithm may ask. The last is intended to minimise nuisance to a human operator due to being asked too many questions. The sub-routines in the algorithm are explained as follows.

**Algorithm 1:** Active Learning with an IVM

---

**Data:** training data  $\mathcal{D} = (X, \mathbf{y})$ , stream of test data  $X^*$   
**Input:** initial kernel parameters  $\theta_0$ , batch size  $b$ , active set size fraction  $\gamma$ , minimal retraining score  $\vartheta$ , maximum number of questions  $r$   
**Output:** stream of output labels  $\mathbf{y}^*$   
 $i \leftarrow 0$   
**while**  $X^* \neq \emptyset$  **do**  
     $(\theta_{i+1}, \mathcal{I}_{i+1}) \leftarrow \text{TrainIVM}(X, \mathbf{y}, \gamma, \theta_0)$   
    move next  $b$  test points from  $X^*$  into  $X_i^*$   
     $\mathcal{P} \leftarrow \emptyset$   
    **forall** the  $\mathbf{x}^* \in X_i^*$  **do**  
         $z \leftarrow \text{IVMPrediction}(\mathcal{I}_{i+1}, \theta_{i+1}, \mathbf{x}^*)$   
         $s \leftarrow \text{ComputeRetrainingScore}(z)$   
        **if**  $s > \vartheta$  **then**  $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\mathbf{x}^*, s)\}$   
    sort  $\mathcal{P}$  by decreasing values of  $s$   
     $\mathcal{D}^+ \leftarrow \emptyset$   
    **for**  $j \leftarrow 1$  **to**  $\text{MIN}(r, |\mathcal{P}|)$  **do**  
         $(\mathbf{x}_j^+, s_j) \leftarrow$  element  $j$  of  $\mathcal{P}$   
         $y_j^+ \leftarrow \text{AskLabelFromUser}(\mathbf{x}_j^+)$   
         $\mathcal{D}^+ \leftarrow \mathcal{D}^+ \cup (\mathbf{x}_j^+, y_j^+)$   
     $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}^+$ ,  $i \leftarrow i + 1$

---

`TrainIVM` uses the current training set, the active set fraction  $\gamma$ , and the initial kernel parameters to find optimal kernel parameters  $\theta_{i+1}$  and an active set  $\mathcal{I}_{i+1}$  as described in Sec. 3.1. Throughout this work we employ a squared exponential kernel (which is the same as the Radial Basis Function kernel) with additive white noise:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 e^{-\frac{(\mathbf{x}_i - \mathbf{x}_j)^2}{2l^2}} + \sigma_n^2 \delta_{ij}, \quad (4)$$

where  $\delta_{ij}$  is the Kronecker delta, and  $\theta = \{\sigma_f^2, l, \sigma_n^2\}$  are the signal variance, the length scale, and the noise variance.

`IVMPrediction` returns an estimate of the probability  $z$  that the next test datum  $\mathbf{x}^*$  has a particular class label, as given in Eq. (2). Based on this probability, the normalised entropy measure is then computed. The top ranked  $r$  test data exceeding the retraining threshold  $\vartheta$  are labelled by the user and added to the training set for the next epoch.

#### 4.2 Data Selection Strategy: What Questions to Ask?

The key element of an active learning algorithm is the strategy by which a new test point  $\mathbf{x}^*$  is considered for re-training. In Algorithm 1, this is done in the sub-routine `ComputeRetrainingScore`. An intuitive and well-explored indicator of

which data might be suitable for inclusion is the classification *uncertainty* associated with  $\mathbf{x}^*$ . To characterise the uncertainty of the classification from the given class prediction  $z = p(y_* | X, \mathbf{y}, \mathbf{x}_*)$ , we adopt the measure of *normalised entropy*  $H(z)$ , such that for the binary case,

$$H(z) = -z \cdot \log_2(z) - (1 - z) \cdot \log_2(1 - z), \quad (5)$$

where  $H(z) \in [0, 1]$ , with high values representing high uncertainty.

This, indeed is central to our work. While, in principle, any classification framework which provides a distribution over class labels as output can be used in our active learning framework, intuitively we expect those with more realistic estimates of these probabilities to be more effective for active learning. Thus, we expect more introspective classifiers to perform better in the sense that they will ask more informative questions, leading to a higher learning rate. In Sec. 5, we will show that this is indeed the case when comparing the proposed framework based on an IVM with one based on a more commonly used, probabilistically calibrated SVM.

### 4.3 Forgetting Uninformative Data to Bound Memory Use

The main problem with the active learning framework as we presented it so far is that in theory the training set can grow indefinitely, because there are no guarantees that the algorithm will stop asking new questions. This makes the algorithm less flexible, especially if the input data can not be guaranteed to be within certain locality bounds, for example in a life-long learning application. Therefore, and for run time efficiency, we bound the size of the training set by removing points from it when it exceeds a given target size  $n_t$ . To decide which points to remove, we leverage the information-theoretic instruments that the IVM already provides. After each training round, we keep the entropy differences given in Eq. (3) for all training points and sort them in increasing order. Those training data which correspond to the first  $n_t - n_i$  values, where  $n_i$  is the current training set size, are then removed before training in the next epoch. Intuitively, this method discards the data that were least informative during the last training round. One caveat with this method is that it assumes independence between the training data, which is not generally given. For example, two data may both have small individual  $\Delta H$  values, but when removing both of them the entropy could change significantly. In this work we acknowledge but do not explore this phenomenon. Instead, we note that in our experiments we did not observe a deterioration in classification performance when we applied our method for forgetting.



## 5 Experimental Results

In this section we investigate the performance of our introspective active learning approach in terms of learning rate, data selection strategy, classification performance and tractability. We compare and contrast our approach with one based on the much more commonly used SVM classifier (calibrated to provide probabilistic output). The task we set both learners is to detect traffic lights in a third-party image dataset. Specifically, we use the publicly available Traffic Lights Recognition (TLR) data set [13], which comprises 11,179 colour images taken at 25 Hz from a car driven through central Paris at speeds under 31 mph. It has ground-truth labels for traffic light positions and subtype labels ‘green’, ‘orange’, ‘red’, ‘ambiguous’ (though here we are only concerned with the detection of traffic lights, irrespective of their state). As recommended by the authors of the dataset, we disregard labels of type ‘ambiguous’ and exclude sections where the vehicle was stationary for long periods of time. We use data from the first 5,800 frames for training and the remainder for testing. We compute a template-based feature set inspired by Torralba *et al.* [20] which has a successful track record in the detection of traffic lights [7]. Each training or test window is represented by a feature vector of length 200.

When training the IVM we used an active set fraction  $\gamma$  of 0.2, which means that informative points will be added to the active set until its size is 20% of the training set size. We use a Squared Exponential (SE) with white noise kernel. Training such a classifier takes approximately 1.5 seconds on a single 3.4GHz core.

The SVMs used here are trained using *libsvm* [2], and use the isotropic Radial Basis Function (RBF) kernel, which is equivalent to the SE kernel used by the IVM. They are trained using 10-fold cross-validation on top of a grid-search over the parameters  $C$  (the penalty parameter for the error term) and  $\gamma$  (the inverse of the length scale for the isotropic RBF kernel), both in the space  $2^k$  where  $k = \{-7, -6, \dots, +4\}$ . Training takes approximately 10 minutes.

### 5.1 Does Introspection Improve Active Learning?

One of the central claims of this paper is that the use of an introspective classifier will lead to more informative questions being asked of the human expert. In order to test this claim we perform a cross-over experiment (see Fig. 2) which starts with both an IVM and an SVM are initially trained on the same data, 200 traffic lights (positive) and 200 background patches (negative). Then, 1,000 new data (with a class fraction of 1:1, the same as during training) are shown to both classifiers for testing. Each chooses up to 50 data points (providing their normalised entropies are over a threshold empirically set to be  $\vartheta = 0.97$ ) to add to their own training set for the next round, resulting in *two* new and different training sets: the ‘IVM set’ and the ‘SVM set’. A new IVM and SVM are now trained on *each* of the two new sets and evaluated on a further 1,000 new data points. This process thus gives rise to four classifiers: two IVMs trained on data selected by an IVM and a SVM respectively, and two equivalent SVMs. We compute precision and recall for all four classifiers.

The results after 100 repetitions of this experiment are shown in Fig. 3. As expected, both the IVM and the SVM perform better when trained on the dataset chosen by the initial IVM, suggesting that the questions asked by the IVM tend to be more informative. An unpaired t-test shows this result to be significant to a level of over 95%.

The overall effect of introspection in an active learning setting seems to be an increased learning rate, a claim which we support with the following active learning experiment, performed over 11 epochs. As described in Sec. 4, our active learning algorithm is retrained after having seen a batch of test points, as opposed to running the training algorithm after every new datum encountered. Every epoch consists of a training phase, a classification phase, and a feedback phase. At the very start of epoch 0, the classifiers are trained on 50 positive (traffic light) windows and 500 negative (background) windows extracted at random from the training frames. We choose this class fraction disparity to reflect the fact that in real data sets, negative examples are much more prevalent than positive examples. During each classification phase, the classifiers are then tested on a batch of 1,000 windows extracted from the test frames. The class fraction for these test windows is 1:10, the same as for training. Next, the 50 points with the highest normalised entropy (providing they are over  $\vartheta = 0.97$ ) are added to the training set, ready for retraining at the start of the next epoch. Note that each classifier (IVM and SVM) makes its own choices regarding which points to add for the next epoch.

The results are shown in Fig. 4, where the IVM learner starts off with a worse  $f_1$  measure at epoch 0 but has already exceeded the SVM by epoch 2, and is better (with non-overlapping 95% confidence bounds) in the steady state from then onwards. The gradient of the plot in Fig. 4 is shown in Fig. 5, and shows that the rate of increase of  $f_1$  measure (the learning rate) for the IVM is better than that of the SVM over the first few epochs, and then always at least as good subsequently.

Fig. 4 further serves to justify empirically our choice of normalised entropy as a valid criterion for data selection, by comparing it to randomly selecting new training data. Intuitively, both methods should improve classification by virtue of the fact that they increase the training set size. However, the results indicate that for both the IVM and the SVM, using normalised entropy leads to more rapidly improving classification performance.

## 5.2 Does Forgetting Affect the Performance?

Our work aims to contribute an introspective active learning algorithm that is efficient in terms of computational effort and scalable with respect to its memory requirements. In this section we investigate the efficacy of the mechanism we have put in place to provide this tractability: forgetting. In experiments thus far, new training data were added in each epoch. The IVM active set size is a fixed proportion of the training set size, which has the benefit of increasing classification performance, but is detrimental to processing time. In the context of a life-long-learner, this is not a scalable solution.

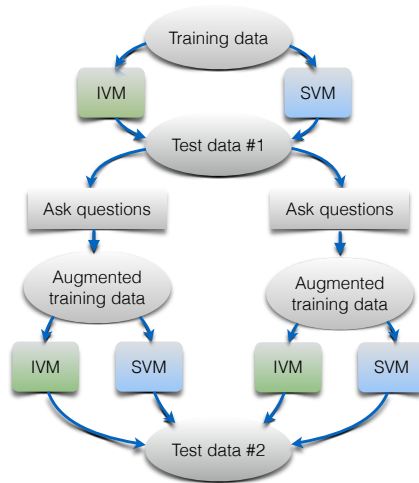


Fig. 2: Here we show the procedure for the cross-over experiment, designed to test whether one classifier chooses points which do not only benefit itself in the next round, but are consistently more useful for the other type of classifier as well. We compare an IVM and an SVM, and choose the test points with highest normalised entropy to be labelled to augment the original training set.

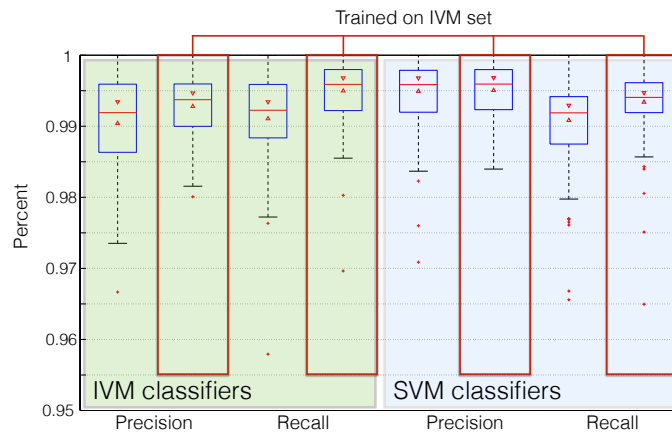


Fig. 3: Data selected by the IVM lead to an improved learning rate in terms of precision and recall for both an IVM and SVM over those selected by the SVM. Results are shown for 100 experimental runs, and increases are significant to the 95% level. See text and Fig. 2 for more details.

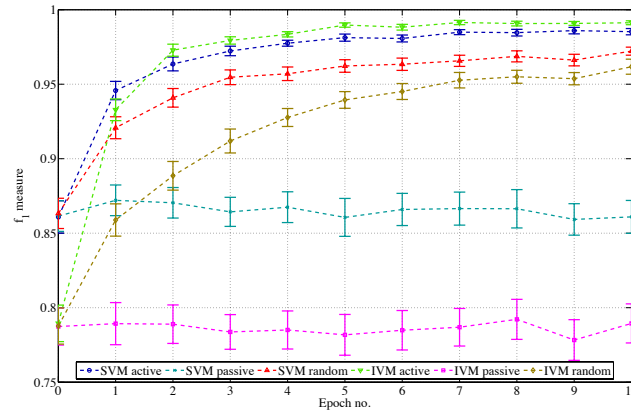


Fig. 4: Classification performance for both IVM and SVM variants as indicated by the  $f_1$ -measure after each epoch. Measurements are averaged over 100 runs. Error bars indicate the 95% confidence region of the mean. The IVM using a normalised entropy-based data selection strategy (IVM-active) consistently outperforms all other active learning variants in terms of learning rate and final classification performance.

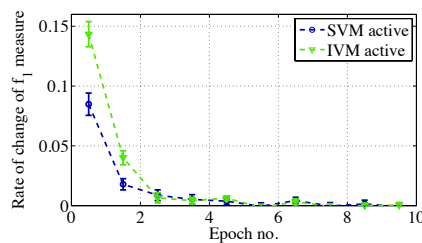


Fig. 5: The gradient of the  $f_1$  measure of the active learners from Fig. 4

We therefore elect to cap the size of the training set at  $n_t = 550$  data, which makes the computational effort constant. This ‘*TVM with forgetting*’ learner can add new data, but only by simultaneously discarding enough data to reduce the training set size to the target size  $n_t$ . Fig. 6 (left) shows the training set size for the normal IVM with unbounded training set, and an IVM with forgetting, capped at 550 data (the initial training amount). Fig. 6 (right) shows the corresponding classification performance as characterised by the  $f_1$  measure. It indicates that in this scenario, the IVM with forgetting mechanism has the same performance as the unbounded IVM. We note that this is likely to be dataset dependent.

### 5.3 What Does the Active Learner Ask?

In Fig. 7 we show the 27 most certain and 27 least certain test cases for an IVM at epochs 0, 3, and 10, and whether they were correctly classified or not. Firstly, it

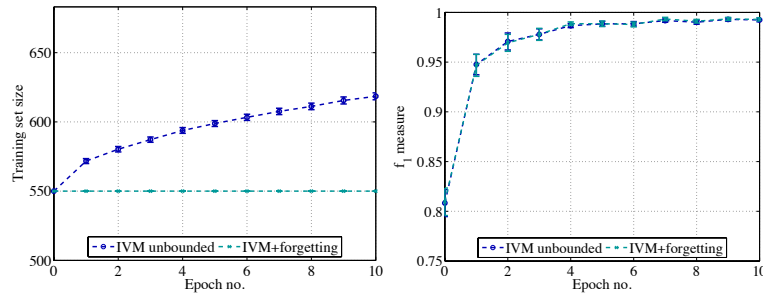


Fig. 6: *Forgetting* results in commensurate classification performance while successfully bounding the active set size of the classifier. Each datum represents the mean (and associated 95% confidence interval) over 100 experimental runs. **Left:** The evolution of the training set size. The IVM+forgetting learner has a target training set size  $n_t = 550$ , the initial training set size. **Right:** Classification performance with and without *forgetting*. For corresponding SVM results, see Fig. 5.

is reassuring to confirm that the certain classifications are always correct. At epoch 0 we see that the confident classifications are all of the background class, almost entirely of fairly uniformly textured surfaces like tarmac, and that the uncertain classifications are all regarding traffic lights. As the learners gather more data, the traffic lights which at epoch 0 were uncertain, are now very confident at epoch 3. At epoch 10, the uncertain group are more balanced in terms of traffic lights and background, and we see that although there is a little more variation in terms of the confident patches, they are very similar to the confidence classifications at epoch 3. This is consistent with the learning algorithm having reached an equilibrium after epoch 3 in Fig. 4.

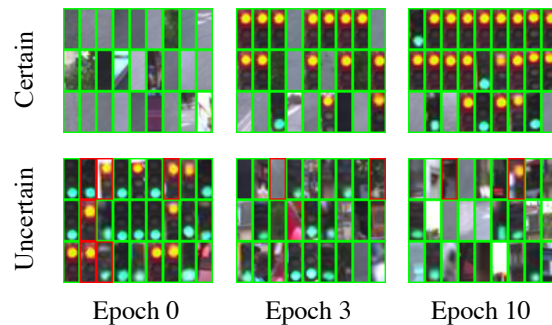


Fig. 7: The 27 most certain and 27 least uncertain test classifications of an IVM at epochs 0, 3, and 10 during the active learning experiment. A green border indicates a correct classification, and a red border indicates an incorrect classification.

### 5.4 The Effects of Sparsity

In [7] we showed that the GPC is more introspective than other more commonly used classification frameworks. In this paper we have argued the necessity of using a sparse formulation for the sake of computational complexity, however, it is necessary to ensure that the IVM is introspective in its own right. The useful characteristic of an introspective classifier is that it tends to be confident when it is making true predictions, and uncertain when it may be making false predictions. In addition, we would like to see whether the introspective quality changes with the active set size; intuitively, a truly introspective classifier will be more confident if it is exposed to more data, and vice versa.

Similarly to the approach in [7] we have plotted the cumulative true and false classifications against uncertainty in Fig. 8 for a single round of training and testing. In the legend, “IVM  $\gamma = 0.4$ ” indicates an IVM with an active set fraction of 0.4, such that the active set contains 40% of the training set. These particular IVMs have been trained on 550 data and tested on 11000, with the ratio 1:10 positive:negative. There are several things to notice from the graph. Firstly, we can see that by looking at the curves for the IVMs with  $\gamma = \{0.2, 0.4, 0.6, 0.8, 1.0\}$ , indeed as we would hope, having a larger active set results in a more confident classifier; however it is interesting to see that there are diminishing returns: very little confidence is gained between an active set fraction of 0.6 and 1.0. Secondly and most importantly, *the IVM is introspective*: the incorrect classifications occur with *high* uncertainty, whereas the majority of the correct classifications occur with *low* uncertainty. Thirdly, we would expect that as the level of sparsity decreases, we approach the behaviour of the GPC, which is indeed what happens; the full GPC is commensurate with the IVMs with  $\gamma = \{0.6, 0.8, 1.0\}$ .

## 6 Conclusion

The contributions of this paper are three-fold: firstly, the notion of introspective classification introduced earlier shows promise in the context of active learning, where a reliable estimate of the classification uncertainty is required. We do this by showing an improvement in both classification performance and learning rate over a non-introspective classifier (Sec. 5.1). Secondly, an efficient version of the Gaussian Process Classifier, namely the Informative Vector Machine is used, which makes the approach particularly useful for robotics applications with large amounts of data. We show visual examples of where it is confused and where it is confident (Sec. 5.3), and use it to create the first offline semantic mapping algorithm via active learning. Finally, we present an information-theoretic solution to the problem of increasing memory requirements by forgetting the least informative data, which maintains a high classification performance in our experiments, but more extensive experimentation is required to confirm the success of this approach for the wider scope of mobile robotics applications.

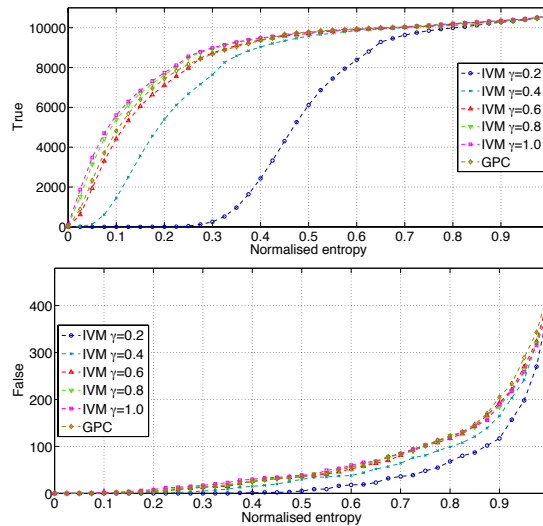


Fig. 8: The introspective capacity of the IVM. We show the number of true (**top**) and false (**bottom**) classifications (positive and negative classes together) which are made with a normalised entropy lower than a chosen value. For instance, if we were to threshold at NE = 0.5, we would have 6000 correct classifications with the IVM  $\gamma = 0.2$  and < 10 incorrect classifications.

## 7 Acknowledgements

This work is funded under the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement Number 269916 (V-CHARGE) and by the UK EPSRC Grant Number EP/J012017/1.

## References

- [1] C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.
- [2] C.C. Chang and C.J. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.
- [3] DARPA. Urban Challenge Route Network Definition File (RNDF) and Mission Data File (MDF) Formats, March 2007.
- [4] C. Dima, M. Hebert, and A. Stentz. Enabling learning from large datasets: Applying active learning to mobile robotics. In *IEEE Intern. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 108–114. IEEE, 2004.
- [5] N. Fairfield and C. Urmson. Traffic light mapping and detection. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

- [6] Y. Freund, H.S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2):133–168, 1997.
- [7] H. Grimmert, R. Paul, R. Triebel, and I. Posner. Knowing when we don't know: Introspective classification for mission-critical decision making. In *Proc. IEEE Intern. Conf. on Robotics and Automation (ICRA)*, 2013.
- [8] A.J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2372–2379, 2009.
- [9] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian processes for object categorization. *International Journal of Computer Vision*, 88(2):169–188, 2010.
- [10] N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, 15:609–616, 2002.
- [11] N. D. Lawrence, J. C. Platt, and M. I. Jordan. Extensions of the informative vector machine. In *Proceedings of the First international conference on Deterministic and Statistical Methods in Machine Learning*, pages 56–87. Springer-Verlag, 2005.
- [12] A. McCallum and K. Nigam. Employing em in pool-based active learning for text classification. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, 1998.
- [13] Robotics Centre of Mines ParisTech. Traffic lights recognition (TLR) data set.
- [14] R. Paul and P. Newman. Self help: Seeking out perplexing images for ever improving navigation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 445–451. IEEE, 2011.
- [15] J. Platt. Probabilistic outputs for support vector machines and comparison to regularize likelihood methods. In A.J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 61–74, 2000.
- [16] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.
- [17] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2010.
- [18] S. Tellex, P. Thaker, R. Deits, T. Kollar, and N. Roy. Toward information theoretic human-robot dialog. In *Robotics: Science and Systems*, 2012.
- [19] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2: 45–66, 2002.
- [20] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(5):854–869, 2007. ISSN 0162-8828.
- [21] R. Triebel, H. Grimmert, R. Paul, and I. Posner. Introspective active learning for scalable semantic mapping. In *Workshop. Robotics Science and Systems (RSS)*, June 2013.
- [22] C. Urmson et al. Autonomous driving in urban environments: Boss and the Urban Challenge. *J. of Field Robotics*, 25(8):425–466, 2008. ISSN 1556-4967.



# Active Online Learning for Interactive Segmentation Using Sparse Gaussian Processes

Rudolph Triebel, Jan Stühmer, Mohamed Souiai, and Daniel Cremers

Computer Vision Group, Dep. of Computer Science TU Munich,  
Boltzmannstrasse 3, 85748 Garching, Germany  
{rudolph.triebel, jan.stuehmer, mohamed.souiai}@in.tum.de  
cremers@tum.de

**Abstract.** We present an active learning framework for image segmentation with user interaction. Our system uses a sparse Gaussian Process classifier (GPC) trained on manually labeled image pixels (user scribbles) and refined in every active learning round. As a special feature, our method uses a very efficient online update rule to compute the class predictions in every round. The final segmentation of the image is computed via convex optimization. Results on a standard benchmark data set show that our algorithm is better than a recent state-of-the-art method. We also show that the queries made by the algorithm are more informative compared to randomly increasing the training data, and that our online version is much faster than the standard offline GPC inference.

## 1 Introduction

Automatic image segmentation is one of the most important problems in computer vision. Its attractiveness stems from its very large range of applications, including medical imaging and robotics. However, in general the image segmentation problem is ill-posed, because a correct segmentation depends strongly on the application. Therefore, we focus on the *interactive* segmentation problem, where the user provides information about the regions to be segmented, e.g. by manually sampling image pixels and assigning them to a predefined region class. These *user scribbles* are used as ground truth information, and the aim is to infer a good segmentation using these scribbles as constraints on the labelling. To do this, many approaches have been presented in the literature with impressive results. However, current methods can reach high classification rates only by requiring comparably many user scribbles, and the number of user scribbles needed usually grows very fast as the segmentation quality approaches 100%.

In this paper, we present a method that asks for user input more intelligently by actively querying pixels to be labeled where the classification was made with high uncertainty. This way, only a few user scribbles are needed to obtain a high quality segmentation. Our method uses an efficient sparse Gaussian Process classifier (GPC) to learn background and foreground models, providing an accurate estimation of the classification uncertainty. We also present a very efficient way to compute the class predictions on every round using an online update rule.

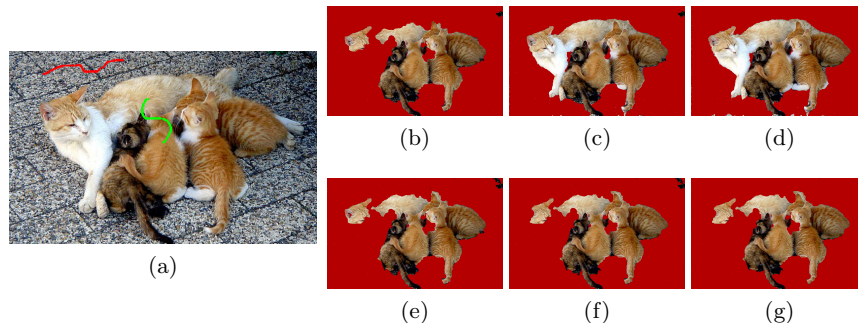


Fig. 1: Comparison between the Parzen window estimator [8] and our sparse GP classifier for foreground classification. From the initial scribble image (a) both approaches learn a model for the foreground. As none of the scribble pixels for the foreground class is white, both approaches fail to classify the white neck of the cat correctly (b, e). However, in the next active learning round, the GP manages to query this part from the user based on its accurate estimation of the predictive uncertainty (c). In contrast, the Parzen window estimator does not query this part, because its uncertainty is low despite its incorrect classification, i.e. it is over-confident (f). After 6 rounds the GP achieves a very good segmentation (d), while the Parzen window estimator still gives a lower-quality segmentation (g).

### 1.1 Related Work

Many previous works use energy minimization for image segmentation, and since the work of Boykov *et al.* [1], intensive research, e.g. [13, 7], has been done on embedding the input image onto a discrete lattice and computing a segmentation using the min-cut framework. Another line of work [16, 8] models segmentation in the continuous domain and is based on the convex relaxation technique of Nikolova *et al.* [9]. Both discrete and continuous approaches impose spatial consistency as a prior on the image labelling. Our work is related to [8] where the data term describing the pixel class probabilities includes spatial information while estimating the colour distribution using a Parzen window estimator. However, we use an Informative Vector Machine (IVM) [5], a sparse version of the Gaussian Process Classifier, and employ active learning, which improves the segmentation result quickly after only a few training rounds (see Fig. 1). In contrast to the sparse GP algorithm of Csató and Oppér [2], the IVM has advantages in the context of active learning, mainly due to the information-theoretic criterion used to select the subset of the training points.

In the field of active learning, Kapoor *et al.* [4] address object categorization using a GP classifier (GPC) where data points possessing large uncertainty (using posterior mean and variance) are queried for labels and used to improve classification. Triebel *et al.* [15] use an IVM to actively learn traffic lights in urban traffic images. Here, we use a similar approach, but with a very efficient online update method for the classification step of the GPC. Vezhnevets *et al.* [17], as well as

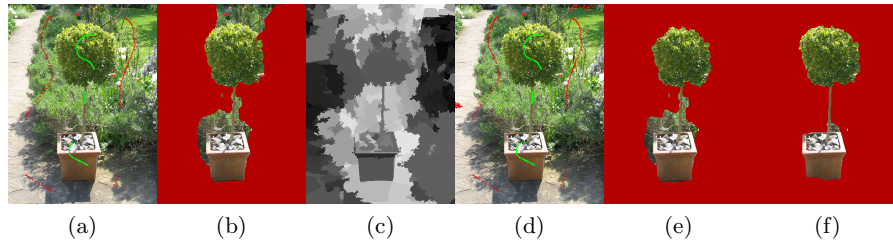


Fig. 2: Example sequence of our proposed active learning framework. The algorithm starts with initial user scribbles as shown in (a). It then learns a sparse GP classifier and segments the image using the GP prediction and a regularization term (b). Then, candidate regions for new, informative user scribbles are computed (c). These are based on the normalized entropy of the GP prediction, i.e. bright regions represent a higher classification uncertainty than darker regions. In this case, a segment at the upper right border is chosen. A label is queried for these pixels (here it is background), and a sub-set of uniformly sampled pixels together with the class labels is added to the training data (d). In the next round, the classification is improved and the result is refined (e). After a few rounds (here 4 in total), the final segmentation is obtained (f).

Wang *et al.* [18] also use active learning for interactive image segmentation, but either with a CRF+NaiveBayes [17] or a Gaussian Mixture Model (GMM) [18] as an underlying classifier. We use a GPC, because it is non-parametric, i.e. it does not assume a functional model for the data, and it was shown to provide very accurate uncertainty estimates, which is crucial in active learning.

## 2 Algorithm Overview

Fig. 2 shows an example sequence of our active learning framework for interactive image segmentation. From a set of initial user scribbles from both foreground and background regions (Fig 2a), our algorithm learns a sparse Gaussian Process Classifier (GPC) and classifies the remaining pixels. Then, a segmentation is obtained using regularization (Fig. 2b), and an uncertainty measure is computed from the predictive variance returned by the GPC. We use a GPC, because its uncertainty estimates are more reliable than those produced by other learning methods such as Support Vector Machines, where reliable refers to a strong correlation between uncertain and incorrectly classified samples (see, e.g., [10]). Then, we perform an over-segmentation of the original image based on superpixels [3] and compute the average classification uncertainty (entropy) for each segment (see Fig. 2c). In the next step, the algorithm selects the segment with the highest uncertainty to query a ground truth label from the user, samples pixels uniformly from the segment, and adds the samples with the obtained labels to the training data set (see Fig. 2d). Note that, due to imperfections in the

segmentation, some segments can contain both foreground and background pixels. In that case, the user can select a “don’t know” option, and the next segment is chosen in the order of decreasing entropies. This however, occurs only rarely when the segmentation is done sufficiently fine-grained. The whole learning and classification process is then repeated for a fixed number of times or until an appropriate stopping criterion is met (Fig. 2e and 2f).

### 3 Gaussian Process Classification

Every round of our active learning algorithm starts by training a Gaussian Process Classifier (GPC) on the current set of user scribbles. If we denote the scribbles as pairs  $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$ , where  $\mathbf{x}_i$  are feature vectors<sup>1</sup> and  $y_i \in \{-1, 1\}$  are binary labels denoting background or foreground, then the task is to compute a *predictive distribution*  $p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*)$ . Here,  $(\mathbf{x}_*, y_*)$  is an unseen pixel/label pair,  $\mathcal{X}$  the set of all training pixels, and  $\mathbf{y}$  the training labels. To compute the predictive distribution, the GPC first estimates a distribution  $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})$  over the *latent variables*  $\mathbf{f} \in \mathbb{R}^N$ , approximating it with a multivariate normal distribution with mean  $\boldsymbol{\mu}$  and covariance matrix  $\Sigma$ , i.e.:  $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) \approx \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \Sigma)$ . This is done using Bayes’ rule:

$$p(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathcal{X})}{\int p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathcal{X})d\mathbf{f}}, \quad (3.1)$$

where  $p(\mathbf{f} \mid \mathcal{X}) = \mathcal{N}(\mathbf{f} \mid \mathbf{0}, K)$  is the prior of the latent variables, and

$$p(\mathbf{y} \mid \mathbf{f}) = \prod_i p(y_i \mid f_i) \quad (3.2)$$

are the likelihoods, which are conditionally independent. These likelihoods are determined using a *sigmoid function*  $\Phi$ , i.e.  $p(y_i \mid f_i) = \Phi(y_i f_i)$ , which has the effect that Eq. (3.1) cannot be computed in closed form. Here, Expectation Propagation (EP) and Assumed Density Filtering (ADF) are commonly used approximations based on a Gaussian  $q(y_i \mid f_i)$  that minimises the Kullback-Leibler (KL) divergence between  $q(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathcal{X})$  and the numerator of Eq. (3.1).

Then, for a given new test data point  $\mathbf{x}_*$ , the GP classifier computes the mean  $\mu_*$  and the variance  $\sigma_*^2$  of the latent variable distribution

$$p(f_* \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \int p(f_* \mid \mathcal{X}, \mathbf{x}_*, \mathbf{f})p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})d\mathbf{f} \quad (3.3)$$

and uses that to compute the predictive distribution

$$p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \int \Phi(f_*)p(f_* \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*)df_* \quad (3.4)$$

<sup>1</sup> These can be either RGB pixel values or a combination of image coordinates and RGB values of the pixels. In our implementation, we use the latter, because it also provides locality information about background and foreground.

If  $\Phi$  is the cumulative Gaussian function this can be done in closed form using

$$p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) = \Phi\left(\frac{\mu_*}{\sqrt{1 + \sigma_*^2}}\right). \quad (3.5)$$

### 3.1 Information-theoretic Sparsification

One problem with the GPC is its huge demand of memory and run time, because it maintains an  $N \times N$  covariance matrix, and the number of training samples  $N$  can be very large. Therefore, we use a sparsification known as the Informative Vector Machine (IVM) [6]. The main idea here is to only use a sub-set of training points denoted the *active set*  $\mathcal{I}_D$ , from which an approximation  $q$  of the posterior is computed. As above,  $q$  is Gaussian, i.e.  $q(\mathbf{f} \mid \mathcal{X}, \mathbf{y}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ . The IVM computes  $\boldsymbol{\mu}$  and  $\boldsymbol{\Sigma}$  incrementally, i.e. in step  $j$  a new  $\boldsymbol{\mu}_j$  and  $\boldsymbol{\Sigma}_j$  are computed:

$$\boldsymbol{\mu}_j = \boldsymbol{\mu}_{j-1} + \boldsymbol{\Sigma}_{j-1} \mathbf{g}_j \quad (3.6)$$

$$\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}_{j-1} - \boldsymbol{\Sigma}_{j-1} (\mathbf{g}_j \mathbf{g}_j^T - 2\Gamma_j) \boldsymbol{\Sigma}_{j-1} \quad (3.7)$$

where

$$\mathbf{g}_j = \frac{\partial \log Z_j}{\partial \boldsymbol{\mu}_{j-1}}, \quad \Gamma_j = \frac{\partial \log Z_j}{\partial \boldsymbol{\Sigma}_{j-1}}, \quad (3.8)$$

and  $Z_j$  is the approximation to the normalizer in Eq. (3.1) using the estimate  $q_j$ . Initially,  $\boldsymbol{\mu}_0 = \mathbf{0}$ , and  $\boldsymbol{\Sigma}_0 = K$ , where  $K$  is the prior GP covariance matrix. Then, at iteration  $j$  the training point  $(\mathbf{x}_k, y_k)$  that maximizes the entropy difference between  $q_{j-1}$  and  $q_j$  is selected into the active set. The algorithm stops when the active set has reached a desired size  $D$ . In our implementation, we choose  $D$  as a fixed fraction of  $N$ .

Due to a circular dependence between  $\mathcal{I}_D$  and the kernel hyper parameters  $\theta$ , the IVM training algorithm loops a given number of times over two steps: estimation of  $\mathcal{I}_D$  from  $\theta$  and minimizing the *marginal likelihood*  $Z_D$  using  $\partial Z_D / \partial \theta$ , thereby keeping  $\mathcal{I}_D$  fixed. Although there are no convergence guarantees, in practice a few iterations are sufficient to find good kernel hyper-parameters.

## 4 Online Update of the IVM

In addition to its sparsity, the IVM differs from the standard GP also by its ability to compute the posterior distribution  $p(\mathbf{f} \mid \mathcal{X}, \mathbf{y})$  *incrementally*. Thus, the algorithm loops over all active points and updates mean vector  $\boldsymbol{\mu}$  and covariance matrix  $\boldsymbol{\Sigma}$  by increasing their lengths in every iteration. In particular, it keeps the lower triangular matrix  $L_d$  of a Cholesky decomposition in memory and updates it using rank-1 Cholesky updates, where  $L_d$  is of size  $d \times d$  and  $d = 1, \dots, D$ . Further details of this procedure are given in Algorithm 1 of Lawrence *et al.*[6]. For our purpose, this incremental scheme is particularly useful, because it avoids the complete re-computation of the GP parameters in every training round and adds only a fixed number of rows and columns to  $L_d$ . This decreases the training time substantially, as we show below. For an efficient *class prediction*, we furthermore propose a novel online update rule, as described next.

#### 4.1 Online Computation of the Class Prediction

To predict a class label  $y_*$  for a new test data point  $\mathbf{x}_*$ , the IVM computes the mean  $\mu_*$  and the covariance  $\sigma_*$  of the approximation to the predictive distribution given in Eq. (3.3), and uses them to obtain the class probability (Eq. (3.5)). With the notation of Rasmussen and Williams [12], this can be expressed as

$$\mu_* = \mathbf{k}_*^T (K + \tilde{\Sigma})^{-1} \tilde{\boldsymbol{\mu}} \quad (4.1)$$

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \tilde{\Sigma})^{-1} \mathbf{k}_*, \quad (4.2)$$

where  $\tilde{\boldsymbol{\mu}}$  and  $\tilde{\Sigma}$  are the *site parameters* of the approximate Gaussian likelihood  $q(\mathbf{y} | \mathbf{f})$ ,  $K$  is the prior covariance matrix, i.e. the kernel function  $k$  applied to all pairs of training points  $\mathbf{x}_1, \dots, \mathbf{x}_N$ , and  $\mathbf{k}_* = (k(\mathbf{x}_*, \mathbf{x}_1), \dots, k(\mathbf{x}_*, \mathbf{x}_N))$ . Note that  $\mathbf{k}_*$ ,  $\tilde{\boldsymbol{\mu}}$ , and  $B := K + \tilde{\Sigma}$  are only computed for  $D$  active points with  $D < N$ .

In general, Eqs. (4.1) and (4.2) have to be computed completely anew for every new test point  $\mathbf{x}_*$ , and it is usually unlikely to observe the same test point again. In active learning, this means that the complexity of making predictions increases quadratically with the training rounds, because in every training round the matrix  $B$  is larger due to the additional active points in the training data. However, for interactive image segmentation, we can use the fact that class predictions are made on the same pixels (i.e. test points) in every round. This means that  $\mathbf{k}_{*,t}$  from round  $t$  can be obtained from  $\mathbf{k}_{*,t-1}$  of the previous round by appending the covariances  $k(\mathbf{x}_*, \mathbf{x}_{D_{t-1}+1}), \dots, k(\mathbf{x}_*, \mathbf{x}_{D_t})$  between  $\mathbf{x}_*$  and the new active points, where  $D_t$  is the total number of active points in round  $t$ . This can be used to compute  $\mu_{*,t}$  and  $\sigma_{*,t}$  incrementally from  $\mu_{*,t-1}$  and  $\sigma_{*,t-1}$ . To do this, we note that  $B_t$  is given by its Cholesky decomposition  $L_t L_t^T$ , and

$$L_t := \begin{pmatrix} L_{t-1} & 0 \\ A & L_+ \end{pmatrix}, \quad (4.3)$$

where  $L_+$  is lower-triangular. To compute  $B_t^{-1}$ , we use

$$B_t^{-1} = \begin{pmatrix} L_{t-1} L_{t-1}^T & L_{t-1} A^T \\ A L_{t-1}^T & A A^T + L_+ L_+^T \end{pmatrix}^{-1}, \quad (4.4)$$

and compute the Schur complement as

$$S = A A^T + L_+ L_+^T - A L_{t-1}^T (L_{t-1} L_{t-1}^T)^{-1} L_{t-1} A^T = L_+ L_+^T.$$

With this, we obtain

$$B_t^{-1} = \begin{pmatrix} C & -L_{t-1}^{-T} A^T S^{-1} \\ -S^{-1} A L_{t-1}^{-1} & S^{-1} \end{pmatrix}, \quad (4.5)$$

where  $C = (L_{t-1} L_{t-1}^T)^{-1} + L_{t-1}^{-T} A^T S^{-1} A L_{t-1}^{-1}$ . We now formulate Eq. (4.2) as:

$$\sigma_* = k(\mathbf{x}_*, \mathbf{x}_*) - (\mathbf{k}_{*,t-1} \ \mathbf{k}_{*,+}) B_t^{-1} \begin{pmatrix} \mathbf{k}_{*,t-1} \\ \mathbf{k}_{*,+} \end{pmatrix}, \quad (4.6)$$

where  $\mathbf{k}_{*,+}$  is the vector of newly added covariances in round  $t$ . Plugging Eq. (4.5) into Eq. (4.6) we obtain for the rightmost term  $r$  of Eq (4.6):

$$r = \hat{\mathbf{k}}_{t-1}^T \hat{\mathbf{k}}_{t-1} + \hat{\mathbf{k}}_{t-1}^T A^T S^{-1} A \hat{\mathbf{k}}_{t-1} - 2 \hat{\mathbf{k}}_{t-1}^T A^T S^{-1} \mathbf{k}_{*,+} + \mathbf{k}_{*,+}^T S^{-1} \mathbf{k}_{*,+} \quad (4.7)$$

where  $\hat{\mathbf{k}}_{t-1} = L_{t-1}^{-1} \mathbf{k}_{*,t-1}$ . It follows that the first term of  $r$  in Eq. (4.7) and the first term in Eq. (4.6) define the predictive variance of the previous round  $\sigma_{*,t-1}$

$$\sigma_{*,t-1} = k(\mathbf{x}_*, \mathbf{x}_*) - \hat{\mathbf{k}}_{t-1}^T \hat{\mathbf{k}}_{t-1}, \quad (4.8)$$

whereas the remaining terms of  $r$  can be subsumed into

$$(L_+^{-1} \mathbf{k}_{*,+} - L_+^{-1} A \hat{\mathbf{k}}_{t-1})^T (L_+^{-1} \mathbf{k}_{*,+} - L_+^{-1} A \hat{\mathbf{k}}_{t-1}), \quad (4.9)$$

which simplifies into

$$(L_+^{-1} \Delta \mathbf{k})^T (L_+^{-1} \Delta \mathbf{k}) \quad (4.10)$$

where  $\Delta \mathbf{k} = \mathbf{k}_{*,+} - A \hat{\mathbf{k}}_{t-1}$ . This results in an efficient way to compute  $\sigma_{*,t}$ : We store  $\hat{\mathbf{k}}_{t-1}$  from the previous round and compute  $\Delta \mathbf{k}$  and  $L_+^{-1} \Delta \mathbf{k}$ . Then we multiply the result with itself (Eq. (4.10)) and subtract it from  $\sigma_{*,t-1}$ . Similarly, we can compute  $\mu_{*,t}$  from  $\mu_{*,t-1}$  of the previous round using the difference vector  $\Delta \boldsymbol{\mu} := \boldsymbol{\mu}_{*,+} - A \hat{\boldsymbol{\mu}}_{t-1}$ , where  $\hat{\boldsymbol{\mu}}_{t-1} = L_{t-1}^{-1} \tilde{\boldsymbol{\mu}}_{t-1}$ . To summarize, we have

$$\mu_{*,t} = \mu_{*,t-1} + (L_+^{-1} \Delta \boldsymbol{\mu})^T (L_+^{-1} \Delta \mathbf{k}) \quad (4.11)$$

$$\sigma_{*,t} = \sigma_{*,t-1} - (L_+^{-1} \Delta \mathbf{k})^T (L_+^{-1} \Delta \mathbf{k}). \quad (4.12)$$

## 4.2 The Kernel Hyper-Parameters

As mentioned before, finding optimal hyper parameters for the kernel function involves several iterations over active set determination and gradient-descent on the marginal likelihood. However, doing this in every training round has several disadvantages: first, it requires a large computational effort, and second it makes the formulation of the online computation developed in the previous section invalid. The reason for the latter is that the online formulation relies on the fact that the active set does not change across the learning rounds, because otherwise  $\mathbf{k}_*$  would have to be recomputed completely in every round. Fortunately, it turns out that the kernel hyper parameters do not change significantly across the training rounds, and, even when they do change, they only have a minor impact on the classification results of the GPC. This is another strength of the GPC framework, because essentially it represents a non-parametric model. In our implementation, we obtain the kernel hyper-parameters using cross-validation on a hold-out set. Compared to the usual gradient-descent based maximization of the log marginal, this has the advantage that the kernel parameters are optimized *across* a number of images, and not for each individual image. Especially, as we use locality and color features in combination with an Automatic Relevance Determination (ARD) kernel, the obtained length scales represent a general weighting between position and color. This turned out to achieve much better results than a per-image training of the ARD kernel parameters.

8 Triebel et al.

## 5 Segmentation

The class predictions from the IVM are only local estimates, and they disregard global properties of the image  $I$ . Therefore, we formulate the segmentation problem on the image domain  $\Omega \subset \mathbb{R}^2$  as finding a *foreground region*  $\hat{\Omega}_F$  so that

$$\begin{aligned} \hat{\Omega}_F = \arg \min_{\Omega_F} & \lambda \int_{\Omega_F} -\log p(y_* = 1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) \, d\mathbf{x} \\ & + \lambda \int_{\Omega \setminus \Omega_F} -\log p(y_* = -1 \mid \mathcal{X}, \mathbf{y}, \mathbf{x}_*) \, d\mathbf{x} + \text{Per}_\alpha(\Omega_F), \end{aligned} \quad (5.1)$$

where  $\text{Per}_\alpha$  is the perimeter of  $\Omega_F$ , weighted by a local metric  $\alpha(\mathbf{x}) = e^{-\gamma|\nabla I|}$  that depends on the image gradient, and  $\lambda$  is the weight of the dataterm. This functional favours spatial regularity by penalizing the boundary length of the foreground region. First, we define an indicator function  $u(\mathbf{x})$  that is 1 for  $\mathbf{x} \in \Omega_F$  and 0 otherwise. Then, the segmentation problem can be written in a variational formulation:

$$\min_{u \in [0,1]} \int_{\Omega} \varrho(\mathbf{x})u(\mathbf{x}) \, d\mathbf{x} + \frac{1}{\lambda} \int_{\Omega} \alpha(\mathbf{x})|\nabla u(\mathbf{x})| \, d\mathbf{x}, \quad (5.2)$$

where the first term encodes the cost of a pixel to belong to the foreground and  $\varrho(\mathbf{x}) = \log p(u(\mathbf{x}) = 0) - \log p(u(\mathbf{x}) = 1)$ . The second term of Eq. (5.2) is the total variation (TV) of the indicator function  $u$  which penalizes the perimeter of the foreground region. Since the TV is not differentiable everywhere, we rewrite Eq. (5.2) as a saddle point problem:

$$\min_{u \in [0,1]} \max_{|v| \leq \alpha(\mathbf{x})} \int_{\Omega} \varrho(\mathbf{x})u(\mathbf{x}) + \int_{\Omega} u(\mathbf{x}) \operatorname{div} v(\mathbf{x}). \quad (5.3)$$

This can be efficiently minimized using a first-order primal-dual method [11].

## 6 Experimental Results

We evaluate our active learning approach on the benchmark data set from the University of Graz [14]. It consists of images with ground truth segmentations and user scribbles. As our method applies for foreground and background segmentation we chose a subset of 44 images from the dataset which contain only two object classes. As performance measure for this benchmark we use the  $f_1$  measure, which is defined as the harmonic mean of precision and recall.

### 6.1 Benefits of the GP classifier

We compare our approach with the method of Nieuwenhuis and Cremers [8]. There, the data term is computed using a Parzen window (PW) estimator, and



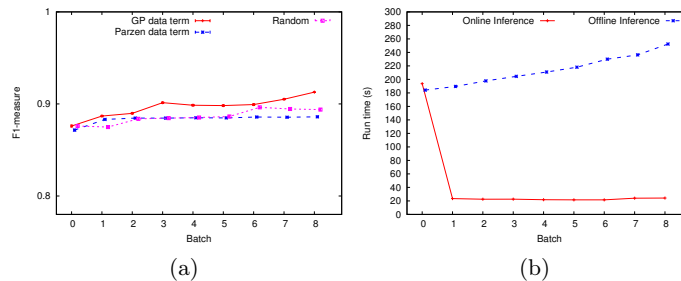


Fig. 3: (a) Average f-measure over 8 active learning rounds. The GPC steadily improves the segmentation, because its label queries are more informative for classification. In contrast, the Parzen window only improves slightly and then remains at a lower performance level. We also show GPC results where new user scribbles are chosen randomly and not based on the entropy. This also improves the segmentation, as it increases the training data, but it is worse than the entropy-based method. (b) Run time of online and offline inference, averaged over all images. Note that in batch 0, the online and the offline method take the same time, because they both build up the initial covariance matrix. However, in later steps the online computation time drops down significantly.

the training data consists of color information and positions of user scribbles. We use the same idea, but employ a GPC instead of the PW. Our benefit is the ability to detect misclassifications using the predictive uncertainty, which is more strongly correlated to incorrect classifications than for the PW. As a result, in active learning the GPC generates more informed questions (see Fig. 1). For a quantitative evaluation, we ran active learning with the GPC and the PW on the Graz data set (Fig. 3a). Both approaches perform equally well in the first rounds, but then the GPC (red curve) outperforms the PW (blue curve), because it asks more informed label queries, while the PW tends to be overconfident. We also show the results for randomly selected scribbles (magenta curve) instead of those with the highest uncertainty. We see that random sampling also improves the classification, as it provides more training data in every round, but the improvement is smaller compared to selecting the most uncertain segments. This is because the GP requests the more informative user scribbles.

Some results from the Graz data set are shown in Fig. 4. The left column shows the images with the initial user scribbles. Columns two and three show the uncertainties of the GPC (brighter is more uncertain) and the segmentation after the first learning round. The general segmentation is good, but small misclassifications occur. However, these often correspond to locations of high uncertainty, e.g. the lower right corner of the helicopter image or the third peg on the wardrobe: here the classification is incorrect, but the uncertainty is also high. This enables the classifier to correct the error in subsequent training rounds.

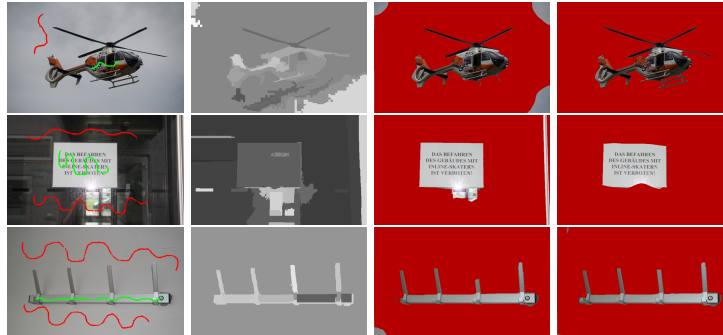


Fig. 4: Examples from the Graz benchmark. **First column:** original images with initial user scribbles. **Second column:** classification uncertainties after the first learning round. **Third column:** resulting segmentation after the first round. Note how the algorithm misclassifies some small areas, but the classification in those same areas is often very uncertain (see, e.g., the third peg on the wardrobe). Thus, the errors can be corrected by querying more useful, i.e. informative user scribbles. **Last column:** final results, obtained after a few further active learning rounds (between 1 and 5). Here, a high-quality segmentation is obtained.

## 6.2 Advantage of the Online Inference Algorithm

As mentioned in Sec. 4.1, we use a very efficient online class prediction step. Note that this is different from an online *training* step: while the latter is inherently provided by the IVM approach, the former is a novel contribution. In Fig. 3b, we show its benefit over the standard offline technique in every active learning round. Observe that for all but the first learning round the average run time drops from the order of minutes to the order of seconds. Also note that the increase in run time over the learning rounds is super-linear in the offline case, where for the online method it is roughly linear. In the first round, the online and the offline method perform the same steps, because every pixel is compared to all training points. Currently, we compute this in parallel on 8 CPU threads, but we expect a substantial speed-up when using a GPU implementation.

## 7 Conclusions

We present an efficient active learning approach and show its application to interactive segmentation. Our method learns models for background and foreground adaptively by informed questions based on the classification uncertainty and uses a regularizer that favors regions with smooth contours. To make the classification process efficient, we use an online update method that incrementally estimates the class posteriors. This reduces computation time substantially, without reducing the high segmentation performance of the active learning method.

*Acknowledgment* This work was partly funded by the EU project SPENCER (ICT-2011-600877).

## References

1. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. *Trans. on Patt. Analysis and Machine Intell.* 23(11), 1222–1239 (2001)
2. Csató, L., Opper, M.: Sparse on-line gaussian processes. *Neural Computation* 14(3), 641 – 668 (2002)
3. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. *Intern. Journal of Computer Vision* 59(2), 167–181 (2004)
4. Kapoor, A., Grauman, K., Urtasun, R., Darrell, T.: Gaussian processes for object categorization. *Intern. Journal of Computer Vision* 88(2), 169–188 (2010)
5. Lawrence, N., Seeger, M., Herbrich, R.: Fast sparse gaussian process methods: The informative vector machine. *Adv. in neural inf. proc. systems* 15, 609–616 (2002)
6. Lawrence, N.D., Platt, J.C., Jordan, M.I.: Extensions of the informative vector machine. In: *Proc. of the First Intern. Conf. on Deterministic and Statistical Methods in Machine Learning*. pp. 56–87. Springer-Verlag (2004)
7. Lombaert, H., Sun, Y., Grady, L., Xu, C.: A multilevel banded graph cuts method for fast image segmentation. In: *Intern. Conf. on Computer Vision (ICCV)*. pp. 259–265 (2005)
8. Nieuwenhuis, C., Cremers, D.: Spatially varying color distributions for interactive multi-label segmentation. *Trans. on Patt. Analysis and Machine Intell.* 35(5), 1234–1247 (2013)
9. Nikolova, M., Esedoglu, S., Chan, T.F.: Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics* 66(5), 1632–1648 (2006)
10. Paul, R., Triebel, R., Rus, D., Newman, P.: Semantic categorization of outdoor scenes with uncertainty estimates using multi-class Gaussian process classification. In: *Proc. of the Intern. Conf. on Intelligent Robots and Systems (IROS)* (2012)
11. Pock, T., Cremers, D., Bischof, H., Chambolle, A.: An algorithm for minimizing the piecewise smooth mumford-shah functional. In: *Intern. Conf. on Computer Vision (ICCV)* (2009)
12. Rasmussen, C.E., Williams, C.K.I.: *Gaussian processes for machine learning* (2006)
13. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. on Graphics (TOG)* 23(3), 309–314 (2004)
14. Santner, J., Pock, T., Bischof, H.: Interactive multi-label segmentation. In: *Asian Conf. on Computer Vision*. pp. 397–410. Springer (2011)
15. Triebel, R., Grimmett, H., Paul, R., Posner, I.: Driven learning for driving: How introspection improves semantic mapping. In: *Proc of Intern. Symposium on Robotics Research (ISRR)* (2013)
16. Unger, M., Pock, T., Trobin, W., Cremers, D., Bischof, H.: TVSeg - interactive total variation based image segmentation. In: *British Machine Vision Conf.* (2008)
17. Vezhnevets, A., Buhmann, E.J., Ferrari, V.: Active learning for semantic segmentation with expected change. In: *Conf. on Comp. Vision and Patt. Recog.* (2012)
18. Wang, D., Yan, C., Shan, S., Chen, X.: Active learning for interactive segmentation with expected confidence change. In: *Asian Conf. on Computer Vision* (2012)

# Environment-adaptive Learning: How Clustering Helps to Obtain Good Training Data

Shoubhik Debnath<sup>1,2</sup>, Shiv Sankar Baishya<sup>1,2</sup>, Rudolph Triebel<sup>1</sup>,  
Varun Dutt<sup>2</sup>, and Daniel Cremers<sup>1</sup>

<sup>1</sup> Computer Vision Group, Technical University Munich, Germany

<sup>2</sup> Indian Institute of Technology Mandi, India

{debnath, baishya, triebel, cremers}@in.tum.de  
varun@iitmandi.ac.in

**Abstract.** In this paper, we propose a method to combine unsupervised and semi-supervised learning (SSL) into a system that is able to adaptively learn objects in a given environment with very little user interaction. The main idea of our approach is that clustering methods can help to reduce the number of required label queries from user interaction, and at the same time provide the potential to select useful data to learn from. In contrast to standard methods, we train our classifier only on data from the actual environment and only if the clustering gives enough evidence that the data is relevant. We apply our method to the problem of object detection in indoor environments, for which we use a region-of-interest detector before learning. In experiments we show that our adaptive SSL method can outperform the standard non-adaptive supervised approach on an indoor office data set.

**Keywords:** Semi-supervised learning, active learning

## 1 Introduction

Current machine perception systems often rely on their capabilities to automatically learn a mapping from the set of potential observations to a set of semantic annotations, for example class labels from a natural language. The biggest challenges for the employed learning algorithms are the large amount of labelled data they usually require, and their potential to adapt to new, unseen environments and situations. In many applications, and particularly in mobile robotics, this adaptability is an important requirement, because it is impossible to anticipate all situations that the robot might encounter before deployment. Therefore, we investigate learning mechanisms that are capable of adapting to new observations by updating their internal representation as new information arrives. This implies that the learning step is performed during operation of the system and not beforehand, and that the data used for training is acquired online. However, the main question is: what are good data to train on? A good answer to this

question directly leads to a shorter training time and in a reduced amount of required human data annotations.

In this paper, we address this question using a simple, but effective idea: Before asking the human supervisor for a semantic label, we group the observed data into clusters using unsupervised learning. Then, our algorithm queries one common label for each cluster from the supervisor and uses the so obtained training data in a semi-supervised learning step. This approach has two major advantages: first, it further reduces the amount of human intervention significantly by asking labels for multiple instances at the same time. And second, it gives us the potential to pre-select interesting data to train on, for example by asking labels only for clusters that are significantly represented. We apply our method to the problem of object detection in indoor office environments, and we show in experiments that this adaptive way of learning can outperform the standard approach, where a purely supervised classifier is learned before observing the actual test data.

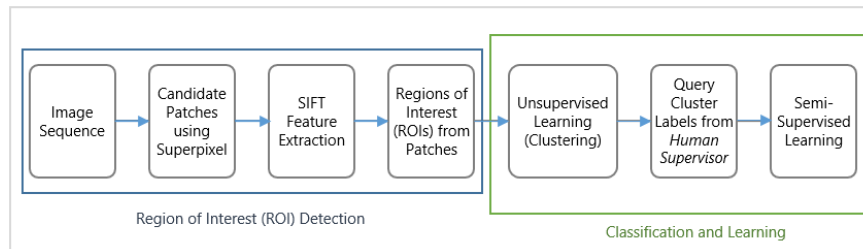
## 2 Related Work

Our work is mostly related to the area of semi-supervised learning (SSL) and transductive learning methods, which have become very popular in the last decade. A good overview of this field is given by Zhu [1,2], who also proposed a graph-based SSL method named Label Propagation. Other methods include the sparse Gaussian Process classifier with null category noise model [3], semi-supervised boosting [4] and the transductive Support Vector Machine (tSVM) [5]. In our work, we also use unsupervised learning as in [6] and combine it with a tSVM to reduce the required interaction with the human supervisor even further. Example applications of SSL in computer vision include image classification from labelled and unlabelled, but tagged images [7], object recognition [8], and video segmentation [9].

Furthermore, our work is also related to the area of active learning, because it involves a user interaction step, for which queries for class labels are actively generated. A good overview on the active learning literature is given by Settles [10]. One interesting example of active learning is the work of Kapoor *et al.* [11] on object categorization using a GP classifier (GPC), where data points possessing large uncertainty (using posterior mean and variance) are queried for labels and used to improve the classification. Triebel *et al.* [12] use active learning for semantic mapping where a sparse GP classifier actively learns to distinguish traffic lights from background. In contrast to classical active learning methods, our approach chooses the data to be asked for labelling based on a relevance criterion rather than, e.g. based on the entropy of the underlying classifier.

## 3 Combined Unsupervised and Semi-Supervised Learning

Fig. 1 gives an overview of our proposed semi-supervised learning method. We start with a sequence of input images and determine first an appropriate set of

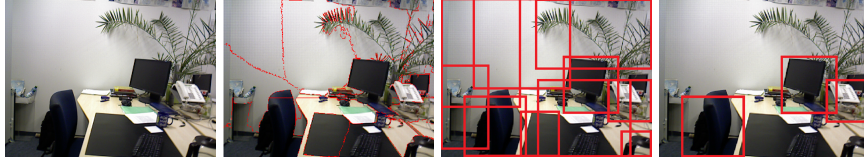


**Fig. 1.** Flow chart of our proposed system. From a sequence of images, regions of interest are detected using super pixel segmentation and by comparing the segments based on SIFT features. Then the resulting patches are clustered. From each cluster, a subset of patches is used to query object labels from a human supervisor. The resulting hand-labelled data together with some unlabelled samples is then used to train a semi-supervised classifier.

rectangular regions of interest named *patches*. From these patches, we extract SIFT features (“Scale-invariant feature transform”, [13]) and use them to define a similarity measure between patches. Based on these similarities, we cluster the patches using spectral clustering. Then, we select a subset of appropriate patches from each cluster and query object labels from a human supervisor as described below. The resulting labelled patches, together with the remaining unlabelled ones are then passed into a multi-class transductive SVM, which then returns predicted labels for the unlabelled patches. In the following sections we describe each step in more detail and give motivations for our algorithm design.

### 3.1 Region of Interest Detection

Object detection for a given image of a scene is much harder than pure object recognition, because it is not even known to the algorithm *if* the object to be recognized exists in the scene and *where* it is. The common approach to this problem is to determine small sub-windows within the image which potentially contain the object(s) to be classified. In the simplest case, these so-called *regions of interest* (ROI) are obtained using a sliding-window approach. However, to reduce the number of potential ROIs, we use a different method: Given an image sequence, we first compute a superpixel segmentation for each image based on the SLIC algorithm [14]. Then, we compute the bounding box for each segment in every image. For each such resulting candidate patch, we extract SIFT features [13] and compare the patches across the image sequence using a similarity measure  $s$ . The motivation for the choice of SIFT descriptor is their high expressive power and their ability to find good matches even under changes of illumination, orientation and scale. In our application, object instances do not vary much in color or texture, which is an ideal condition for the SIFT descriptor. Of course, in a more general setting, where the appearance between the objects of a class



**Fig. 2.** Example result of our ROI detector. From left to right: 1. Original image 2. SLIC superpixels with boundaries in red, 3. Bounding boxes of the super pixels, 4. Detected ROIs after threshold.

may vary more, other descriptors, for example based on the geometry may be more appropriate.

To compute the similarity measure  $s$ , we first define a distance function  $d$  between two patches  $A$  and  $B$  as:

$$d(A, B) = \frac{1}{n} \sum_{i=1}^n \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i\|^2, \quad (1)$$

where  $n$  is the number of matches found by the SIFT algorithm and  $i$  iterates over all these matches. The vectors  $\tilde{\mathbf{x}}_i$  and  $\tilde{\mathbf{y}}_i$  denote the 128-dimensional descriptor values computed at the key points found by the SIFT method in patches  $A$  and  $B$ , respectively. From this distance measure, we define the similarity  $s$  between two patches as:

$$s(A, B) = 1 - \frac{d(A, B)}{\max_{A', B'} d(A', B')}, \quad (2)$$

thus,  $s$  gives values between 0 and 1, where 1 corresponds to maximal similarity. To find patches that contain potentially interesting objects, we compute a similarity score  $p$  for patch  $A$  as follows:

$$p(A) = \sum_{B \neq A} s(A, B), \quad (3)$$

i.e. the score is defined by the sum of similarities to all other patches. The intuition here is that patches that are very similar to many others more likely contain objects of interest, because they give evidence that there are many instances of the same object class. Note that our formulation implicitly deals with the problem that background patches containing walls, the floor, etc., despite occurring very often will not give a high score, because their appearance is usually much more uniform, which means that much less SIFT key points are detected on them.

Using these score values, an ROI is then detected as the patches  $A$  for which  $p(A)$  exceeds the average score over an entire image. This simple statistical method finds patches that stick out in terms of their similarities and has the advantage that it does not require to introduce a threshold parameter. In our experiments, this gave good results (see Fig. 2 for an example sequence of our detector), but of course other methods could be used here.

### 3.2 Clustering of Patches

The main contribution of our work is the idea of using unsupervised learning *before* employing a semi-supervised method for classification. The motivation of this approach is two-fold: first, the number of required user interactions, i.e. label queries, is further reduced compared to standard semi-supervised learning, because we query only one common label for an entire group (cluster) of data instances. And second, the clustering step gives us the opportunity to pre-select interesting data to train on, because typically some clusters can be easily identified as more relevant for the learning task based on simple characteristics such as cluster size or similarities of elements within a cluster. The intuition here is that only those data instances should be learned by the classifier, for which there is enough evidence that they correspond to a meaningful object class. For example, in an office environment, usually there are many instances of classes like telephone, chair or monitor, and the mere fact that there are many very similar instances makes them highly relevant, for example for a mobile robotic system operating in the environment. In contrast, in a home environment, there might be other types of relevant objects, and our approach particularly aims at finding such relevant classes adaptively.

To perform the clustering step, we use the same SIFT descriptors computed earlier for each patch and rely on the same similarity measure  $s$  to cluster the patches. We ran experiments with two different standard clustering methods:  $k$ -means clustering and spectral clustering. Both methods have been used very successfully in many different kinds of applications, and we found that the difference in performance is not very substantial. We evaluated both methods on our data using the V-measure [15], which is defined as the harmonic mean of homogeneity and completeness of the clustering algorithm. In these experiments, the spectral clustering was slightly better, and it has the further advantage that it does not necessarily require the number of clusters specified as a parameter. The reason is that it is based on the eigen decomposition of the graph Laplacian of the data, and that a method called the *eigen gap heuristic* can be used to determine a good value for the number of clusters. For more details on spectral clustering, we refer to the work of Luxburg [16].

### 3.3 Querying Object Labels

The next step in our proposed method is to receive class label information from a human supervisor for the patches that have been clustered beforehand. To perform this label query, some important considerations need to be taken into account: On one side, the algorithm should ask the user as few times as possible to give a label input, because this is one of the main motivations of this work. Thus, we want to ask only once for each cluster. On the other side, we need to make sure that the data we provide as training samples to the semi-supervised learning method is as *pure* as possible, i.e. ideally there should be no instances of different objects labelled by the human with the same label. Unfortunately, no clustering algorithm can guarantee complete purity, neglecting of course the



trivial clustering that assigns every data point to its own cluster. Therefore, we propose to use a quality measure  $q$  for all patches within a cluster, which is based on the similarities  $s$  computed earlier. Concretely, for every patch  $A$  of a given cluster  $\mathcal{C}$ , we compute  $q$  as the sum of similarities *within the cluster*:

$$q(A) = \sum_{B \in \mathcal{C}} s(A, B). \quad (4)$$

Note that this is different from the scores computed in Eq. (3), because here, our goal is to find the best cluster representatives. After computing the  $q$ -values, we sort all elements within a cluster in descending order of  $q$  and ask one common label from the user for the first  $m$  such elements of each cluster. This policy gives a good trade-off between the two opposing objectives of generating few label queries and providing pure training data. Of course, this method does not guarantee that there are no instances of different object classes that receive the same label from the supervisor. However, from our experience, the number of cases where queried data points are inconsistent can be reduced substantially using this method.

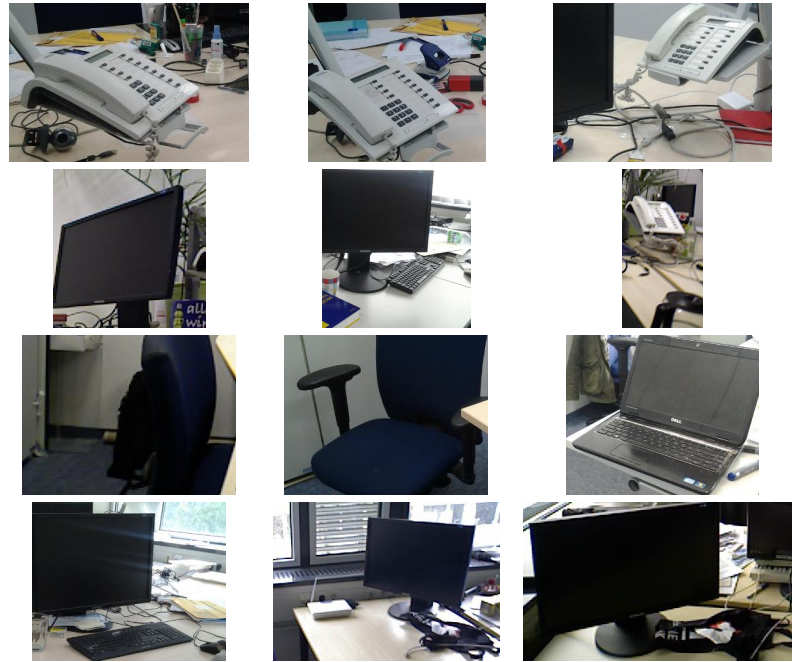
To illustrate this step, Fig. 3 shows an example result of the clustering step, where each row corresponds to a different cluster and only the first 3 elements according to the quality measure  $q$  are shown. As we can see, in two out of four cases the first three cluster elements only contain objects of the same class, and in the other two cases the mistakes made by the algorithm are completely comprehensible. We also note that the clustering result yields more clusters than there are actual classes, i.e. we have an over-clustering. This is only a problem in the sense that it requires the user to give more class labels than actually needed, but this effect was only minor in our experiments.

### 3.4 Training a Classifier

As a final step in our approach, we use the labelled data obtained from the previous step to learn a classifier for the objects discovered in the environment. Here, we considered three different strategies. First, we investigated the use of a standard supervised learning method using a linear Support Vector Machine (SVM). Then, we evaluated two semi-supervised learning techniques, where the first was a simple nearest neighbour rule, i.e. each unlabelled sample was assigned the label of the closest labelled sample according to our similarity measure. And finally, we used a transductive SVM [5] with an RBF kernel. Thus, in addition to the labelled training set  $\mathcal{D}$  of size  $l$ , the algorithm is also given an unlabelled set  $\mathcal{D}^* = \{\mathbf{x}_i^* \in \mathbb{R}^p\}_{i=1}^k$  of test examples to be classified. Formally, a transductive SVM is defined by the following primal optimization problem:

Find  $(\mathbf{y}_1^*, \mathbf{y}_2^*, \dots, \mathbf{y}_n^*, \mathbf{w}, b)$  so that

$$\begin{aligned} & \min \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i[\mathbf{w} \cdot \mathbf{x}_i - b] \geq 1, \quad y_j^*[\mathbf{w} \cdot \mathbf{x}_j^* - b] \geq 1, \\ & y_j^* \in \{-1, 1\} \quad \forall i = 1, \dots, l, \forall j = 1, \dots, k \end{aligned} \quad (5)$$



**Fig. 3.** Examples of clusters obtained from the clustering algorithm (every row corresponds to a different cluster). For each cluster, we show the first three elements according to the quality measure defined in (4).

where  $(\mathbf{x}_i, \mathbf{y}_i)$  are the training examples,  $\mathbf{y}_i^*$  are the predicted labels for the unlabelled test example and  $\mathbf{w}$  is the weight vector. This means, that the transductive SVM learns from both the labelled and the unlabelled examples, and it returns label predictions for the unlabelled ones. In that sense, the training and the inference step are contained within the same common procedure.

From these three methods the worst in our experiments was the standard supervised SVM, and we did not consider this further. The highest classification performance was obtained with the transductive SVM, and we give more details in the experimental section. As feature vectors for training, we compute for every patch the Hierarchical Matching Pursuit (HMP) descriptor introduced by Bo *et al.* [17]. The HMP features are calculated in a multi-layer process where each layer is computed on a different scale, containing the same three steps: Matching Pursuit, Pyramid Max Pooling and Contrast Normalization. The key element in this process is the Matching Pursuit step, which is based on a sparse coding algorithm known as K-SVD. Given a set of  $h$ -dimensional observations  $Y = [y^1, \dots, y^n] \in R^{h \times n}$  (image patches in our case), K-SVD learns

8 Debnath et al.

a dictionary  $D = [d^1, \dots, d^m] \in R^{h \times m}$ , and an associate sparse code matrix  $X = [x^1, \dots, x^n] \in R^{m \times n}$  by minimizing the following reconstruction error,

$$\min_{x_i} \|y_i - Dx_i\|^2 \text{ s.t. } \|x_i\|_0 \leq K, \quad (6)$$

where  $x_i$  are the columns of  $X$ , the zero-norm  $\|x_i\|_0$  counts the non-zero entries in the sparse code  $x_i$ , and  $K$  is the sparsity level, which bounds the number of the non-zero entries. The Matching Pursuit step finds an approximate solution to the optimization problem mentioned above using a greedy approach. Pyramid Max Pooling is a non-linear operator that generates higher level representations from sparse codes of local patches which are spatially close. And Contrast Normalization turns out to be essential for good recognition performance, since the magnitude of sparse codes varies over a wide range due to local variations in illumination and foreground-background contrast. Bo *et al.* [17] used a linear SVM in combination with HMP features and reported very good classification results. We verified these results using data from the Caltech 101 benchmark, and we show them in the results section. From this, we conclude that HMP features exhibit a high amount of expressiveness, because they give very good classification results for a comparably simple classifier such as the linear SVM.

In practice, the use of HMP features consists of two phases: one where the dictionaries are learned from some given training data, and one where feature vectors are computed for new test data based on the sparse codes with respect to the learned dictionaries. While the first phase can require huge computation time, as it usually uses a large training data set, the online phase is comparably fast, as it only requires the computation of a sparse representation for a given dictionary. We note however, that the dictionary learning step is completely unsupervised, as it does not require any human-labelled data.

## 4 Experiments and Results

To measure the performance of our approach, we performed several experiments. First, we evaluated our method to detect regions of interest. Then, we evaluated two different semi-supervised learning methods on a benchmark and on our own data. And finally, we verified experimentally the benefits of using our adaptive, semi-supervised learning method over a standard non-adaptive supervised strategy. More details about all experiments are given in the following.

### 4.1 Evaluating the ROI Detector

As mentioned above, our ROI detector finds patches that occur often with high similarity across images. Therefore, to assess this method quantitatively, we first created ground truth data for the objects that occurred most frequently in our data. Concretely, we labelled those ROIs as correct detections, which contained chairs, monitors or telephones. Results on 7 different images in terms of precision and recall are given in table 1. We see that our detector tends to find more

	Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7
<b>Actual ROIs</b>	2	3	2	1	1	2	2
<b>Predicted ROIs</b>	4	4	2	3	3	4	4
<b>Recall</b>	1	0.67	1	1	1	1	1
<b>Precision</b>	0.5	0.5	1	0.33	0.33	0.5	0.5

**Table 1.** Evaluation of the ROI detector on 7 input images. While the precision is comparably low, recall is good, which is the main purpose of this step.

ROIs than there actually are, and the recall is much better than the precision. However, for ROI detection we are actually more interested in recall than in precision, because missing a candidate for classification is worse than reporting a background patch as a ROI, as the latter can be handled by the classifier.

For a qualitative evaluation, we show an example result of the ROI detector in Fig. 4. As we can see here, the detector found the two regions of actual interest, i.e. the chair and the monitor, and it only returned one false positive.

#### 4.2 Comparison of Adaptive Semi-Supervised Learning and Standard Supervised Learning

To measure the performance of our adaptive semi-supervised learning method, we ran experiments on a subset of the standard benchmark data set Caltech 101, and on our own data. The subset consisted of 10 classes (see Fig. 5), for the Caltech 101 and 3 classes for our data. For both experiments, we used dictionaries for the HMP features that were learned from 10 images per class from the benchmark set. For the Caltech 101 we did not employ the ROI detector, because these images already contain one major object and not much background. Thus, we only clustered the data, computed HMP features for each image and trained a semi-supervised learner on a mixture of labeled and unlabeled images, where the labels were obtained from querying the best 3 representatives of each cluster. The results for the  $k$ -nn method and the transductive SVM with RBF kernel are given in the left column of Table 2. As we can see, the transductive SVM performs much better than the  $k$ -nn approach, and the final accuracy is



**Fig. 4.** Example result of our ROI detector. The ground-truth ROIs are shown on the left and the predicted ROIs on the right.



**Fig. 5.** Examples from each of the 10 classes in the Caltech 101 data set which were used for the experiments.

comparably high, given that only very few data samples used for training were actually labeled.

The same conclusion we can draw for our indoor office data set (see right column of Table 2). Here, we used 25 ROIs for evaluation, consisting of 6 chairs, 13 monitors and 6 telephones. Again, the transductive SVM performs better than the naive  $k$ -nn approach. Also, it is interesting to see that supervised learning works well when trained and tested on the same kind of data, but when tested on data from a different environment, it may fail as in our example. To overcome such problems our adaptive SSL method seems to be an appropriate approach.

Note that our adaptive TSVM approach gives somewhat worse results than the standard SVM method on Caltech101. This is because the clustering step for this data set had to be done using the HMP features and not SIFT, as for our own data: the appearances of the objects in Caltech 101 are simply too diverse to compare them using SIFT. However, we experienced that spectral clustering works worse on HMP features, which means that for Caltech 101 the training data provided to TSVM was of less quality than if we had chosen standard supervised learning. For our evaluation, this is however of little importance, as our method anyhow aims at adapting to a given environment with no previously labelled data where objects of the same class are not very diverse. An application of our method to an environment-independent, pre-labelled data set such as Caltech101 is therefore not very meaningful.

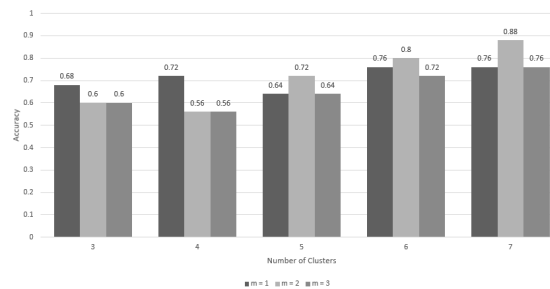
#### 4.3 Number of Generated Label Queries

In another experiment, we investigated the correspondence of the number of label queries made by the algorithm and the classification accuracy. There are two parameters that can be set: the number of clusters  $c$  and the number  $m$  of patches per cluster, which receive a label after the query (see above). On one side, we want to have few clusters, i.e.  $c$  should be low. However, if there are more clusters, then the clusters are smaller and therefore *purer*, i.e. there are more elements that agree on the true class label. Purer clusters means that we

Learning method	Caltech 101	Our data
standard SVM	95.25%	52.00%
adaptive $k$ -nn SSL	55.86%	58.00%
adaptive TSVM	81.43%	88.00%

**Table 2.** Classification accuracy of standard SVM learning and adaptive SSL methods on different data sets. The standard SVM was trained on a subset of Caltech 101 in both cases. Thus, while standard supervised learning gives good results when training and test data are similar, it can perform badly when they are dissimilar. However, our adaptive SSL performs much better, because it queries the relevant class labels from the data before learning the classifier. From the two considered methods, transductive SVMs perform better than the  $k$ -nearest neighbour method.

can increase  $m$ , without assigning wrong labels to patches, thus we obtain better training data. This relationship is shown in Fig. 6. If the number of clusters is small, we get the best accuracy for  $m = 1$ . But for more clusters,  $m = 2$  is better, because by assigning the same label to the first  $m$  elements of each cluster, we get fewer wrong labels. In general we found that having less labels for training is better than having more, but wrong labels.



**Fig. 6.** Accuracy vs. number of clusters and number  $m$  ( $m = 1, 2, 3$ ) of patches receiving a label from the query. More clusters lead to a higher cluster purity. Then, higher values of  $m$  are more effective, because the tSVM receives better training data.

## 5 Discussion and Conclusions

Our proposed approach for adaptive semi-supervised learning for object detection in indoor environments has two major advantages over standard supervised learning methods: first, it is able to select informative data to learn from and to adapt to a given environment by only querying labels for currently observed, situation-relevant data and using them to train a classifier. And second, it reduces the number of required user interactions by making more informed

questions about the data based on a pre-clustering step. Our experiments show that the proposed approach can outperform standard non-adaptive supervised learning when applied to environment-dependent data.

*Acknowledgment* This work was funded by the EU project SPENCER (ICT-2011-600877).

## References

1. X. Zhu, "Semi-supervised learning literature survey," Computer Sciences, University of Wisconsin-Madison, Tech. Rep. 1530, 2005.
2. ———, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, 2005.
3. N. D. Lawrence, J. C. Platt, and M. I. Jordan, "Extensions of the informative vector machine," in *Proc. of the First Intern. Conf. on Deterministic and Statistical Methods in Machine Learning*. Springer-Verlag, 2004, pp. 56–87.
4. A. Saffari, C. Leistner, and H. Bischof, "Regularized multi-class semi-supervised boosting," in *Conf. on Comp. Vision & Patt. Recog. (CVPR)*, 2009.
5. T. Joachims, "Transductive inference for text classification using support vector machines," 1999, pp. 200–209.
6. R. Triebel, R. Paul, D. Rus, and P. Newman, "Parsing outdoor scenes from streamed 3d laser data using online clustering and incremental belief updates," in *Robotics Track of AAAI Conference on Artificial Intelligence*, 2012.
7. M. Guillaumin, J. Verbeek, and C. Schmid, "Multimodal semi-supervised learning for image classification," in *Conf. on Comp. Vision & Patt. Recog. (CVPR)*, 2010.
8. S. Ebert, D. Larlus, and B. Schiele, "Extracting structures in image collections for object recognition," in *European Conf. on Comp. Vision (ECCV)*, 2010.
9. I. Budvytis, V. Badrinarayanan, and R. Cipolla, "Semi-supervised video segmentation using tree structured graphical models," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2751–64, Nov 2013.
10. B. Settles, "Active learning literature survey," Tech. Rep., 2010.
11. A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Gaussian processes for object categorization," *Intern. Journal of Computer Vision*, vol. 88, no. 2, pp. 169–188, 2010.
12. R. Triebel, H. Grimmett, R. Paul, and I. Posner, "Driven learning for driving: How introspection improves semantic mapping," in *The International Symposium on Robotics Research (ISRR)*, 2013.
13. D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the Intern. Conf. on Computer Vision (ICCV)*, 1999, pp. 1150–1157.
14. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, Nov 2012.
15. A. Rosenberg and J. Hirschberg, "V-measure: A conditional entropy-based external cluster evaluation measure," in *Proc. of the Joint Conf. on Empirical Methods in Natural Language Proc. and Comp. Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 410–420.
16. U. Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.
17. L. Bo, X. Ren, and D. Fox, "Hierarchical matching pursuit for image classification: Architecture and fast algorithms," in *In NIPS*, 2011.