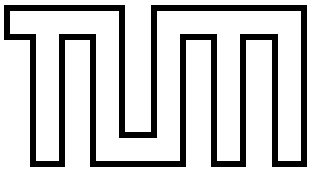


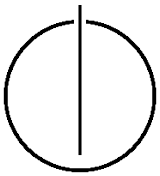
FAKULTÄT FÜR INFORMATIK

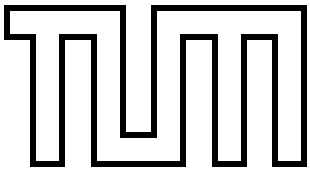
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Feature Selection and Learning for Semantic Segmentation

Caner Hazırbaş





FAKULTÄT FÜR INFORMATIK

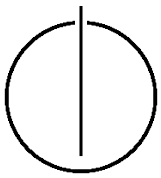
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Feature Selection and Learning for Semantic Segmentation

Selektion und Lernen von Merkmalen für Semantische Segmentierung

Author:	Caner Hazırbaş
Supervisor:	Prof. Dr. Daniel Cremers
Advisors:	M.Sc. Julia Diebold
	Dipl.-Inf. (Univ.) Mohamed Souiai
Date:	June 26, 2014



I confirm that this master's thesis is my own work and I have documented all sources and material used.

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

München, den 26. Juni 2014

. . . . . .

Caner Hazırbaş

Acknowledgments

First and foremost, I would like to offer my sincere gratitude to Prof. Dr. Daniel Cremers who gave me an invaluable guidance to decide on my research subject and also motivated me with his immense knowledge on the topic.

Besides my supervisor, I would like to offer my special thanks to my advisors, M.Sc. Julia Diebold and Dipl.-Inf. (Univ.) Mohamed Souiai, for being the most indestructible advisors ever against my countless questions, for endless motivation and their support during the thesis. Without their assistance and dedicated involvement in every step throughout the process, this thesis would have never been accomplished. Furthermore, I particularly thank to Dr. Rudolph Triebel for his kindness and assistance to solve my problems regarding to machine learning.

"Success is not a target but a very long journey." I could not make it through and would have been lost without my friends. I thank my dearest friends, Bilal Porgalı and Yağmur Sevilmiş for being with me through this exhausting journey.

My special thanks are extended to my friends, my fellow labmates in the Computer Vision and Pattern Recognition Research Group at TU Munich: Alfonso Ros Dos Santos, Sahand Sharifzadehgolpayegani, Zorah Lähner, Emanuel Laude, Benjamin Strobel and Mathieu Andreux for the stimulating discussions and also for having a great time together.

Last but not the least, I would like to thank my mother Hüsniye Hazırbaş for giving me moral in the very dark moments and spiritual support throughout my life, my father Birol Hazırbaş for being with me all the time with his prayers, and also my beloved sister Melike Hazırbaş for giving me all her trust and support.

Abstract

This work presents a comprehensive study on feature selection and learning for semantic segmentation. Various types of features, different learning algorithms in conjunction with minimizing a variational formulation, are discussed in order to obtain the best segmentation of the scene with minimal redundancy in the feature set. The features are scored in terms of relevance and redundancy. A clever feature selection reduces not only the redundancy but also the computational cost of object detection. Additionally different learning algorithms are studied and the most suitable multi-class object classifier is trained with the selected subset of features for detection based unary potential computation. In order to obtain consistent segmentation results we minimize a variational formulation of the multi-labelling problem by means of first order primal-dual optimization.

Experiments on different benchmarks give a deep understanding on how many and what kind of features and which learning algorithm should be used for semantic segmentation.

Contents

Part I.

Introduction and Theory

1. Introduction

Building man-like machines, also called humanoid robots, has been a dream of humanity since the life started. The fundamental problem to address in developing such robots is how to understand the environment as we humans do. For many years, human vision has kept its secret in visual understanding of the scene and making robust inference about unknown environments or objects using previous experience. Although scientists have spent many years to figure out how humans actually see the environment, we are still struggling to express how this perfect learning and fully understanding happens. Nevertheless, we know that the eyes allow individuals to interpret their surroundings and therefore we developed cameras. Invention of the cheap cost cameras has opened a gate for robots in the direction of visual perception. Now, cameras are taking an important role on visual understanding in many applications.

Development of the autonomous, unmanned vehicles and devices is essential not only to replace man-power with mechanical power, but also to support the daily life activities. Transportation, production, surveillance systems, aero-space industry and human supporting systems are only some of application areas. However, the remaining major challenge is how to understand the environment and move autonomously using cameras. Perceptual understanding of the scene is at the very heart of this problem and is still being studied by many scientists.

Computer vision studies investigated perceptual interpretation of the objects in the scene using just a single image, retrieved from a low-cost camera. Today, with the high computational processing power of computers, many researchers consider recognition and segmentation problems together to increase the capacity of current systems for perceptual scene analysis. The integration of the visual semantics of objects into the segmentation problem is nowadays the main concern in the field of semantic segmentation.

Segmentation of images based on semantics is essential for real-time scene understanding, surveillance systems and 3D scene reconstruction, *e.g.* in [5, 26, 46]. The performance of these segmentation algorithms heavily depend on the quality of the appearance model which is computed using pixel-wise object detection algorithms.

The major challenge is to find the most representative features to distinguish dissimilar objects in terms of their shape, color and textural differences. Conventional object detectors deal with the task of finding bounding boxes around each object [14, 32, 49]. In contrast, dense object detection approaches [29, 44] focus on detecting the objects at pixel level which provides a preliminary segmentation. Several other studies pursued different approaches for object detection and learning algorithms in order to perform semantic segmentation, *e.g.* [3, 28, 43].

In this work we propose a method to systematically select the best subset of features for semantic segmentation. Our approach is able to outperform state-of-the-art results in terms of runtime and even in accuracy in several cases.

1.1. Related Work

In 2001, Viola and Jones presented simple, but robust Haar-like features for real-time face detection [49]. Haar-like features are simple to implement, computationally low cost and very accurate in capturing the shape of objects. These features can be used for detection of any type of object and this flexibility makes them convenient for semantic segmentation applications. Lowe [32] presented a distinctive, scale-invariant feature transform (SIFT) and Dalal and Triggs [14] presented so called histograms of oriented gradients (HOG) which are computationally more expensive. SIFT can robustly identify the objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. Moreover, Zhang *et al*. [52] proposed a set of novel features based on oriented gradients to detect the cat heads. They have shown that exploiting shape and texture features jointly outperforms the existing, leading features, *e.g*. Haar, HOG. However, these proposed methods are mostly used in the context of sliding window techniques. In the scope of this research we aim at detecting the objects densely at pixel level.

In [44], Shotton *et al*. proposed texture-layout filters based on textons which jointly model patterns of texture and their spatial layout for dense object detection. They presented a two-step algorithm for semantic segmentation of photographs. In the first step unary classification and feature selection are achieved using shared boosting to give an efficient classifier. Then, in the second step, an accurate image segmentation is achieved by incorporating an unary classifier in a conditional random field to enforce the neighboring pixels to have same labels.

Ladický *et al*. [29] proposed a hierarchical random field model, that allows integration of features computed at different levels of the quantisation hierarchy. Moreover, Ladický *et al*. [30] combined different features for unary pixel classification by using Joint Boosting [48]. However, their approach is sensitive to a large set of parameters.

Fröhlich *et al*. [21] proposed an iterative approach for semantic segmentation of a facade dataset. They learn a single random forest and incrementally add context features derived from coarser levels. This approach uses different kinds of features in a joint, flexible, and fast manner and refines the semantic segmentation of the scene iteratively.

Hermans *et al*. [26] discussed 3D semantic reconstruction of indoor scenes by exploiting 2D semantic segmentation approach based on Randomized Decision Forests for RGB-D sensor data. They used depth features in addition to a very basic set of features to train a classifier for unary pixel classification.

However, neither of the approaches above does give any justification for the chosen set of features nor address the problem of how to choose the best feature set for object detection in semantic segmentation. Feature selection plays an important role on the quality of the object detector and also on runtime of the entire system. It is also known that the m best features are not necessarily the best m features [12].

In [37], Peng *et al*. proposed a theoretical framework to rank the features based on minimum-Redundancy-Maximum-Relevance (mRMR). They formulate a cost function composed of a relevance and a redundancy term. The relevance between features and class labels is maximized and the redundancy between feature pairs is minimized.

1.2. Contributions

In this work, we adapt the approach of [37] to the task of semantic segmentation. We study different types of features and learning algorithms and perform experiments on various segmentation benchmarks. The main goal is to find the best subset of features for semantic segmentation. This is done by systematically selecting them via minimizing redundancy and maximizing relevance and thereby improving the unary pixel potentials. Exhaustive experiments on various benchmarks show that reducing the redundancy in feature sets significantly reduces the runtime while preserving the performance. Our approach is able to outperform state-of-the-art algorithms in terms of runtime and even in accuracy in several cases.

2. Image Segmentation and Object Recognition

This chapter presents the theoretical background of the proposed method. Variational image segmentation, object learning (classification) algorithms and mutual information based feature selection method are discussed in detail to provide an insight into image segmentation and object recognition.

2.1. Variational Image Segmentation

Image segmentation is recently the most popular research topic in computer vision. It deals with finding disjoints sets of elements from an observation data such that similarity inside the sets and dissimilarity in-between the disjoint sets are maximum.

Image segmentation contains finding object regions and accurate boundaries between them. Let Ω be the image domain and k be the number of disjoint regions, the segmentation criteria of an image can be written as:

$$\Omega = \{\Omega_1, \Omega_2, ..., \Omega_{k-1}, \Omega_k\},$$

where

$$\bigcup_{i=1}^{k} \Omega_i = \Omega \quad \wedge \quad \Omega_i \cap \Omega_j = \emptyset \;\; \forall_{i,j} \in \{1, .., k\} \;\; \text{s.t.}\, i \neq j.$$

In the simplest case, with only background and foreground, regions can be segmented using very basic and conventional methods such as edge-detection based segmentation. However, since this is not realistic, one should take more complex cases into consideration to be able to segment the images into multiple regions. The image segmentation approaches are generally composed of graphical based models, *e.g.* Graph-Cut based image segmentation [16] or variatonal image segmentation such as the generalized Mumford-Shah Functional [36] for multi-region segmentation.

In this study, we only focus on the variational multi-region segmentation problem. Variational image segmentation minimizes an energy functional to find the consistent segmentation. An energy functional for image segmentation is typically formulated as:

$$E(u(x)) = \sum_{i=1}^{k} E_{data}(u_i(x), I) + \lambda \cdot \sum_{i=1}^{k} E_{reg}(u_i(x))$$

where $u(x) = (u_1, ..., u_k)$ is the set of indicator functions with:

$$u_i(x) = \begin{cases} 1 & \text{for } x \in \Omega_i \\ 0 & \text{else} \end{cases}$$

E_{data} is called data term and presents the fidelity of the segmentation of a given image I. E_{reg} is called regularizer term and ensures the resulting segmented regions should have homogeneous class labels with optimum class boundaries. The parameter λ is a weight and denotes the amount of smoothing on the segments. The trade-off for the parameter λ is that the segmented regions get more and more smoothed as the λ increases. If the parameter λ is set to a very low value, then the segmented regions will have noisy labels and lose homogeneity property.

2.1.1. Mumford-Shah Functional

Mumford and Shah [36] presented an energy functional to segment images with piece-wise smooth approximation in the following form:

$$E(u, C) = \int_{\Omega} (I - u)^2 dx + \lambda \int_{\Omega \setminus C} |\nabla u|^2 dx + \nu |C| \tag{2.1}$$

where $u \colon \Omega \to \mathbb{R}$ is an approximation of the segmented image and $C \subset \Omega$ is the one-dimensional discontinuity set. $(I - u)$ converges to zero while u gets similar to I and this provides a good approximation of the segments. The second term smooths the regions everywhere except for the set C. The last term ensures that the length of boundary $|C|$ between classes is minimal with a weighting ν. However, C is part of the energy itself and no numerical solution is given to minimize this functional. There are multiple approaches presented to solve the Mumford-Shah functional. This thesis employs a related model of the Mumford-Shah functional to solve multi-region segmentation problems.

2.1.2. The ROF Model

In 1992, Rudin-Osher-Fatemi [40] pioneered the concept of nonlinear image denoising which aims at removing noises by preserving the edges on an image. The principle is to remove unwanted excessive and possibly spurious details, induce high total variation, that is, the integral of the absolute gradient of the image. According to this principle, reducing the total variation of the image removes the unwanted details and preserves the image edges.

Given a noisy image $f \colon \Omega \to \mathbb{R}$, where Ω is bounded open subset of $\mathbb{R}^2$, we seek for a clean image u, the denoised image version of f with bounded variation. To solve this problem, Rudin *et al*. minimize the total variation of an image with the following energy functional:

$$\min_{u \in BV(\Omega)} \frac{1}{2} ||f - u||_2^2 + \lambda \int_{\Omega} |\nabla u| dx \,. \tag{2.2}$$

This variational energy has a data term, minimizes the L_2 norm of desired and noisy image and a regularizer $\int_{\Omega} |\nabla u|$, so called total variation, minimizes the length of object boundaries. λ is a scale parameter that determines the level of smoothing.

Total variation is not differentiable under the constraint that u takes only 0 and 1. For this reason, bounded variation of u is used to address the problem; $u \in BV(\Omega)$ is constrained

to be of bounded variation, defined as:

$$BV(\Omega) := \{u \in L^1(\Omega) \,\big|\, TV(u) < +\infty\}$$

where the total variation is finite and given as [22]:

$$TV(u) := sup\Big\{ - \int_\Omega u(x) \cdot \operatorname{div} \xi dx \,\Big|\, \xi \in C_c^1(\Omega, \mathbb{R}^n), ||\xi||_{L^\infty(\Omega)} \leq 1\Big\} \tag{2.3}$$

If u is differentiable and Ω is a bounded open set, then we can re-write the total variation equation using Gauss' theorem:

$$\int_\Omega |\nabla u| = \int_\Omega -u \cdot \operatorname{div} \xi = \int_\Omega \nabla u \cdot \xi \leq ||\xi||_\infty \int_\Omega |\nabla u|$$
$$\text{yielding equality with } \hat{\xi} := \frac{\nabla u}{|\nabla u|} = \int_\Omega \nabla u \cdot \hat{\xi} = \int_\Omega |\nabla u|$$

and $\hat{\xi}$ can be approximated by a sequence $(\xi)_n \subset C_c^1(\Omega)$ it holds the following:

$$\int_\Omega -u \cdot \operatorname{div} (\xi)_n = \int_\Omega \nabla u \cdot (\xi)_n \longrightarrow \int_\Omega \nabla u \cdot \hat{\xi} = \int_\Omega |\nabla u| \,.$$

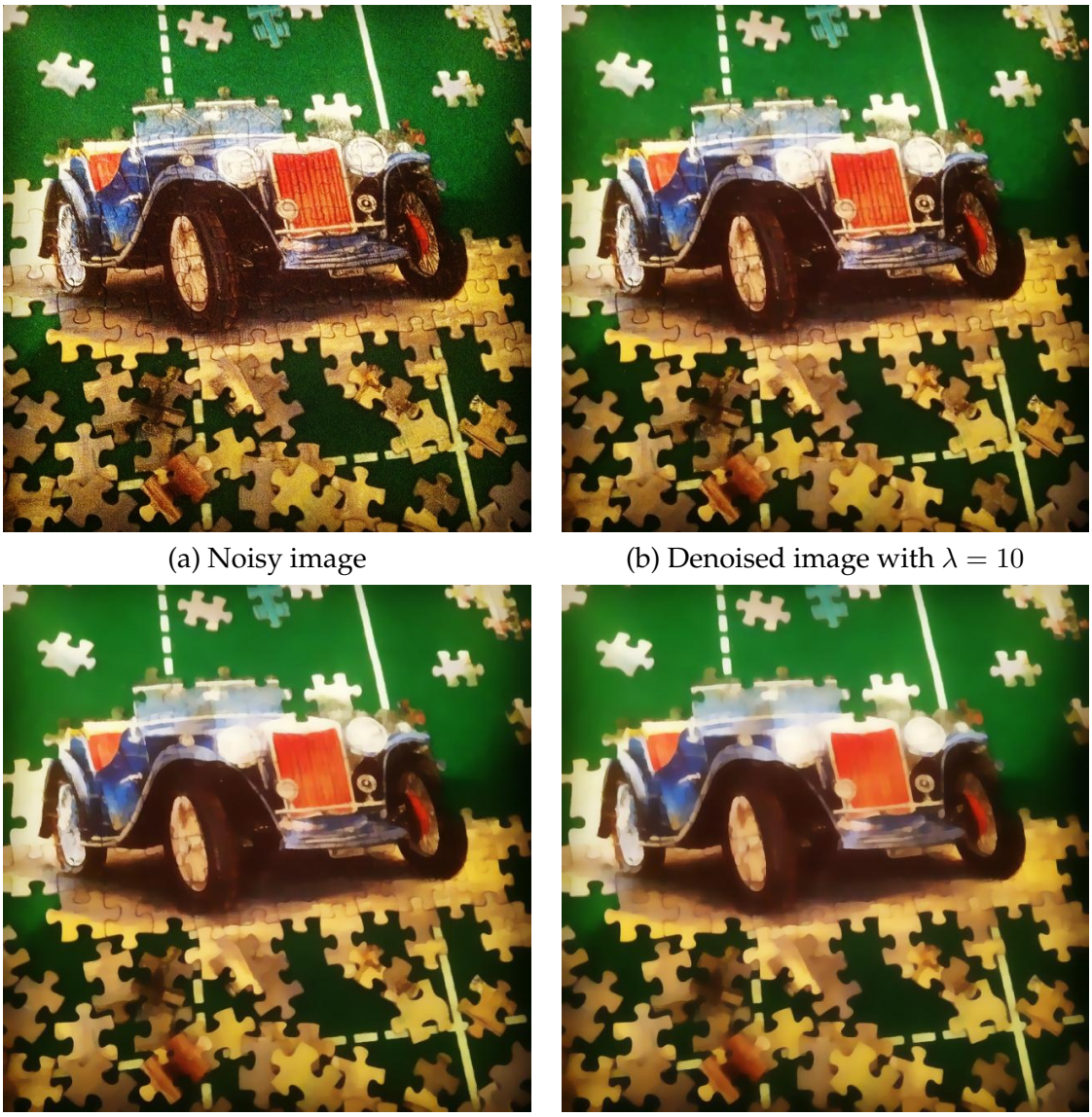
The ROF model is also called $TV - L^2$ model and it has two advantages;

- TV preserves discontinuity (edges on the images) and therefore the method is well suited for image processing tasks,
- TV is a convex function; a function f is convex if and only if its epigraph $epi(f) := \{(x, y) | f(x) \leq y\}$ is a convex set. And it can be easily shown that if f is convex and there exist a minimizer of f, then its minimizer is indeed globally optimal. Convex functions have the advantage to be minimized with well-researched optimization solvers regardless of initialization .

Equation 2.2 can be minimized with gradient descent and the equation for this task is defined by:

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) - \frac{1}{\lambda}(u - f)$$
$$\text{with boundary condition } \frac{\partial u}{\partial n}\Big|_{\partial\Omega} = 0$$

However, in the case of constant functions, $|\nabla u|$ will be equal to 0 and the optimization scheme will face the singularity problem because of zero division. Therefore, this problem will be circumvented using a first-order primal-dual formulation of the Total-Variation given in Equation 2.3 . First-order primal-dual convex optimization scheme will be discussed in Section 4.2 . See Figure 2.1 for an image denoising example with ROF model.



(a) Noisy image

(b) Denoised image with $\lambda = 10$

(c) Denoised image with $\lambda = 30$

(d) Denoised image with $\lambda = 50$

Figure 2.1.: **Image Denoising.** Sample noisy image (a) was denoised with gradient descent minimization on the ROF model, implemented by Magiera and Löndahl on Matlab [33]. The denoised images (b), (c), (d) were produced with 100 iterations and different smoothing scales (λ). Each RGB color channel was processed independently. The resulting image gets smoother as the λ increases.

2.1.3. Total-Variation Segmentation

A continuous model for TV based image segmentation, that will be used, has the following form:

$$\min_{\Omega_1,\dots,\Omega_k} \sum_{i=1}^{k} \int_{\Omega_i} f_i(x)\,dx + \frac{\lambda}{2} \sum_{i=1}^{k} Per(\Omega_i, \Omega)$$

where $f_i \colon \Omega \to \mathbb{R}_+$ are non-negative potential functions defined for each pixel with object classification methods. The λ-weighted second term, is half the sum of the perimeters of the sets $\Omega_1, ..., \Omega_k$ and corresponds to the total length of the partition interface $\bigcup_{i<j} \partial\Omega_i \cap \partial\Omega_j$ (in order to count each perimeter once.). Thus, we segment the image into k sub-partitions that yield the lowest energy and ensure a minimal surface at the same time.

The partitions are represented by characteristic functions, also known as indicator, functions $u_1, ..., u_k \colon \Omega \to \{0, 1\}$ so that the energy takes a computationally tractable form:

$$u_i(x) = \begin{cases} 1 & \text{for } x \in \Omega_i \\ 0 & \text{else} \end{cases} \quad \text{satisfying} \quad \sum_{i=1}^{k} u_i(x) = 1 \text{ a.e. } x \in \Omega$$

Thus, the first term becomes:

$$\sum_{i=1}^{k} \int_{\Omega_i} f_i(x)dx = \sum_{i=1}^{k} \int_{\Omega} u_i(x) f_i(x)dx$$

Moreover, if indicator functions $u_i \in BV(\Omega)$ of measurable sets $\Omega_i \subset \Omega$ are scalar-valued functions of bounded variation, the co-area [17] formula tells us that the the perimeter is equal to the total variation:

$$Per(\Omega_i, \Omega) = Per(u_i, \Omega) = TV(u_i) = \int_{\Omega} |\nabla u_i| dx.$$

As a consequence of the re-formulations of the problem, the energy functional takes the form of an intermediate minimization problem as follows:

$$\min_{u \in \mathcal{B}} \sum_{i=1}^{k} \int_{\Omega} u_i(x) \cdot f_i(x)\,dx + \frac{\lambda}{2} \sum_{i=1}^{k} \int_{\Omega} g(x) \cdot |\nabla u_i|\,dx\,. \tag{2.4}$$

$$\mathcal{B} := \left\{ (u_i, ..., u_k) \in BV(\Omega, \{0, 1\})^k \,\middle|\, \sum_i u_i = 1 \text{ a.e. } x \in \Omega \right\}$$

with coherency constraint $\mathcal{B}$ on the indicator functions u_i. The minimizer $\mathbf{u}$ now lies on $\mathcal{B}$, that is the k- dimensional space of binary functions with bounded variation and fulfills the point-wise characteristic property (at every position x only one u_i is allowed to be non-zero). Also note that, TV in Equation 2.4 is weighted with a space-dependent $g(x)$ function given in Equation 2.5 , which takes low values if the derivative at position x is high and takes high values otherwise, meaning that discontinuity will be preserved on the boundaries of the input image.

$$g(x) = \exp\left(-\frac{|\nabla f(x)|^2}{2\sigma^2}\right),\ \sigma^2 = \frac{1}{|\Omega|}\int_\Omega |\nabla f(x)|^2 dx \tag{2.5}$$

The energy functional is still hard to optimize since it is non-convex and non-differentiable because of the binary indicator functions (u_i). Therefore, to convexify the energy, the u_i functions are relaxed by mapping them into the whole interval $u_i \colon \Omega \to [0,1]$ as described in [9]. With this, indicators change from hard to soft assignments, thus at a position x, $u(x)$ can have multiple non-zero entries although the coherecency constraint still holds in that case, *i.e.* the sum has to yield 1. TV is replaced with its dual representation (Equation 2.3) to be a differentiable function.

The energy function now takes the form of a convex and differentiable optimization problem as follows:

$$\min_{u\in\mathcal{S}} \sup_{\xi\in\mathcal{K}} \sum_{i=1}^{k} \int_\Omega u_i(x)\cdot f_i(x)\,dx - \lambda \sum_{i=1}^{k} \int_\Omega \operatorname{div}\xi_i(x)\cdot u_i(x)\,dx\,. \tag{2.6}$$

with minimization over the set $\mathcal{S}$ of BV functions moving into the k-dimensional simplex and the dual variable ξ is in the convex set $\mathcal{K}$:

$$\mathcal{S} := \left\{ \mathrm{u} = (u_1,...,u_k) \in BV(\Omega,[0,1])^k \,\middle|\, \sum_i u_i(x) = 1 \ \text{ a.e } \ x\in\Omega \right\}.$$

In [31, 51], the authors use variations of a straight-forward formulation that arises from the TV definition of the regularizer. ξ is enforced to stay inside a norm boundary, *e.g.* $(\sum_i ||\xi||^2)^{\frac{1}{2}} \leq 1$. This enforcement limits the amount of flow happening at every position. Since, we weight the TV with a space-dependent weighting function g, the dual space can be re-written as:

$$\mathcal{K} := \left\{ \xi = (\xi_i,...,\xi_k) : \Omega \to \mathbb{R}^{2\times k} \,\middle|\, \sqrt{\sum_i ||\xi_i(x)||^2} \leq \frac{g(x)}{2} \ \text{ a.e. } \ x\in\Omega \right\}.$$

Alternative approach is presented by Chambolle *et al.* [10]. They introduce the notion of a paired calibration of the dual variables' components that represents a local convex envelope of the energy:

$$\mathcal{K}_C := \left\{ \xi = (xi_i,...,x_k) : \Omega \to \mathbb{R}^{2\times k} \,\middle|\, |\sum_{i_1\leq i\leq i_2} \xi_i(x)| \leq \frac{g(x)}{2} \ \ (a.e.) \ \ x\in\Omega \right\}.$$

It is proven that this model has a tighter bound for $k > 2$ in comparion to others which means that the found minimizer is closer to the original global optimum. This dual space creates tighter solutions and the projection of a variable onto $\mathcal{K}_C$ is computationally burdensome, because it involves the projection of a variable onto multiple convex sets. On the other hand, the projection onto $\mathcal{K}$ is very fast since it consists of point-wise truncation operations.

The relaxed energy in Equation 2.6 is just an approximation of the original problem and does not guarantee an optimal solution. The literature has shown that in the case of two regions ($k = 2$), the thresholded solution of the relaxed version yields a global optimizer independent of the chosen threshold. Nevertheless, this assumption no longer holds for any binarized solution in the multi-region case.

Data: $(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Result: Final strong classifier $H(x)$
for $t = 1, .., T$ **do**

- Train a weak learner using weight distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

  where Z_t is a normalization factor such that D_{t+1} will be a distribution.

end
Output the final hypothesis:

$$H(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t h_t(x)\Big).$$

Algorithm 1: **Discrete AdaBoost.** The Adaptive Boosting Algorithm

2.2. Object Classification Methods

In machine learning and statistics, classification is the problem of identifying the categorical class label of a new observation, on the basis of training data which contains observations whose category memberships are known. The output of a classification algorithm is called *classifier*, that is, a function maps the observation data to a category (a class label).

This section gives a brief description of Adaptive Boosting [20] and Random Forests [7] machine learning algorithms which are used in the proposed method among various linear and non-linear supervised learning algorithms, *e.g.* Artificial Neural Networks [39], Support Vector Machines [4], linear classifiers, deep learning.

2.2.1. Adaptive Boosting

The AdaBoost is a boosting algorithm, introduced by Freud and Schapire [18] in 1997 and is capable of solving the binary classification problem. Adaptive Boosting is a linear discriminative learning model which combines weak learners as strong classifier to find the best separation between two classes. Algorithm 1 shows the algorithm of Discrete AdaBoost [19].

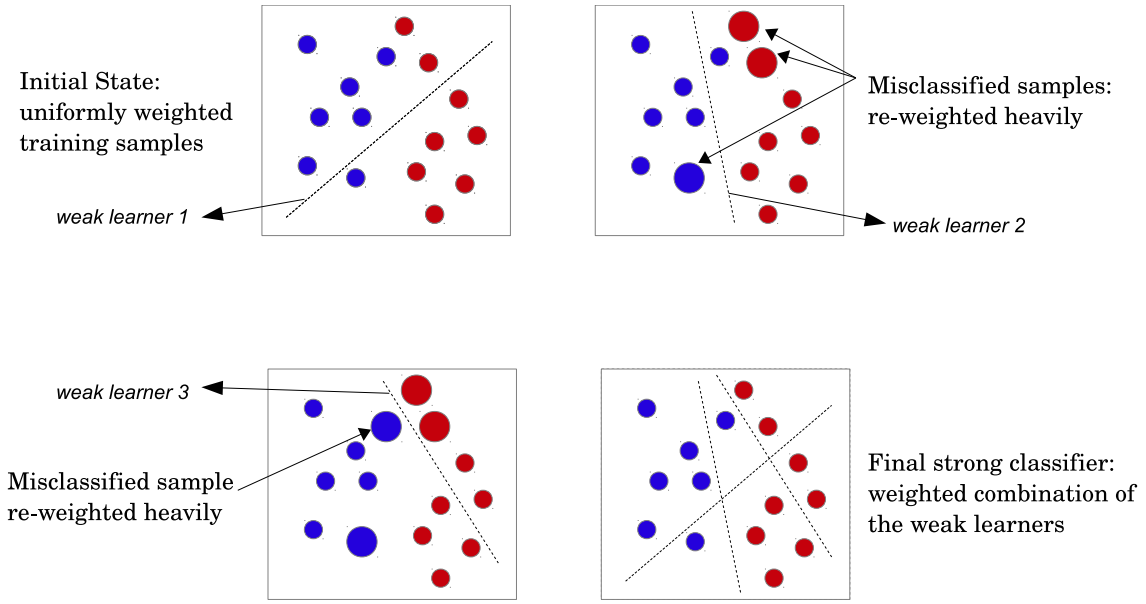


Figure 2.2.: **Boosting Classifier.** Illustration of the Adaptive Boosting Algorithm.

Given an observation set (X) with corresponding class labels (Y), the algorithm iteratively finds the weak learners and combines them linearly to a strong classifier. Usually a weak learner is a decision tree with one level, also called decision stump.

After receiving the weak hypotheses h_t, the algorithm computes the importance rate α_t of assigned to h_t. Note that α_t is proportional to ϵ_t and gets larger if ϵ gets smaller. One of the main ideas of the algorithm is to maintain a distribution of weights (D_i) over the training set. In each round, weights of misclassified samples are increased and correctly classified ones are decreased. As a result of this re-weighting, the weak learner h_t is forced to focus on hard examples in the training set. Thus, the weight tends to concentrate on hard examples.

The *final hypotheses* H, which is also called strong classifier, is a majority vote of linearly combined weak learners h_t, weighted with corresponding α_t.

The weak learner is responsible to find a *weak hypotheses* $h_t : X \to \{-1, +1\}$ appropriate for the sample weights D_t and the quality of this weak learner is measured by its error:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x) \neq y_i} D_t(i).$$

Figure 2.2 illustrates the boosting algorithm on a given simple training set. Strong classifier is shown as a combination of three weak classifiers.

There are different kinds of Adaptive Boosting algorithm and each of them is named depending on the error function:

- **Real AdaBoost** : The output of decision trees is a class probability estimate $p(x) = P(y = +1|x)$, the probability that x is in the positive class [20]. Each leaf node in the

decision tree is changed to output half the logit transform of its previous value:

$$f_t = \frac{1}{2} \ln \left(\frac{1-x}{x} \right).$$

- **Logit AdaBoost** : This is a convex optimization problem and minimizes the logistic loss:

$$\sum_i \log \left(1 + e^{-y_i f(x_i)} \right), \text{where } f(x_i) = \alpha_i h_t(x_i)$$

- **Gentle AdaBoost** : Previous boosting algorithms choose f_t greedily, minimizing the overall test error as much as possible at each step Gentle AdaBoost features a bounded step size. f_t is chosen to minimize $\sum_i w_{t,i}(y_i - f_t(x_i))^2$, and no further coefficient is applied. Thus, in the case where a weak learner exhibits perfect classification performance, Gentle AdaBoost will choose $f_t(x) = \alpha_t h_t(x)$ exactly equal to y, while steepest descent algorithms will try to set $\alpha_t = \infty$. Empirical observations about the good performance of Gentle AdaBoost appear to back up Schapire and Singer's remark that allowing excessively large values of α can lead to poor generalization performance [19, 41].

2.2.2. Random Forests

Random Forests is an ensemble learning method, proposed by Breiman [7] in 2001 and can handle multi-class classification problems. Random Forests grow many decision trees, introduced in [8], to construct more robust classifier against outliers.

The construction of a decision tree requires a training sample of m observations:

$$O = (x_1, y_1), ..., (x_m, y_m), i = 1, .., m$$

where $x_i \in X$, variable set and $y_i \in Y$, corresponding class labels. A sample constructed decision tree is shown in Figure 2.3 . Each interior node corresponds to a variable in X, while each leaf node corresponds to a class label in Y. A decision tree is trained by splitting the input training set into subsets based on an attribute value test. Splitting is repeated recursively on each node until all samples on the node have same target class label or splitting no longer gives noticeable improvement on the predictions.

A decision tree is usually constructed from top to down and at each interior node, one attribute is selected, which gives the best split of the subset. The quality of a split is determined by the average value of importance of each attribute, computed with the following metrics:

- **Information Gain**: The concept of information gain comes from information theory and it is based on entropy measurement. Information gain of an attribute gives us the importance of that attribute given set of examples. Let T denote a set of training samples, each has the form of $(x, y) = (x_1, x_2, ..., x_k, y)$ where $x_a \in V(a)$ is the

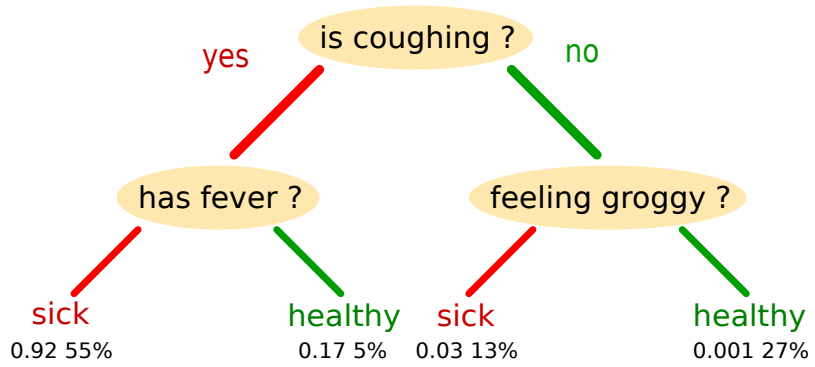


Figure 2.3.: **Decision Tree.** A tree showing the sickness of a child in a city where there is an epidemic disease. The numbers under the leaf nodes show the probability of sickness and the percentage of observations in the leaf.

value of ath attribute of the sample x and y is the corresponding class label. The information gain for an attribute a is defined in terms of entropy H as follows [35]:

$$IG(T,a) = H(T) - \sum_{v \in V(a)} \frac{|\{x \in T | x_a = v\}|}{|T|} \cdot H(\{x \in T | x_a = v\})$$
$$H(X) = -\sum_i P(x_i) \cdot \log_{|k|} P(x_i)\,, \sum_i P(x_i) = 1$$

- **Gini Importance**: Random Forests gain its superior performance from its implicit feature selection strategy. In this strategy, Gini importance is the indicator of feature relevance. This metric measures how often a particular attribute θ is selected for a split and how effective its overall discriminative value for the classification. At each node t within a binary decision tree, the optimal split is found using Gini impurity $i(t)$ and calculated as follows [34]:

$$i(t) = 1 - p_1^2 - p_0^2.$$

where $p_k = \frac{n_k}{n}$ is the fraction of the n_k samples from class $k = \{0, 1\}$. By following this formula Gini importance I_G of all variables θ is computed as:

$$I_G(\theta) = \sum_t \Delta i_\theta(t)$$
$$\Delta i(t) = i_\theta(t) - p_l i_\theta(t_l) - p_r i_\theta(t_r)$$
$$p_l = \frac{n_l}{n}\,, p_r = \frac{n_r}{n}$$

A decision tree has many advantages compared to the other learning methods. It is simple to construct, very robust and reliable since it has a self-validation strategy, can handle both categorical and numerical data, performs well even in case of a large dataset and most importantly, a decision tree is fast to predict the class label of the new observations. A decision tree has also some drawbacks; the quality of the classifier highly depends on the

training samples and a decision tree may over-fit and not generalize well on the training data.

Random Forests consist of multiple decision trees. To train a forest, the initial training data is split into subsets such that each tree has the same amount of samples. Each tree in the forest gives a vote (classification) which is a mapping from an input vector to a class label and the forest chooses the classification having the most votes over all the trees in the forest.

In contrast to single decision tree, Random Forests do not over-fit. Adding more trees to the forest increases the overall performance of the classifier, but the prediction time also increases. Nevertheless, Random Forests can be easily parallelized since the prediction of the new observation is evaluated at each tree independently. In Random Forests, there is no need for an external test or cross-validation to estimate the forest error since the error computation is done internally during training. One-third of the training subset of each tree is left out to estimate the forest error while the trees are constructed. This error is called *the out-of-bag* error and gives the unbiased classification error of the forest.

Moreover, in the original paper of Random Forests [7], it is shown that the forest error depends on two factors which are the correlation between trees and the strength of each individual tree in the forest. The forest error increases as *the correlation* gets higher, but decreases as *the strength* increases.

Random Forests are scalable, reliable, robust, insensitive to the outliers, capable of solving multi-class classification problem on high dimensional data and also outperform the other machine learning algorithms in terms of speed and accuracy. The strong characteristics of Random Forests make them popular in the area of visual object detection and recognition.

2.3. Feature Selection based on Mutual Information Criteria

Conventional machine learning algorithms aim to find the best subset of features which distinguish the disjoint sets (classes) well for classification. However, most of the methods suffer from local minima or over-fitting problem due to high sensitivity to outliers and the final classifier does not generalize well on the given training data which is usually noisy because of the high redundancy in the feature set. Therefore, in statistical analysis, this problem is addressed with feature reduction using some probabilistic frameworks such as maximizing the dependency. This feature analysis provides minimal classification errors by maximizing the statistical dependency of the target class c on the data distribution in an unsupervised situation. However, a maximum dependency approach typically involves computation of multivariate joint probability which is often difficult and inaccurate. One approach to overcome this problem is to realize maximum dependency with *maximizing the relevance*, selection of features with the highest relevance to the target class c. Relevance measures the dependency of the variables with correlation or mutual information. Ding and Peng [15, 37] proposed a feature selection framework based on mutual information. The idea is to maximize the relevance and minimize the redundancy of the feature set. Peng *et al*. [37] proved that maximizing dependency is the same as maximizing the relevance between features and class labels while minimizing the redundancy of dependent features in the feature set. Thus, the noise and outliers in data are reduced and

the redundancy is minimized which induces the classifiers to perform better. This feature selection is applied to the training data as a preprocessing step before learning the object classifiers. As a result of redundancy minimization, the classifier performance is significantly increased and the efficiency of the learning algorithm is boosted by the selection of feature subset [15, 37].

In theory, mutual information is defined in terms of probabilistic density functions $p(x)$, $p(y)$, and $p(x, y)$ of two random variables x and y:

$$I(X;Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \, dxdy \, .$$

Max-Dependency finds the subset of features S with m features $\{x_i\}$ and is defined as:

$$\max D(S,c), D = I(\{x_i, i = 1, ...m\}; c)$$

Max-Relevance selects the features x_i which are required to have the largest mutual information $I(x_i; c)$ with the target class c, reflecting the largest dependency on the target class. An approximation of $D(S, c)$ in the maximum relevance approach is computed as:

$$\max D(S,c), \; D = \frac{1}{|S|} \sum_{x_i \in S} I(x_i; c)$$

However, it is very likely the case that there exist selected features which are dependent and give strong relevance. For this reason while the relevance is maximized, the redundancy of the correlated features should be minimized:

$$\min R(S), \; R = \frac{1}{|S|^2} \sum_{x_i, x_j \in S} I(x_i; x_j)$$

A combination of the two criteria described above, formulates the *minimal-redundancy-maximal-relevance* (mRMR) feature selection method. The final optimization problem is defined as the difference of the two terms:

$$\max \Phi(D, R), \; \Phi = D - R$$

In this method, the subset of the features are iteratively selected such that at each step, one feature is selected depending on all previously selected features with an incremental search:

$$\max_{x_j \in X - S_{m-1}} \left[I(x_j; c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} I(x_j; x_i) \right] .$$

Comparing to the Max-Dependency, the mRMR avoids the estimation of multivariate probability densities $p(x_1, ..., x_m)$ and only calculates the bivariate densities, *i.e.* $p(x_i, x_j)$ and $p(x_i, c)$, that is much easier and applicable to big data. Moreover, this also leads to more efficient feature selection algorithms. On the other hand, mRMR algorithms do not guarantee the global maximization of the problem due to the difficulty in searching the whole space. However, global optimum solution in this approach might lead to overfitting on the data, whereas, the mRMR is a practical way to achieve a superior classification accuracy and it also reduces the complexity of the computation [37].

Part II.

Feature Selection and Learning for Semantic Segmentation

3. Feature Ranking and Object Learning

In this chapter, we introduce our approach for feature selection and learning for semantic segmentation. Our purpose is to find the most distinctive, discriminative and robust feature set for object recognition in the task of semantic segmentation. We employ a feature ranking strategy to compute the importance of each feature in a given set of features and reduce the redundancy in the feature set by analyzing the change of the performance accuracy depending on each feature. Thus, the redundant features are eliminated whilst the accuracy and the efficiency of the system is enhanced.

In the scope of the proposed method, Section 3.1 presents our appearance model, also called *unary pixel potential*, based on object detection/recognition. Section 3.2 shows how the features are ranked using mutual information based feature selection strategy. Furthermore, Section 3.3 compares two machine learning methods, which are used for object learning in the majority of state-of-the-art studies, in order to address the question which one is the most convenient learning strategy in the task of semantic segmentation.

3.1. Appearance Model

Our appearance model is a unary pixel potential $\rho_i(x)$ assigned for every class i given in Equation 3.1 and the potentials are inferred with multi-class object detectors i.e:

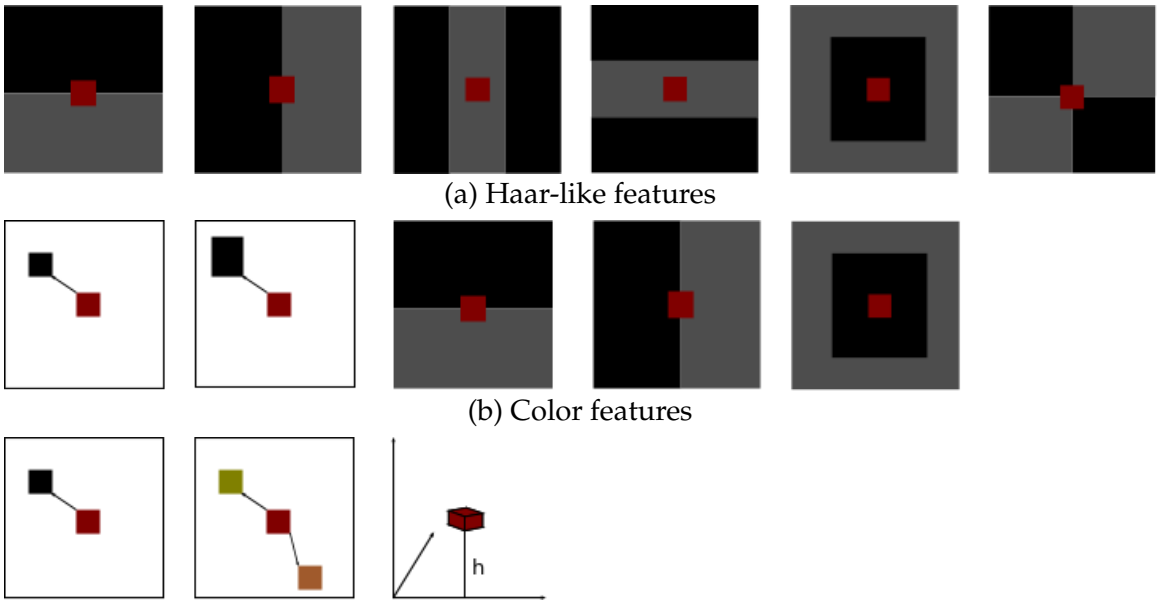
$$\rho_i(x) = -\log \widetilde{P}(i|f,x),\ i = 1,...,n \tag{3.1}$$

where $\widetilde{P}(i|f,x)$ is the probability of class i for a given feature vector f of pixel x. By taking the negative logarithm of probability distributions, we ensure that our potential function is a monotonously decreasing function. Hence, the potential is a convex function, meaning that there exists a global minimizer.

Objects can be categorized with their shape, color and textural properties. To detect an object, one should exploit these features jointly so that each different types of object can be distinguished from others whilst the ones in the same class are kept together. In this study, we employ six types of Haar-like features as shape features, 5 color and 17 texton features (Gaussian filters, Laplacian of Gaussians and a first order derivative of Gaussian filters with various bandwidths) as color-texture features, in addition, we use normalized canonical location features in Equation 3.2 and when applicable depth features as shown in Figure 3.1 . Object classifiers are trained with these 37 joint features to build a robust multi-class object detector.

$$f_l(p) = \left(\frac{x_p}{I_{width}}, \frac{y_p}{I_{height}} \right) \tag{3.2}$$

with (x, y) is the location of the pixel p on an image I.



(a) Haar-like features

(b) Color features

(c) Depth features

Figure 3.1.: **Feature set.** First row, Haar-like features: horizontal/vertical edges and lines, center surround and four square. Second row, color features: relative pixel/patch, horizontal/vertical edge and center surround on color channels. Third row, depth features: relative pixel, relative pixel comparison and height of a pixel.

Filters are applied in a patch surrounding the pixel on the image. We use the CIELab colour space [1] since it is a perceptual uniform color model and is designed to approximate the human vision. This color space is a device-independent model and describes all visible colors to human eye. Luminance (L) channel varies from 0 to 100, indicates the brightness, 'a' and 'b' are color channels. 'a' axis varies between green and red while 'b' axis varies between blue and yellow. Colors are usually numbered from -128 to 127. Figure 3.2 shows the three dimensional Lab space. We sample the image pixels for training. Filter responses are computed on a $\Delta_{ss} \times \Delta_{ss}$ grid on the image to reduce the computational expense [46] during training. For testing, however, filter responses are computed for each pixel.

3.1.1. Shape Features

In 2001, Viola and Jones [49] presented very simple, but robust Haar-like features for face detection. These features are reminiscent of Haar basis functions (Haar wavelets), that were proposed in 1909 by Alfred Haar [24]. A Haar wavelet allows a function defined in an interval to be represented in terms of orthogonal basis functions. The only disadvantage is that Haar wavelets are not continuous and therefore not differentiable. The Haar mother

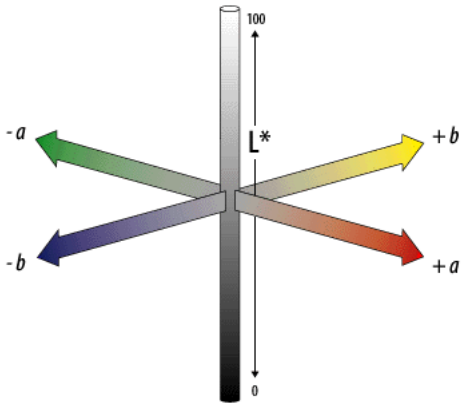


Figure 3.2.: **CIELab color space.** Lab color space has three dimension. 'L' ranges from 0 to 100 and shows the brightness while 'a' and 'b' represent colors and they both range from -128 to 127. Source: [2].

basis and k-th Haar function is defined by:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2}, \\ -1 & \frac{1}{2} < x \leq 1, \\ 0 & \text{else.} \end{cases}$$
$$\psi_{jk}(x) \equiv \psi(2^j x - k)$$

for j a non-negative integer and $0 \leq k \leq 2^j - 1$ as shown in Figure 3.3 .

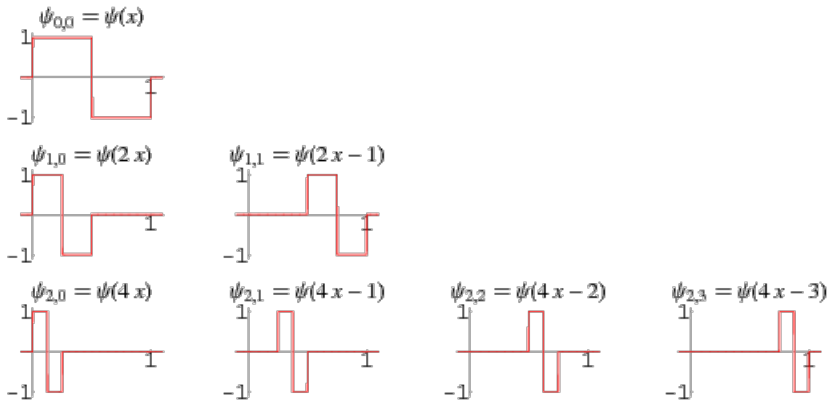


Figure 3.3.: **Haar Wavelets.** Haar mother basis function and haar wavelets [50].

These simple Haar-like features can be rapidly computed using an intermediate representation for the image which is called the integral image [49], also known as summed area table, introduced to computer graphics by Frank Crow in 1984 [13]. The integral image (I) at a pixel position (x, y) contains the sum of all pixel intensities of the pixels above and to

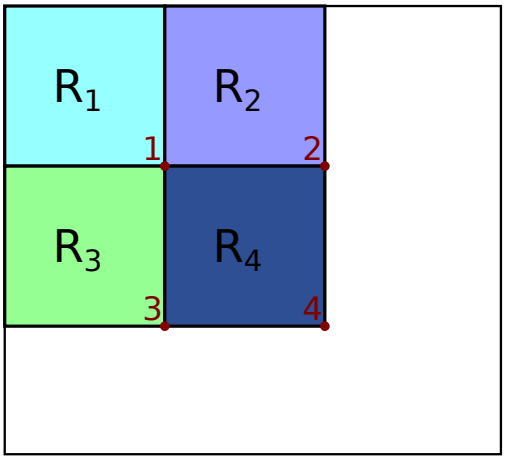


Figure 3.4.: **Haar-like feature calculation via Integral Image.** On the integral image above, the pixel 1 contains the total sum of all pixels in the region R_1, 2 contains R_1+R_2, 3 is equal to R_1+R_3 and 4 is the sum of all regions: R_1+R_2+R_3+R_4. The sum of the pixels in the region R_4 can be computed with four basic operations: $R_4 = (4 + 1) - (2 + 3)$.

the left:

$$I(x, y) = \sum_{x\prime \leq x, y\prime \leq y} image(x\prime, y\prime).$$

With this small modification on the image, sum of the intensities in a rectangle can be computed efficiently only with 4 simple operations illustrated in Figure 3.4 . To compute the feature responses, the sum of the pixels which lie within the white rectangles is subtracted from the sum of the pixels in the grey rectangles.

Haar-like features are very good to capture shapes such as lines, edges, corners and so forth, and can be extensively used for all kinds of objects. The most important advantage of Haar-like features is that the object boundaries can be recognized very sensitively and perfect object separations can be retrieved as a result of segmentation. Another big advantage of these features is that objects can be detected in any scale by scaling the feature window. When doing that, all the rectangle sums must be normalized by the size of rectangle for scale-invariant feature responses. We only employ six Haar-like features, however, one can also use different types and enlarge the feature set. All shape features are computed only on the luminance (L) channel in the Lab color space.

3.1.2. Color and Texture Features

Pixel colors are very basic features and give the local information about the object particles. However, objects usually consist of many different colors and individual pixel colors cannot be used directly as a feature to the object as a whole. Therefore, we extract the color features for each pixel in a surrounding window. We are inspired by Hermans' simple color features [26]: the relative pixel and relative patches, shown in Figure 3.1 are used for color relevancy of the object parts and different objects. Thus, we do not only learn

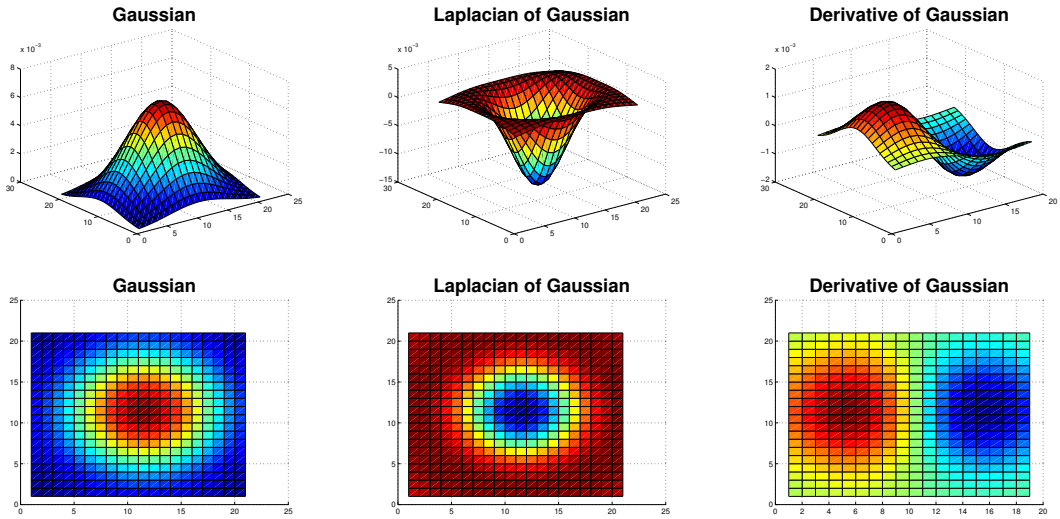


Figure 3.5.: **Texture features.** Gaussian, Laplacian of Gaussian and derivative of Gaussian convolution kernels used as texture features. Rows show 3D and 2D illustration of the kernels, respectively. Warm colors represent the higher function values.

the general color model of the object, but also the color relation between object and its surroundings. This also allows to learn the co-occurrency of the objects and increase the classification accuracy in some sense. In addition to the relative color features, we also use three of the Haar-like features since they are extremely powerful to discover object boundaries. Color features are computed on each channel in the Lab color space.

Colors are not always so discriminative and robust for different textures. For that reason, we exploit the simple, but relatively more expensive basic spatial image filters for texture analysis, that is Gaussian (Equation 3.3), Laplacian of Gaussian (Equation 3.4) and derivative of the Gaussian (Equation 3.5) convolution kernels. The kernels are computed locally around each pixel and the kernel size is determined by its bandwidth ($2\kappa \cdot \sigma + 1, \kappa = 1$). Figure 3.5 shows the 3D and 2D texture kernels used in this application.

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.3}$$

$$LoG(x, y; \sigma) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.4}$$

$$DoG(x, y; \sigma) = \left[\frac{\partial G_\sigma}{\partial x}, \frac{\partial G_\sigma}{\partial y}\right]^T = \left[x e^{-\frac{x^2+y^2}{2\sigma^2}}, y e^{-\frac{x^2+y^2}{2\sigma^2}}\right]^T \tag{3.5}$$

3.1.3. Location Features

More or less, every object occurs at a specific region on the images. For example, sky is always expected to be on the top as car, road and pavements invariably locate at the

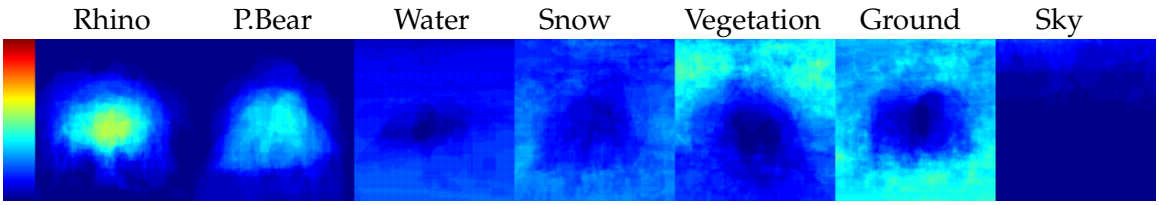


Figure 3.6.: **Location Potential Map.** Location potential map of the 7-class Corel Benchmark [25], learnt from a training data. Pixels are normalized to 100-by-100 canonical size and w_α is chosen 1. Colors show the occurrence likelihood of the objects at each pixel. Warmer colors represent higher probabilities.

bottom. The location information about the objects can be considered in two different ways:

- **Location as a feature:** In the simplest case, locations can be integrated into the object learning strategy in the same way as the other types of features. The location feature is basically computed as a normalized canonical form of a pixel location given in Equation 3.2 . The most important location features will be implicitly selected during training.

- **Location as a potential:** Another way of exploiting the location attribute of the objects is to learn a potential, captures the weak dependence of the class label on the absolute location of the pixel in the image [44]. The location potential(θ_λ) can be adapted into the appearance model as follows:

$$\rho_i(x) = -\log\left(\widetilde{P}(i|f,x)\cdot\theta_\lambda(i,\hat{x})\right).$$

  with $\theta_\lambda(i,\hat{x})$ being the location potential of class i at normalized pixel location $\hat{x}$. The location potential is then learnt from a training data with Equation 3.6 , as used in [44].

$$\theta_\lambda(i,\hat{x}) = \left(\frac{N_{i,\hat{x}} + \alpha_\lambda}{N_{\hat{x}+\alpha_\lambda}}\right)^{w_\lambda} \tag{3.6}$$

  where $N_{i,\hat{x}}$ is the number of pixels of class i at normalized location $\hat{x}$ in the training set, $N_{\hat{x}}$ is the total number of pixels at location $\hat{x}$ and α_λ is a small integer, corresponding to a weak Dirichlet prior on θ_λ. The location potential is raised to the power of w_λ to compensate for overcounting.

Figure 3.6 shows an example location map, learnt for the 7-class Corel dataset of [25]. This map shows the object occurrences on the image. The color gets warmer as the occurrence potential of the object increases. As shown in the figure, there is a high uncertainity on the locations. The appearance model is composed of a scalar multiplication of the object detector confidence with the location confidence and the weak location confidence causes high uncertainty in the model. Therefore, in this study we decided to use the location attribute as a feature.

3.1.4. Depth Features

With the development of cheap RGB-D cameras, the popularity of using these cameras in the applications has increased. RGB-D sensors are very cheap and provide depth information of the scene along with the color images. Therefore, nowadays researchers work on integrating the depth features into the system. By doing that, the accuracy of the system will be increased by only exploiting computationally very low cost and simple features. See Figure 3.7 for a sample gray scaled depth map of a given image, taken from NYU version 1 Depth Benchmark [47]. The Gaussian filter is applied to the all channels (Lab) whilst Laplacian of Gaussian and derivative of Gaussian filters are only applied to the luminance (L) channel.

We use three deep features, also used in [26]:

1- Depth of a relative pixel in a surrounding window, normalized by the furthest depth in the corresponding column.

2- Depth comparison by subtraction of two relative depths as described in [45]. At a given pixel x, the features compute:

$$f_\theta(I,x) = d_I\left(x + \frac{u}{d_I(x)}\right) - d_I\left(x + \frac{v}{d_I(x)}\right),$$

where $d_I(x)$ is the depth at pixel x on an image I, and u and v are randomly chosen offsets in a window ($\theta = (u,v)$). The relative distances (offsets) are normalized by the depth of the current pixel $\left(\frac{1}{d_I(x)}\right)$ to make the features more robust to camera translation.

3- The height of a point in 3D [47]: The height f_h of a point $x = (x_r, x_c)$ with respect to the camera center is computed as follows:

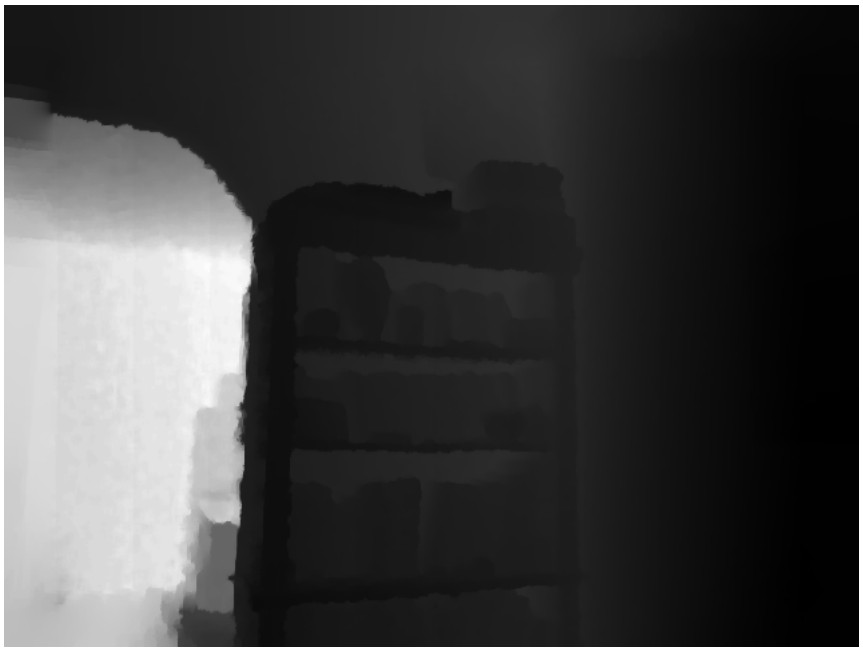
$$f_h(I,x) = -d_I(x) \cdot x_r\,.$$

3.2. Feature Ranking

All the features used for object recognition in this study have been selected only because of their popularity in state-of-the-art works. However, randomly chosen features do not always perform well or combining strong robust features does not guarantee any improvement on the performance. In addition, another drawback of using many features is the huge computation time. As the features are computed for each pixel, the calculation time for the features exponentially grows by adding new ones to the system and the run-time thus drastically increases. Another important issue when using different types of features jointly is that there might be a redundant feature and therefore no longer a feature but a noise, will definitely reduce the classification performance. For all these reasons, we aim at reducing the redundancy and picking the most distinctive and robust feature set for a given benchmark.

Among many other feature selection methods, we use a feature ranking algorithm based on mutual information. In [37] Peng *et al.* present an algorithm which allows ordering

(a) RGB image



(b) Depth map of the image (a)

Figure 3.7.: **Depth map of the scene.** RGB-D cameras provide both RGB color image (a) and corresponding depth map (b) of the scene. Depth values are normalized to gray scale. Brightness increases as the regions get away from the camera.

features by means of maximizing the relevance between features and the corresponding class labels and minimizing the inter-feature redundancy based on mutual information. The Minimum-Redundancy-Maximum-Relevance method (mRMR) maximizes the objective function in Equation 3.8 where $MI(X,Y)$ denotes the mutual information of two continuous random variables:

$$MI(X;Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \ , \tag{3.7}$$

$$\max_{f_j \in X - S_{m-1}} \left[MI(f_j;c) - \frac{1}{m-1} \sum_{x_i \in S_{m-1}} MI(f_j;f_i) \right] . \tag{3.8}$$

The mRMR method (see Section 6.1 for detailed explanation) orders the features incrementally by selecting the one with highest score at each iteration. After ranking the features depending on their relative importance scores, we run the whole system adding the ranked features one by one to the feature set and performing the segmentation on all benchmarks independently. The list of ranked features for each benchmark is given in Appendix A . Further evaluations will be presented in Chapter 6 .

3.3. Object Learning

After feature computation, the most important, class-separator features are selected with one of the machine learning feature selection methods. In this research, we only study commonly used supervised learning algorithms to learn the object classifiers. For our field of application, we consider Gentle AdaBoost [20] and Random Forests [7] to be the most relevant. Classification algorithms generally estimates a class label for a given feature vector. However, since we need to compute the appearance model for segmentation which is consisted of class distributions per pixel, we adapt these two learning algorithms to our purpose as described below.

Random Forests (see Section 2.2.2) are an ensemble of decision trees which can handle large data. The learning process starts with one decision tree and adds another one to the forest if the so called out-of-bag (oob) error is bigger than a certain threshold or the maximum number of trees is not reached. Each tree T_w returns a tree vote $T_w(f)$ for a given feature vector f. The probability distribution for each class i given f and a pixel x is then estimated as follows:

$$\widetilde{P}(i|f,x) = \frac{\sum_{w=1}^{N} [T_w(f) = i]}{N} , i = 1, ..., C \tag{3.9}$$

where N is the number of trees in the forest and C is the total number of classes.

Random Forests exhibit a good detection rate and are very robust against outliers. Another alternative object learning method is the Boosting algorithm (see Section 2.2.1). Boosting is a greedy learning algorithm that linearly combines the weak learners to train a strong classifier. A one-vs-all strategy is used for the training and one strong classifier for each class is trained. This method can either output a class label or a confidence value

(a) Input Image (b) RF unary classifier (c) AB unary classifier

(d) Ground Truth (e) RF Segmentation (f) AB Segmentation

Figure 3.8.: **Random Forest vs. Gentle AdaBoost.** Images (a) and (d) show the input and ground truth images, respectively. Images (b) and (c) illustrate the detection results ($\arg\max_{i,x} \widetilde{P}(i|f,x)$) for Random Forest and Gentle AdaBoost and images (e) and (f) compare the variational segmentation results.

which is sum of the weighted votes of the weak learners. However, we require a probability distribution. Therefore, the distribution is computed using a soft-max transfer function:

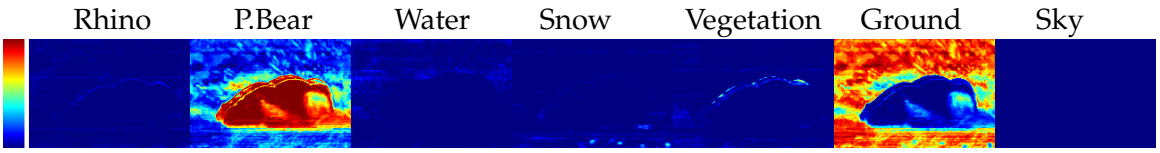
$$\widetilde{P}(i|f,x) = \frac{\exp(H_i(f,x))}{\sum_{i=1}^{C} \exp(H_i(f,x))} \,. \tag{3.10}$$

where $H_i(f,x)$ denotes the confidence score class i computed as follows:

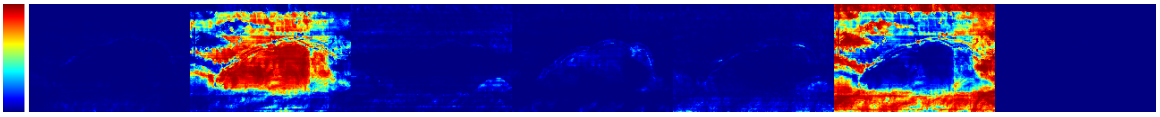
$$H_i(f,x) = \sum_{w=1}^{W} \alpha_w^i(x) \cdot h_w^i(x) \,. \tag{3.11}$$

For our experiments, we set the same learning parameters for all datasets. We trained the Random Forests with a maximum of 50 decision trees, each having a depth of 15 at most. In the Gentle AdaBoost, we perform the training using 100 weak learners per class.

Figure 3.8 shows the experimental results for each unary pixel classifier. Note that the Random Forests give better classification results compared to the Gentle AdaBoost. This is due to the unbalanced training data. Hence in most benchmarks the pixel count for each class is non-uniformly distributed. This leads to a suppression of classes with less pixels. In addition, the soft-max transfer function over-smooths the class potentials. The reason for this is that each classifier is trained independently and that the confidence scores are discrete, real numbers. So the one-vs-all Boosting approach does not provide a good appearance model. In contrast, the Random Forests circumvent this problem by



(a) Random Forest Potential Map



(b) Gentle AdaBoost Potential Map

Figure 3.9.: **Probability maps** provided by Random Forest (a) and AdaBoost (b) classifiers. Warmer color represents higher probability.

penalizing the class errors with the corresponding class weights during the training and perform equally well for each class.

Figure 3.9 shows the class-wise probability maps for both learners. We have more uncertainty (cold colors) in the AdaBoost map (Figure 3.9 b) compared to probabilities generated by the Random Forest (Figure 3.9 a). This problem can be overcome by increasing the number of weak learners. However, this drastically increases the prediction time. Apart from that, prediction time in the Random Forests is faster than the one in the Gentle AdaBoost. For the above reasons, we decided to use Random Forests in the following experiments.

3.3.1. Measurement of the Classification Uncertainty

Random Forests classifier does not return a class label but a distribution over classes in our study. By using this probability distribution $\rho(x)$ for each pixel on an image (see Figure 3.9 a), we measure the quality of object recognition with entropy. Entropy in information theory is a measure of the uncertainty associated with a random variable. This term is also known as Shannon entropy [42], which quantifies the expected value of the information contained in a message, that is a specific realization of a random variable. The entropy $H(C)$ of C random variables (in our case, the total number of classes) in a pixel x is defined as follows:

$$H(C) = -\sum_{i}^{C} \rho_i(x) \cdot \log_C \rho_i(x).$$

Uncertainty of the detection at a pixel x is maximum ($H(C) = 1$) if the Random Forests assign equal probabilities to all classes, and the uncertainty is minimum ($H(C) = 0$) if one of the classes has the highest probability ($p(x) = 1.0$). Figure 3.10 illustrates a diagram that plots the entropy versus probability for the case of 2 random variables.

We expect that a good distribution will retrieve the best segmentation at the end. Therefore, entropy is used to measure the quality of the classifier. Figure 3.11 shows an example

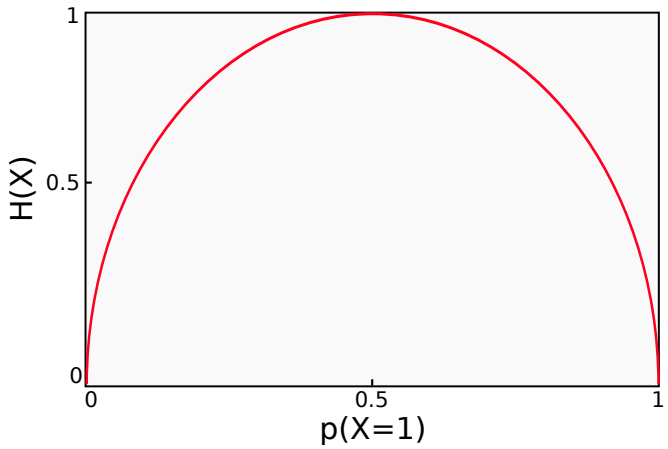
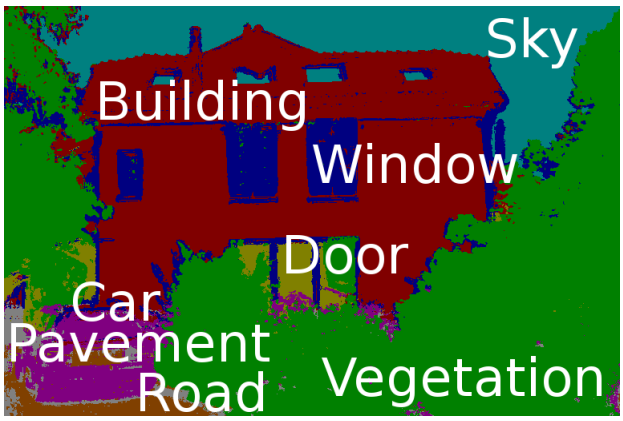


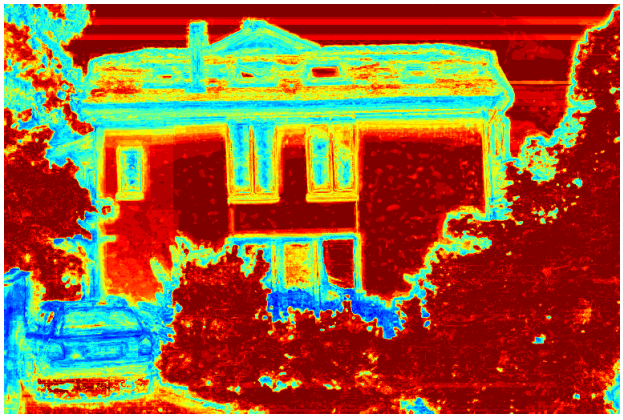
Figure 3.10.: **Entropy** $H(X)$**.** Entropy in the case of two random variables with the probabilities $p(X = 1)$ and $1 - p(X = 1)$. Uncertainty is maximal ($H(X) = 1$) when two random variables have equal probability.

for entropy based uncertainty measurement. The global uncertainty measurements of the unary pixel classifiers for each benchmark are presented in Section 6.2 .

(a) Example Image



(b) Unary pixel classification ($\arg\max_{i,x} \widetilde{P}(i|f,x)$)



(c) Entropy Image

Figure 3.11.: **Detection Uncertainty.** (a) and (b) demonstrate the input image and the unary pixel classification result, respectively. (c) shows the detection accuracy at each pixel computed with entropy. Warmer colors represent ***high certainty***. As shown in (b) and (c), vegetation, sky and inner part of the building have a high certainty in contrast to other regions. Due to the insufficient training samples of cars in the eTrims [27] Benchmark, the pixels, belonging to the car, have the lowest certainty.

4. Variational Optimization and Image Segmentation

This chapter presents the energy functional based on unary pixel potential for multi-class image segmentation and gives a brief description of the convex energy optimization with a first-order primal-dual formulation.

4.1. Variational Image Segmentation

Given an input image $I : \Omega \rightarrow \mathbb{R}^3$ defined on the image domain $\Omega \subset \mathbb{R}^2$, we compute the appearance model $\rho_i(x)$ for all classes i. A subsequent minimization of a regularizing energy which is based on variational multi-labeling [9] allows to obtain consistent pixel labeling and thereby removing label noise in the unary potentials. The energy functional in Equation 4.1 is minimized using a first-order solver [11]. As a regularizer, the total variation of the indicator function u_i of an object of class i is used. This penalizes the boundary length of the object associated with u_i. In order to favor the coincidence of the objects and the image edges, the total variation term is weighted with a non-negative function $g : \Omega \rightarrow \mathbb{R}^+$ given in Equation 2.5 .

$$E(u) = \sum_{i=1}^{n} \int_{\Omega} \rho_i(x) \cdot u_i(x)\, dx + \lambda \sum_{i=1}^{n} \int_{\Omega} g(x) \cdot |\nabla u_i(x)|\, dx \tag{4.1}$$

4.2. First-order Primal-Dual Convex Optimization

In order to optimize the solution, the energy functional must meet the following two conditions:

- **Necessary condition.** If there exist any minimum of the energy functional, the derivative of the functional at the minimum must be equal to zero:

$$\frac{dE(u)}{du} = 0\,.$$

- **Sufficient condition.** To obtain a global minimizer, functional must meet the convexity criteria.

As already discussed in Section 2.1 , our energy functional in Equation 4.1 satisfies the above conditions. However, the minimization problem is now a saddle-point optimization since the total variation is replaced with its primal-dual formulation:

$$|\nabla u| = \sup_{|\xi| \leq 1} \xi \cdot \nabla u\,,$$

where $|\xi| \in \mathbb{R}^2$ is a dual variable and the supremum is attained at $\xi = \frac{\nabla u}{|\nabla u|}$ if $|\nabla u| \neq 0$. This re-formulation allows to re-write the TV as in Equation 2.4 .

Since we replace the total variation with primal-dual formulation, we minimize the energy functional to find a saddle-point. In this study, we exploit an efficient algorithm for minimizing this saddle-point problem, introduced in [11, 38]. We bring our energy function (Equation 4.1) into a primal form as follows:

$$\min_{u \in \mathcal{S}} \sup_{\xi \in \mathcal{K}} \sum_{i=1}^{n} \int_{\Omega} \rho_i(x) \cdot u_i(x)\, dx - \lambda \sum_{i=1}^{n} \int_{\Omega} \operatorname{div} \xi_i \cdot u_i(x)\, dx$$

This problem can now be optimized easily using gradient descent/ascent aspect, also used in [53]. This solving scheme descents in the primal variable and ascents in the dual variable until convergence at the saddle-point. As following the necessary condition, we fix the variables and derive the energy to formulate the update scheme for optimization:

$$\frac{\partial E}{\partial u} = \rho - \operatorname{div} \xi \;\; , \;\; \frac{\partial E}{\partial \xi} = \nabla u$$

In the first step, the primal variable u^0, dual variable ξ^0 and an auxiliary variable v^0 used in the acceleration step, are initialized with 0. Then the algorithm iteratively runs the following steps:

$$\begin{aligned} \xi_i^{n+1} &= \Pi_{\mathcal{K}}(\xi_i^n + \tau \cdot \nabla v_i^n) \\ u_i^{n+1} &= \Pi_{\mathcal{S}}(u_i^n + \eta \cdot (\operatorname{div} \xi_i^{n+1} - \rho_i)) \\ v^{n+1} &= 2u^{n+1} - u^n \end{aligned}$$

$\tau > 0, \eta > 0$ are step sizes and the solution provably converges for sufficiently small step sizes. $\Pi_{\mathcal{K}}$ and $\Pi_{\mathcal{S}}$ are the projections onto the corresponding sets. For the primal variable u, the projection onto the set $\mathcal{S} = \mathrm{BV}(\Omega; [0, 1])$ is calculated by clipping:

$$(\Pi_{\mathcal{S}} u)(x) = \min\{1, \max\{0, u(x)\}\} = \begin{cases} u(x), & \text{if } u(x) \in [0, 1] \\ 1, & \text{if } u(x) > 1 \\ 0, & \text{if } u(x) < 0 \end{cases}$$

For the dual variable ξ, the projection onto the unit disk $\mathcal{K}$ is done as follows:

$$(\Pi_{\mathcal{K}})(x) = \frac{\xi(x)}{\max\{1, |\xi(x)|\}}$$

This update scheme is applied on each pixel independently on the image. Therefore, the optimization process can be accelerated on a GPU.

Additionally, we also optimize the smoothing parameter λ for each benchmark by using binary search as given in Algorithm 2 . Hence, we obtain the best possible multi-region segmentation of the image.

Data: Unary pixel-wise potentials $\rho(x)$
Result: λ_{opt}
$\epsilon \leftarrow 0.005$;
increment $\leftarrow 3$;
in-between $\leftarrow$ false;
$\lambda \leftarrow 1$;
$\lambda_{opt} \leftarrow 0$;
$P_{opt} \leftarrow 0$, $P_{current} \leftarrow 0$, $P_{previous} \leftarrow 0$;
Solve the segmentation with λ and calculate the performance P;
while $P_{current}$ - $P_{previous} \geq \epsilon$ **do**
  if $P_{opt} < P$ **then**
    $P_{opt} \leftarrow P$;
    $\lambda_{opt} \leftarrow \lambda$;
  end
  $P_{previous} \leftarrow P_{current}$;
  $P_{current} \leftarrow P$;
  if $P_{current}$ - $P_{previous} > 0$ **then**
    if in-between **then**
      increment $\leftarrow$ increment / 2;
    end
    $\lambda \leftarrow \lambda$ + increment;
    in-between $\leftarrow$ false;
  else
    increment $\leftarrow$ increment / 2;
    $\lambda \leftarrow \lambda$ - increment;
    in-between $\leftarrow$ true;
  end
  Solve the segmentation with λ;
  Calculate the performance P;
end

Algorithm 2: **Optimization of the smoothing parameter λ.** For each benchmark, λ is optimized with binary search.

Part III.

Experimental Evaluation

5. Benchmarks

In this chapter, we first introduce the benchmarks, used for the scientific evaluation. Our framework has been evaluated on four challenging databases. The benchmarks consist of landscape images, wild-scenes, facade and indoor scenes. This chapter gives a quick look for each benchmark.

5.1. eTrims Benchmark

E-Training for Interpreting Images of Man-Made Scenes(eTrims) [27] dataset is consisted of 60 facade/street images. Each pixel is labeled with a color which corresponds to a class label. Thus, this dataset offers the ground truth annotation on both pixel and region levels. There are in total 8 classes, which are 'Window', 'Vegetation', 'Sky', 'Building', 'Car', 'Road', 'Door' and 'Pavement'. Typical objects in the images exhibit considerable variations in both shape and appearance and hence represent a challenge for the object based image interpretation. This benchmark is one of the most challenging datasets since the shape and appearances of objects are more detailed in the images with high resolution. and generalization of the complex structures on the objects is a difficult task in object classification. Figure 5.1 shows a sample image and the ground truth labeling from eTrims Benchmark. The pixels close to the object boundaries are non-labeled (black) due to the ambiguity on the original images.

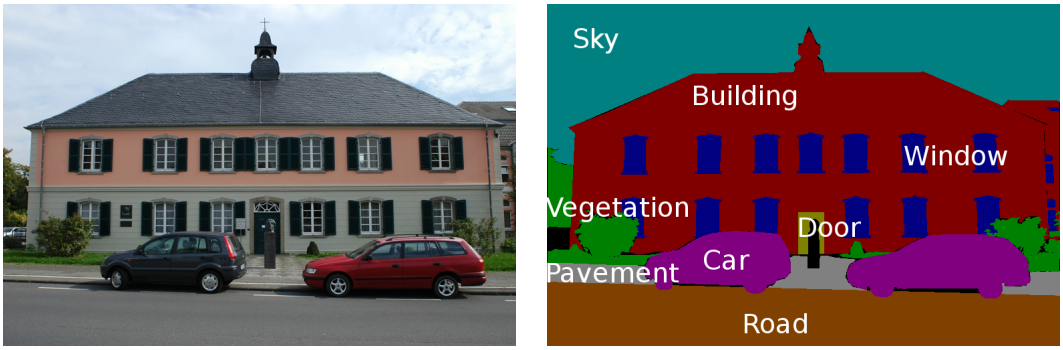


Figure 5.1.: **eTrims Benchmark.** Sample image (left) from eTrims dataset with its associated ground truth labeling (right). Each class is represented with a color. Labels are imposed on the respective color regions.

5.2. NYU version 1 Depth Benchmark

Cheap and sufficiently accurate RGB-Depth cameras increase their popularity in computer vision applications since besides color images, these cameras also provide a depth map of the scene. In our experiments, we used the same RGB-D indoor benchmark introduced in [47] and also used in [26]. This benchmark is comprised of thousand of objects and each image has 640×480 resolution. In order to compare our results with state-of-the-art, we also categorize the objects into 12 core classes as 'Bed', 'Blind', 'Bookshelf', 'Cabinet', 'Ceiling', 'Floor', 'Picture', 'Sofa', 'Table', 'TV', 'Wall', 'Window'. Depths are normalized to grey level (range from 0 to 255), the whiter color represents the further distances. Unknown regions are labeled with white color on the ground truth and these regions are not considered during the training. Figure 5.2 demonstrates a sample image with corresponding depth map and ground truth.

Figure 5.2.: **NYU version 1 Depth Benchmark.** From left to right: original image, grey level depth map and corresponding ground truth labeling.

5.3. Corel and Sowerby Benchmarks

We also applied our method to two natural image datasets, used in [25]. First dataset is a 100 subset of the Corel image database, consisting of African and Arctic wildlife natural scenes. The Corel dataset is labeled into 7 classes: 'Rhino/Hippo', 'Polar Bear', 'Vegetation', 'Sky', 'Water', 'Snow' and 'Ground'. Each image is 180×120 pixels. See Figure 5.3 for a sample image and ground truth labeling.

Second dataset, the Sowerby Image Database of British Aerospace, is a set of color images of out-door scenes (consisting objects near roads in rural and suburban area) and their associated labels. This benchmark is comprised of 104 images with 8 labels: 'Sky', 'Grass', 'Roadline', 'Road', 'Building', 'Sign', 'Car'; each image is 96×64 pixels.

Figure 5.3.: **Corel Benchmark.** An example image (left) from Corel benchmark and its associated ground truth (right) with imposed class names on the corresponding labels.

Figure 5.4.: **Sowerby Benchmark.** An example image (left) from Sowerby benchmark and its associated ground truth (right) with imposed class names on the corresponding labels.

6. Experimental Results

This chapter presents our evaluations for feature selection and segmentation separately. In Section 6.1 , we discuss how many and what kind of features give the best performance for each benchmark independently. Then, in Section 6.2 , we present our segmentation results with the selected subset of features. We compare our experimental results with state-of-the-art methods in terms of quantitative scores and run-time.

We have implemented our approach in C++, NVidia CUDA with OpenCV (Open Computer Vision) [6] Library and DARWIN [23] Machine Learning Framework on Ubuntu 12.04 LTS (Precise Pangolin) Operating System. All experiments have been executed on Intel® Core^{TM} i7-3770 processor equipped with 32 GB RAM and a NVIDIA GeForce GTX480 graphics card.

We tested our framework on four different benchmarks, the 8-class facade dataset introduced in [27], the 7-class Corel and Sowerby datasets of He *et al*. [25] as well as the NYU Depth v1 [47] with 12 core classes. Except for the eTrims Benchmark, we randomly split each dataset into training and test sets by 50%. Table 6.1 shows the patch size and the sampling rate Δ_{ss} used for each benchmark. Due to huge memory consumption on feature ranking with mRMR algorithm, we only use 100 randomly sampled images for NYU Depth v1 Benchmark.

Table 6.1.: **Parameters.** Patch size and sampling rate, used for each benchmark.

Parameter	Sowerby	Corel	eTrims	NYUv1
Patch size	6	10	24	24
Δ_{ss}	3	3	5	5

For the object learning, the same parameters are used for all benchmarks during the training of Random Forests and Gentle AdaBoost classifiers. AdaBoost classifiers are trained with 100 weak learners for each class where each weak classifier is a decision stamp. Random Forests are trained with maximum 50 trees where each has at most 15 depths. Maximum categories parameter is set to 15 (any bigger number than the total class number) so that none of the classes in the benchmark are grouped during training. Splitting is repeated unless at least 10 samples are left in the node. The number of the trees in the forest is determined by the forest accuracy, also known as the out-of-bag error, which is set to 1%. Apart from the learning parameters, due to high unbalanced sample distribution over the classes in benchmarks, we set the class weights for each classes so that during the out-of-bag error calculation, class errors are weighted with the corresponding class weights to balance the class-wise classification accuracy. Although this weighting does not improve the overall performance, it has a significant effect on the class-wise per-

formance scores. Table 6.2 shows the class weights used for each benchmark. The class weights are set to 1 for each class in the NYUv1 Depth Benchmark.

Table 6.2.: **Class Weights.** Class priors to balance the detection performance of each class in all benchmarks.

(a) Sowerby

Labels	Sky	Grass	Roadline	Road	Building	Sign	Car
Weight	1.0	1.0	**1.5**	1.0	1.0	**1.5**	**1.5**

(b) Corel

Labels	Rhino	Polar Bear	Water	Snow	Ground	Vegetation	Sky
Weight	1.0	1.0	**1.05**	1.0	1.0	1.0	**1.1**

(c) eTrims

Labels	Building	Vegetation	Door	Window	Car	Sky	Pavement	Road
Weight	1.0	1.0	**1.15**	**0.65**	1.0	1.0	1.0	1.0

6.1. Feature Selection

We rank the features using the mRMR method on the training images and select the best subset of features, by incrementally adding the ranked features one by one to our feature set and comparing the results for each feature set. Figure 6.1 illustrates the overall performance scores vs. the total number of used features for all benchmarks. The features which are actually increasing the performance as well as the number of features leading to a redundant-free feature set can be easily found in this illustration.

The eTrims Benchmark is composed of high resolution images, thus object textures have significant importance for detection. Similar to Fröhlich *et al.* [21] we split the dataset by a ratio of 60%/40% for training/testing. As shown in Figure 6.1 , there is a jump at the 24th feature which is a first order derivative of a Gaussian on the '*L*' channel. Figure 6.1 indicates that the remaining features are redundant. Therefore, we only use the first 24 features for the eTrims Benchmark.

For a reasonable overall performance on the Sowerby Benchmark according to Figure 6.1 , already the first three features would be enough: the relative color feature on the '*a*' channel, the Haar horizontal edge feature on the '*b*' channel and the relative patch feature on the '*L*' channel. Instead of using a larger set of features, this simple set can be used to obtain similar performance. However, one can observe another jump in the performance at the 21st feature. Additional features beyond the first 21 do not improve the performance, but would increase the computational cost. Hence, for our experiments we used the first 21 features. Most of the dropped features are texture features. In contrast to the eTrims Benchmark, the information gain of the textural features is relatively small because of the low resolution of the images in the Sowerby dataset.

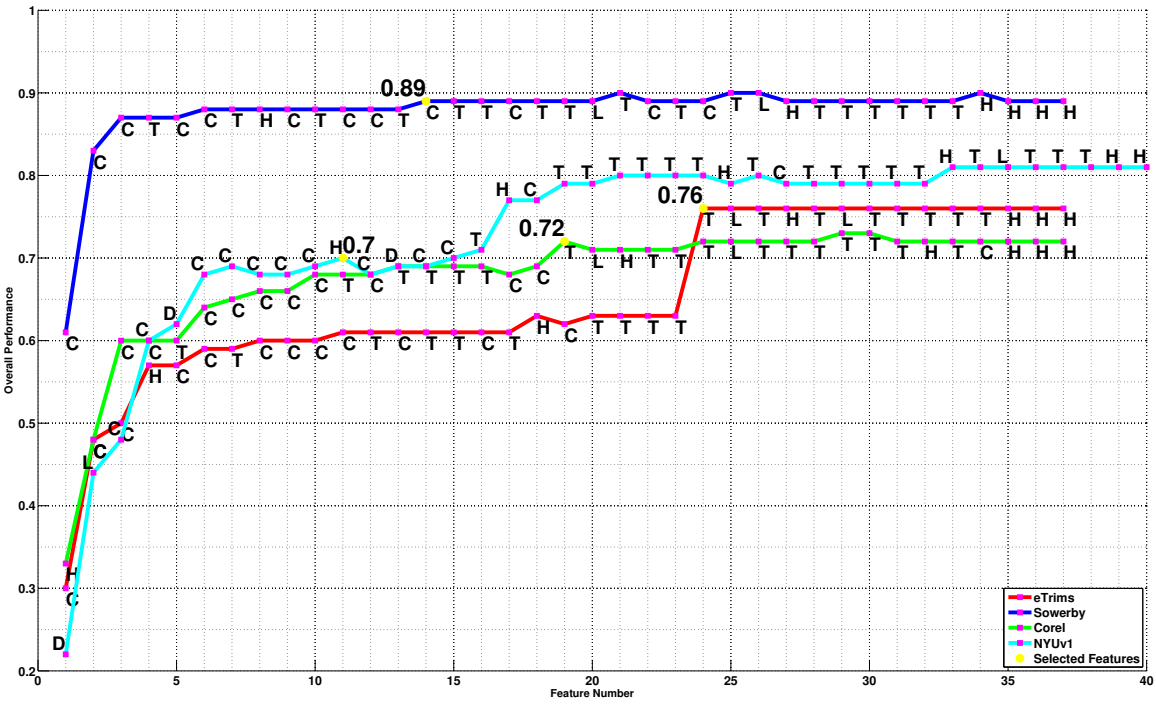


Figure 6.1.: **Overall performance vs. Features.** Features are added to the feature set one by one. Pink dots denote the type of the feature (H: Haar, C: Color, T: Texture and L: Location, D: Depth) added at current step. Yellow circles show how many features are selected for the benchmark.

6.2. Segmentation Results

In this section, we present our qualitative and quantitative segmentation results in addition to the performance/run-time comparisons between ours and state-of-the-art studies.

We, first, present the global entropy scores computed during the evaluation. Entropy is computed at each pixel and averaged as global entropy all over the test set in each random split. Scores present the final mean global entropy scores for each benchmark given in Table 6.3 .

Table 6.3.: **Global Detection Entropy.** Unary pixel classifier global entropy scores for all benchmarks.

	$\widetilde{H}_{global}(C)$
eTrims	0.31
NYUv1	0.38
Corel	0.42
Sowerby	0.23

In Table 6.4 we compare our eTrims benchmark segmentation results with the work of

Fröhlich *et al.* [21]. The algorithm is executed on ten random splits of each benchmark and the scores shown in Tables 6.4, 6.5, 6.6, 6.7 are the average scores of these random splits. The scores give the percentage of the correctly labeled pixels for each class, *overall* denotes the percentage of the correctly labeled pixels in the whole dataset whilst the *average* gives the mean of the class-wise scores.

Table 6.4.: **eTrims Benchmark.** Comparison of the segmentation performance and the efficiency for eTrims

Labels	Building	Vegetation	Door	Window	Car	Sky	Pavement	Road	Overall	Average	Time
Fröhlich *et al.* [21]	63.5	**90.1**	**95.4**	**71.9**	77.4	73.2	**71.1**	**69.9**	**76.1**	**72.3**	≈ 17s
Ours	**75.0**	84.0	64.0	57.0	**78.0**	**96.0**	56.0	69.0	76.0	72.0	≈ **5s**

Our performance scores shown in Table 6.4 are as good as Fröhlich *et al.* [21]'s and we have significant improvement in the evaluation time for the eTrims Benchmark.

Table 6.5 shows that both training and testing speeds are significantly reduced compared to Shotton *et al.* [46] on the Sowerby and Corel benchmarks.

Table 6.5.: **Sowerby and Corel Benchmark.** Segmentation/detection and efficiency comparison with Shotton *et al.*

	Overall		**Speed(Train/Test)**	
	Sowerby	Corel	Sowerby	Corel
Shotton *et al.* – Full CRF model [46]	88.6	**74.6**	5h/10s	12h/30s
Ours – Segmentation	**90.0**	69.4	**13s/96ms**	**45s/280ms**
Shotton *et al.* – Unary classifier only	85.6	**68.4**	-	-
Ours – Unary classifier only	**87.1**	65.0	-	-

Table 6.6.: **Sowerby and Corel Benchmark Average Class Scores**. Average class-wise segmentation performance of ten random splits.

(a) Sowerby

Labels	Sky	Grass	Roadline	Road	Sign	Building	Car
Average Score	93.7	87.5	7.35	96.0	57.0	0.042	0.274

(b) Corel

Labels	Rhino	Polar Bear	Water	Snow	Vegetation	Ground	Sky
Average Score	83.1	65.5	75.1	60.6	71.5	65.6	53.0

Furthermore, we compare our segmentation with the approaches of Ladický *et al.* [29]

and Hermans *et al*. [26] on the NYUv1 Benchmark. Here we use only the best 11 ranked features including the depth features to compete with Hermans *et al*. [26]. Table 6.7 demonstrates that we can significantly improve the detection rate for several classes. We obtain the best score compared to Ladický *et al*. [28] and Hermans *et al*. [26] for more than half of the classes. Furthermore, we obtain the best average performance. We cannot win a direct runtime-competition with the algorithm of Hermans *et al*. [26]. Nevertheless, our system is open to be paralellized for faster evaluation. Our goal is to keep things simple and to give the opportunity for a quick reimplementation without the need of a high-end hardware. Compared to the multi-threaded CPU implementation of Ladický *et al*. [29] our runtime is around 20 times faster. This shows the outstanding performance of our method in the detection accuracy and runtime.

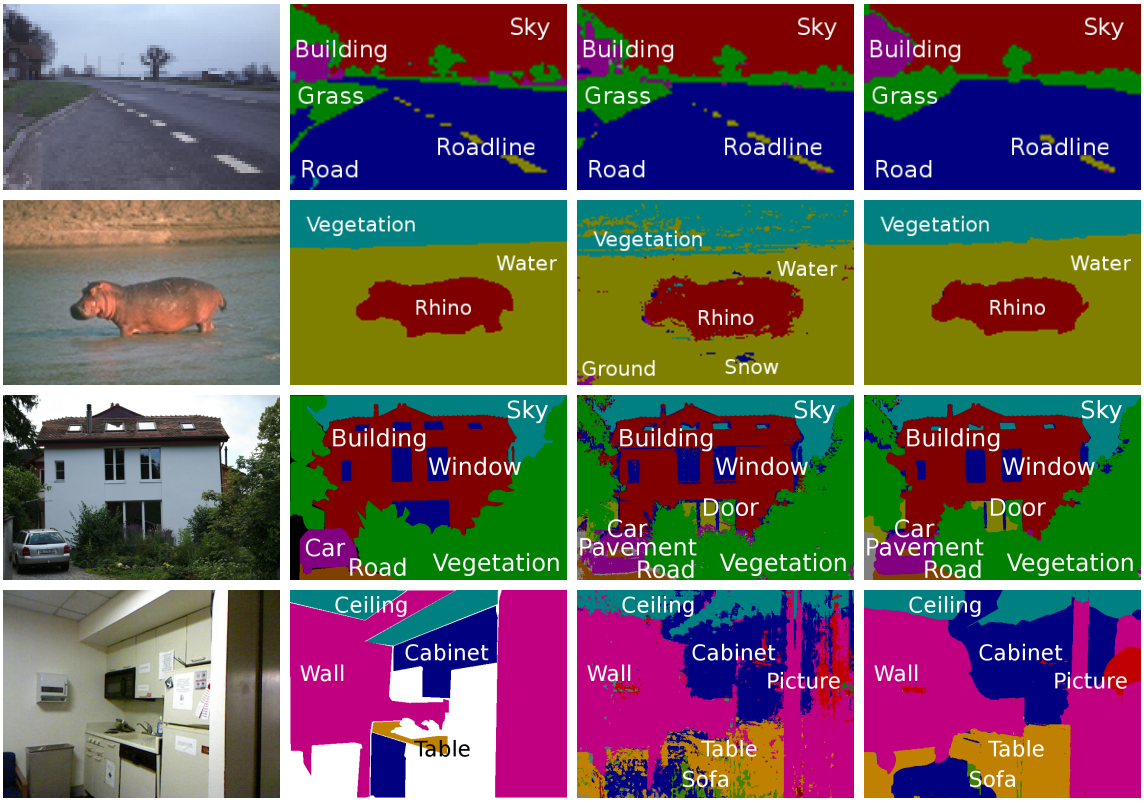
Table 6.7.: **NYUv1 Benchmark.** Comparison of the segmentation/detection performance and the efficiency for NYUv1

Labels	Bed	Blind	Booksh.	Cabinet	Ceiling	Floor	Picture	Sofa	Table	TV	Wall	Window	Overall	Average	Time
Ladický *et al*. [29]	14	3	57	34	59	76	49	34	47	56	**90**	11	68	44	≈3m
Hermans *et al*. [26]	58	57	67	58	**93**	**88**	57	67	46	82	78	17	**71**	64	≈ 1s
Ours	**82**	**59**	**82**	**75**	75	78	**79**	**71**	**81**	**85**	68	**49**	70	**74**	≈ 8s
Hermans *et al*. [26](unary)	51	42	48	54	88	87	62	50	40	73	70	19	65	57	0.2s
Ours (unary)	76	57	72	70	71	75	70	66	74	74	63	44	64	68	≈ 4s

Figure 6.2 illustrates the qualitative results in all benchmarks. In the segmentation result of eTrims in the third row of Figure 6.2 d, we can see that the windows on the roof are labeled as sky because of the reflection and also in the upper part of image we can find a region labeled as pavement although there is none in the ground truth.

Comprehensive experiments on various benchmarks have proven that the segmentation performance is heavily dependent on the pixel-wise object classification where the quality of the feature set is significantly effective. Combining robust and discriminative features does not always yield better performance since the redundancy of the interrelated features causes high amount of misclassification in the object detection, hence decreases the efficiency of the system in terms of computational expense and the accuracy of segmentation. Therefore, the features were ranked based on their mutual information and the most distinctive feature set for each benchmark was determined with performance experiments on the test sets. These selected subset of features improved the segmentation accuracy whilst decreased the computational cost along with the complexity of the classification problem.

It is also shown that Haar-like features are the most robust and the most discriminative features compared to the other simple features and also to the computationally expensive features such as textons. However, in contrast to the conventional object detection studies, these simple, yet powerful features yield better performance if the features are extracted on color channels. Moreover, Haar-like features are indeed robust to capture the object shapes which induces the object detectors to preserve the object boundaries by finding the best separation of the segments accurately.



(a) Original Image (b) Ground Truth (c) Unary Pixel Classification (d) Segmentation

Figure 6.2.: **Qualitative Results.** Rows correspond to the Sowerby, Corel, eTrims and NYU Depth v1 benchmarks respectively. First two columns show input and ground truth images while the third and forth columns show unary pixel classification and segmentation result respectively.

Part IV.

Summary and Conclusion

7. Summary

Semantic segmentation is a joint task of object detection/recognition and segmentation in the field of computer vision. In this work, to address the problem of semantic segmentation, an appearance model based on object detectors is constructed and an energy functional based on this model is optimized with a convex optimization approach.

In this study, we construct our appearance model with Random Forest classifier, trained with various types of features and optimize our energy functional with the total variation, boundary-length minimizer, based variational optimization technique. Although the optimization tries to find the best segmentation on a given appearance model, the robustness and the accuracy of the segmentation is purely dependent on the quality of the appearance model. The major challenge in constructing this sort of appearance model is to accurately detect the objects while preserving the boundaries in-between. In order to segment all disjoint sets on an image, the current studies use a mixture of all types of features so that the various objects, which carry different structural and textural properties, can be easily distinguished from each other. Despite the fact that different types of features are actually designed for capturing different object properties such as color, texture or shape, using these features jointly is not always a good solution because of the high relevance between features yields high redundancy in the feature set. As a result of redundant features, the noise in the training data increases and the classification methods struggle with building robust object classifiers. Moreover, the computational cost is exponentially increased by adding new features to the feature set since the object detection is performed at a pixel level and herewith the segmentation speed gets slower.

By exploiting redundancies in feature sets we have shown that the computational cost for learning and testing in the task of semantic segmentation can be significantly reduced. To this end we adapted a feature selection algorithm to the purpose of semantic segmentation which reduced the redundancy within the feature set while preserving accuracy. In many cases, we are able to outperform state-of-the-art results in terms of performance and in addition to runtime due to a sophisticated feature set selection with reduced dimension.

7.1. Discussion

In this study, we have proven that Random Forests perform better than Gentle AdaBoost in terms of runtime as well as classification accuracy in the task of multi-class object detection for semantic segmentation. Apart from that, the lists of ranked features indicates that Haar-like features are the most robust and the most discriminative features compared to the others. Nevertheless, we should also note that Haar-like rectangles are capable of understanding the shapes robustly and produce much more distinctive features on color channels. Additionally, experiments have shown that the object boundaries can be accurately detected with these simple features.

As every object has its own characteristic, every benchmark has also certain differences and therefore the number and type of features, which should be used, vary from one to another. Simple objects and low resolution images on which the objects have no detail, could be easily segmented by using only a couple of features while high resolution images on which the shape and the texture of the objects can be observed well, could be segmented accurately by using relatively more features. Therefore, the used features should be examined and the best subset of the features should be selected in order to detect the objects and to find the consistent segmentation in addition to reduce the computational expense and runtime.

8. Conclusion

Semantic segmentation is one of the most challenging research topics in the field of computer vision due to the huge diversity of the objects and the limitations of current object classification methods. Although the current multi-class classification algorithms perform well on variant image benchmarks, because of the high relevance between features and the redundancy in the feature set, these algorithms suffer from huge amount of the noise in the training data. As a result of noisy training, the accuracy of the classifiers drastically decreases and the segmentation fails. Therefore, the visual object detection features, which basically represent the characteristic of the objects, should be chosen depending on their importance so that the classification method generates a robust object classifier with a noise-free training set. Hence, the quality of the object detectors increases and the algorithm generates better segmentation outputs with a lower cost of computation.

8.1. Future Work

The feature selection by reducing the redundancy in the feature set with *minimum-redundancy-maximum-relevance* method improves the performance and reasonably reduces the computation time. However, this approach requires a huge memory to run and it is not possible to perform the feature selection on big benchmarks. As a future work, either one should optimize the method to perform on a huge data or another approach should be replaced.

Due to the insufficient memory issue, we could only test several types of features and our framework only allows to select the features for each benchmark independently. In future studies, one could generalize the framework to select the features considering all benchmarks together. Moreover, once the system is eligible, the number of objects in the set of classes can be increased to solve more complex and higher dimensional object classification problems.

Appendix

A. List of Ranked Features

The full list of ranked features for each benchmark is given in this chapter. For eTrims, Corel and Sowerby benchmarks, 37 features, composed of 6 Haar-like features, 12 Color features, 2 location and 17 texture features, are used. For NYU version 1 Depth benchmark, additionally 3 depth features are used. Tables A.2, A.3, A.4, A.5 demonstrates the ranked features with their associated type, color channel which the feature computed on, the patch size and the bandwidth parameters used for the feature. For better visualization, rows in the tables are colored with the corresponding color of features given in Table A.1 .

Table A.1.: **Feature Colors.** Each feature is associated with a color.

Haar	Color	Texture	Location	Depth

Table A.2.: **eTrims Benchmark**. The full list of ranked features.

Rank	Feature	Type	Lab Channel	Patch Size	Bandwidth
1	Color	Relative Patch	b	24	-
2	Color	Vertical Edge	a	24	-
3	Color	Relative Pixel	L	24	-
4	Haar	Vertical Line	L	24	-
5	Color	Relative Patch	a	24	-
6	Color	Center Surround	b	24	-
7	Texture	Gaussian	L	-	1
8	Color	Vertical Edge	b	24	-
9	Color	Relative Patch	L	24	-
10	Color	Horizontal Edge	a	24	-
11	Color	Relative Pixel	b	24	-
12	Texture	Laplacian of Gaussian	L	-	1
13	Color	Relative Pixel	a	24	-
14	Texture	Gaussian	L	-	2
15	Texture	Derivative of Gaussian	L	-	2
16	Color	Center Surround	a	24	-
17	Texture	Gaussian	L	-	4
18	Haar	Center Surround	L	24	-
19	Color	Horizontal Edge	b	24	-
20	Texture	Derivative of Gaussian	L	-	2
21	Texture	Derivative of Gaussian	L	-	4
22	Texture	Laplacian of Gaussian	L	-	2
23	Texture	Laplacian of Gaussian	L	-	4
24	Texture	Derivative of Gaussian	L	-	4
25	Location	Y Direction	-	-	-
26	Texture	Gaussian	a	-	1
27	Haar	Vertical Edge	L	24	-
28	Texture	Gaussian	a	-	2
29	Location	X Direction	-	-	-
30	Texture	Gaussian	b	-	4
31	Texture	Gaussian	b	-	1
32	Texture	Gaussian	a	-	4
33	Texture	Gaussian	b	-	2
34	Texture	Laplacian of Gaussian	L	-	8
35	Haar	Four Square	L	24	-
36	Haar	Horizontal Edge	L	24	-
37	Haar	Horizontal Line	L	24	-

Table A.3.: **NYUv1 Benchmark**. The full list of ranked features.

Rank	Feature	Type	Lab Channel	Patch Size	Bandwidth
1	Depth	Point Height	Depth Map	24	-
2	Location	Y Direction	-	-	-
3	Color	Relative Patch	a	24	-
4	Color	Center Surround	a	24	-
5	Depth	Relative Pixel Depth	Depth Map	24	-
6	Color	Horizontal Edge	a	24	-
7	Color	Relative Patch	b	24	-
8	Color	Vertical Edge	a	24	-
9	Color	Relative Pixel	a	24	-
10	Color	Center Surround	b	24	-
11	Haar	Vertical Line	L	24	-
12	Color	Horizontal Edge	b	24	-
13	Depth	Relative P. Comparison	Depth Map	24	-
14	Color	Vertical Edge	b	24	-
15	Color	Relative Pixel	b	24	-
16	Texture	Laplacian of Gaussian	L	-	1
17	Haar	Horizontal Line	L	24	-
18	Color	Relative Patch	L	24	-
19	Texture	Laplacian of Gaussian	L	-	8
20	Texture	Laplacian of Gaussian	L	-	2
21	Texture	Laplacian of Gaussian	L	-	4
22	Texture	Derivative of Gaussian	L	-	4
23	Texture	Derivative of Gaussian	L	-	4
24	Texture	Gaussian	a	-	1
25	Haar	Horizontal Edge	L	24	-
26	Texture	Gaussian	b	-	2
27	Color	Relative Pixel	L	24	-
28	Texture	Gaussian	b	-	4
29	Texture	Gaussian	a	-	2
30	Texture	Derivative of Gaussian	L	-	2
31	Texture	Gaussian	a	-	4
32	Texture	Gaussian	b	-	1
33	Haar	Vertical Edge	L	24	-
34	Texture	Derivative of Gaussian	L	-	2
35	Location	X Direction	-	-	-
36	Texture	Gaussian	L	-	4
37	Texture	Gaussian	L	-	1
38	Texture	Gaussian	L	-	2
39	Haar	Four Square	L	24	-
40	Haar	Center Surround	L	24	-

Table A.4.: **Corel Benchmark**. The full list of ranked features.

Rank	Feature	Type	Lab Channel	Patch Size	Bandwidth
1	Haar	Horizontal Line	L	10	-
2	Color	Vertical Edge	a	10	-
3	Color	Relative Patch	a	10	-
4	Color	Center Surround	b	10	-
5	Texture	Derivative of Gaussian	L	-	4
6	Color	Center Surround	a	10	-
7	Color	Relative Patch	L	10	-
8	Color	Horizontal Edge	a	10	-
9	Color	Vertical Edge	b	10	-
10	Color	Relative Patch	b	10	-
11	Texture	Derivative of Gaussian	L	-	2
12	Color	Relative Pixel	a	10	-
13	Texture	Gaussian	b	-	4
14	Texture	Derivative of Gaussian	L	-	4
15	Texture	Laplacian of Gaussian	L	-	2
16	Texture	Laplacian of Gaussian	L	-	4
17	Color	Relative Pixel	b	10	-
18	Color	Relative Pixel	L	10	-
19	Texture	Derivative of Gaussian	L	-	2
20	Location	Y Direction	-	-	-
21	Haar	Vertical Line	L	10	-
22	Texture	Gaussian	a	-	4
23	Texture	Gaussian	a	-	1
24	Texture	Gaussian	b	-	1
25	Location	X Direction	-	-	-
26	Texture	Gaussian	L	-	1
27	Texture	Gaussian	L	-	4
28	Texture	Gaussian	b	-	2
29	Texture	Gaussian	L	-	2
30	Texture	Gaussian	a	-	2
31	Texture	Laplacian of Gaussian	L	-	8
32	Haar	Center Surround	L	10	-
33	Texture	Laplacian of Gaussian	L	-	1
34	Color	Horizontal Edge	b	10	-
35	Haar	Four Square	L	10	-
36	Haar	Vertical Edge	L	10	-
37	Haar	Horizontal Edge	L	10	-

Table A.5.: **Sowerby Benchmark**. The full list of ranked features.

Rank	Feature	Type	Lab Channel	Patch Size	Bandwidth
1	Color	Relative Pixel	a	6	-
2	Color	Horizontal Edge	b	6	-
3	Color	Relative Patch	L	6	-
4	Texture	Laplacian of Gaussian	L	-	1
5	Color	Center Surround	a	6	-
6	Color	Center Surround	b	6	-
7	Texture	Derivative of Gaussian	L	-	4
8	Haar	Center Surround	L	6	-
9	Color	Relative Patch	a	6	-
10	Texture	Derivative of Gaussian	L	-	2
11	Color	Relative Pixel	L	6	-
12	Color	Vertical Edge	a	6	-
13	Texture	Gaussian	b	-	4
14	Color	Horizontal Edge	a	6	-
15	Texture	Derivative of Gaussian	L	-	4
16	Texture	Laplacian of Gaussian	L	-	2
17	Color	Relative Pixel	b	6	-
18	Texture	Laplacian of Gaussian	L	-	4
19	Texture	Derivative of Gaussian	L	-	2
20	Location	Y Direction	-	-	-
21	Texture	Gaussian	a	-	4
22	Color	Vertical Edge	b	6	-
23	Texture	Gaussian	a	-	1
24	Color	Relative Patch	b	6	-
25	Texture	Gaussian	b	-	1
26	Location	X Direction	-	-	-
27	Haar	Vertical Line	L	6	-
28	Texture	Gaussian	L	-	1
29	Texture	Gaussian	L	-	4
30	Texture	Gaussian	b	-	2
31	Texture	Gaussian	L	-	2
32	Texture	Gaussian	a	-	2
33	Texture	Laplacian of Gaussian	L	-	8
34	Haar	Four Square	L	6	-
35	Haar	Vertical Edge	L	6	-
36	Haar	Horizontal Edge	L	6	-
37	Haar	Horizontal Line	L	6	-

Bibliography

[1] Cie: 015:2004. Technical report, Colorimetry, 3rd edn. Commission Internationale de l'Eclairage, 2004. 22

[2] Adobe. "Technical Guides". Color Models: CIELAB,. `http://dba.med.sc.edu/price/irf/Adobe_tg/models/cielab.html`. 23

[3] P. Arbelaez, B. Hariharan, Chunhui Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3378–3385, June 2012. 3

[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. 13

[5] Wassim Bouachir, Atousa Torabi, Guillaume-Alexandre Bilodeau, and Pascal Blais. A bag of words approach for semantic segmentation of monitored scenes. *CoRR*, abs/1305.3189, 2013. 3

[6] G. Bradski. *Dr. Dobb's Journal of Software Tools*, 2000. 45

[7] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. 13, 15, 17, 29

[8] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. 15

[9] A. Chambolle, D. Cremers, and T. Pock. A convex approach for computing minimal partitions. Tech. rep. TR-2008-05, University of Bonn, 2008. 12, 35

[10] A. Chambolle, D. Cremers, and T. Pock. A convex approach to minimal partitions. *SIAM Journal on Imaging Sciences*, 5(4):1113–1158, 2012. 12

[11] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. Technical Report R.I. 685, Ecole Polytechnique, Palaiseau Cedex, France, 2010. 35, 36

[12] T.M. Cover. The best two independent measurements are not the two best. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-4(1):116–117, Jan 1974. 4

[13] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 207–212, New York, NY, USA, 1984. ACM. 23

[14] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005. 3, 4

[15] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 523–528, Aug 2003. 17, 18

[16] PedroF. Felzenszwalb and DanielP. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 7

[17] WendellH. Fleming and Raymond Rishel. An integral formula for total gradient variation. *Archiv der Mathematik*, 11(1):218–222, 1960. 11

[18] Yoav Freund and Robert 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. 13

[19] Yoav Freund and Robert E. Schapire. A short introduction to boosting. In *In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406. Morgan Kaufmann, 1999. 13, 15

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998. 13, 14, 29

[21] Björn Fröhlich, Erik Rodner, and Joachim Denzler. Semantic segmentation with millions of features: Integrating multiple cues in a combined random forest approach. In *Asian Conference on Computer Vision (ACCV)*, pages 218–231, 2012. 4, 46, 48

[22] Enrico Giusti. *Minimal Surfaces and Functions of Bounded Variation*. Monographs in Mathematics. Birkhäuser Boston, 1984. 9

[23] Stephen Gould. DARWIN: A framework for machine learning and computer vision research and development. *Journal of Machine Learning Research*, 13:3533–3537, Dec 2012. 45

[24] A. Haar. *Zur Theorie der orthogonalen Funktionensysteme*. 1909. 22

[25] Xuming He, R.S. Zemel, and M.A. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695–II–702 Vol.2, June 2004. 26, 42, 45

[26] Alexander Hermans, Georgios Floros, and Bastian Leibe. Dense 3d semantic mapping of indoor scenes from rgb-d images. In *International Conference on Robotics and Automation*, 2014. 3, 4, 24, 27, 42, 49

[27] F. Korč and W. Förstner. eTRIMS Image Database for interpreting images of man-made scenes. Technical Report TR-IGG-P-2009-01, April 2009. 33, 41, 45

[28] L. Ladický, Russell C., Kohli P., and Torr P. Graph cut based inference with co-occurrence statistics. In *Proceedings of ECCV*, 2010. 3, 49

[29] L. Ladický, C. Russell, P. Kohli, and P. H S Torr. Associative hierarchical crfs for object class image segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 739–746, Sept 2009. 3, 4, 48, 49

[30] Ľubor Ladický, Paul Sturgess, Karteek Alahari, Chris Russell, and PhilipH.S. Torr. What, where and how many? combining object detectors and crfs. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, volume 6314 of *Lecture Notes in Computer Science*, pages 424–437. Springer Berlin Heidelberg, 2010. 4

[31] Jan Lellmann, Jörg Kappes, Jing Yuan, Florian Becker, and Christoph Schnörr. Convex multi-class image labeling by simplex-constrained total variation. In Xue-Cheng Tai, Knut Mørken, Marius Lysaker, and Knut-Andreas Lie, editors, *Scale Space and Variational Methods in Computer Vision*, volume 5567 of *Lecture Notes in Computer Science*, pages 150–162. Springer Berlin Heidelberg, 2009. 12

[32] D. Lowe. Object recognition from local scale-invariant features. Corfu, Greece, September 1999. 3, 4

[33] Philippe Magiera and Carl Löndahl. "ROF Denoising Algorithm",. `http://www.mathworks.de/matlabcentral/fileexchange/22410-rof-denoising-algorithm`, 2008. 10

[34] Bjoern H. Menze, Michael B. Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A. Hamprecht. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. *BMC Bioinformatics*, 10(1), 2009. 16

[35] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. 16

[36] David Mumford and Jayant Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics*, 42(5):577–685, 1989. 7, 8

[37] H. Peng, Fulmi Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, Aug 2005. 4, 5, 17, 18, 27

[38] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the piecewise smooth mumford-shah functional. In *IEEE International Conference on Computer Vision (ICCV)*, Kyoto, Japan, 2009. 36

[39] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958. 13

[40] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1–4):259 – 268, 1992. 8

[41] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *Machine Learning*, pages 80–91, 1999. 15

[42] Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. 31

[43] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. 3

[44] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for mulit-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 1–15, 2006. 3, 4, 26

[45] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew W. Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Commun. ACM*, 56(1):116–124, 2013. 27

[46] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009. 3, 22, 48

[47] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011. 27, 42, 45

[48] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–762–II–769 Vol.2, June 2004. 4

[49] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. 3, 4, 22, 23

[50] Eric. W. Weisstein. "Haar Function." From *MathWorld*– A Wolfram Web Resource. `http://mathworld.wolfram.com/HaarFunction.html`. 23

[51] Christopher Zach, David Gallup, Jan-Michael Frahm, and Marc Niethammer. Fast global labeling for real-time stereo using multiple plane sweeps. In *VMV*, pages 243–252, 2008. 12

[52] Weiwei Zhang, Jian Sun, and Xiaoou Tang. Cat head detection - how to effectively exploit shape and texture features. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Computer Vision – ECCV 2008*, volume 5305 of *Lecture Notes in Computer Science*, pages 802–816. Springer Berlin Heidelberg, 2008. 4

[53] M. Zhu and T. Chan. An efficient primal-dual hybrid gradient algorithm for total variation image restoration. *UCLA CAM Report*, pages 08–34, 2008. 36