DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

BAMOT: Bundle Adjustment for Multiple-Object Tracking - a Stereo-based Approach

Anselm Coogan

DEPARTMENT OF INFORMATICS

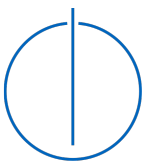
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Robotics, Cognition, Intelligence

BAMOT: Bundle Adjustment for Multiple-Object Tracking - a Stereo-based Approach

BAMOT: Bundle Adjustment für Multi-Objekt Tracking mit Stereo-Kameras

Author:	Anselm Coogan
Supervisor:	Prof. Dr. Laura Leal-Taixé
Advisor:	Dr. Aljoša Ošep and Nikolaus Demmel
Submission Date:	15.03.2021



I confirm that this Master's Thesis in Robotics, Cognition, Intelligence is my own work and I have documented all sources and material used.

Zurich, 15.03.2021 Anselm Coogan

Acknowledgments

First and foremost, I would like to thank Aljoša for his advice, guidance, and support while writing this thesis. In particular, his vast expertise in all things concerning multi-object tracking was invaluable. My gratitude goes towards Niko, whose SLAM proficiency immensely helped when implementing object-level Bundle Adjustment. Last but not least, I thank my parents for their continuous support during my studies.

Abstract

This thesis addresses the problem of 3D Multiple Object Tracking for RGB-based systems. More specifically, we propose a method that performs sparse feature-based object-level Bundle Adjustment for accurate object track localization. Using the Convolutional Neural Network 2D object detector/tracker TrackR-CNN [117], we introduce a procedure for stereo object detections and improve TrackR-CNN's trained association ability. We achieve superior association via a multi-stage association pipeline that combines appearance and 3D localization similarity. Additionally, we leverage a priori knowledge of object shapes and dynamics for both association and keeping a sparse outlier-free point cloud representation of objects.

We evaluate our proposal on the KITTI [38] tracking dataset via the traditional CLEAR [10] and the recently introduced HOTA [77] metrics. However, as the official KITTI tracking benchmark only includes 2D Multiple Object Tracking evaluation, and the extended 3D evaluation from [119] only supports CLEAR via 3D Intersection-over-Union, we implement a customized tracking ability assessment. The evaluation introduces a normalized 3D Generalized Intersection-over-Union [96] detection similarity score to the official HOTA evaluation scripts [1]. We compare our performance to the Light Detection And Ranging-based AB3DMOT [119] for which 3D tracking results are readily available and demonstrate promising results, especially w.r.t. association and rigid, moving objects. Furthermore, we show the impact of various features of our system on overall performance. The code for Bundle Adjustment for Multi-Object Tracking and performance evaluation is publicly available [2].

[1] `https://github.com/JonathonLuiten/TrackEval`
[2] `https://github.com/AnselmC/bamot`

Contents

1 Introduction & Contribution

1.1 Introduction

Kick-started by the first autonomous vehicle race - the 2004 DARPA Grand Challenge - research and development in autonomous navigation for robots has seen substantial advancement in the previous two decades. Research in this area encompasses a multitude of domains. It includes accurately localizing a robot and mapping its environment; understanding the scenery semantically and detecting and tracking other participants in the robot's surroundings is another crucial dimension - even more so in safety-critical situations such as self-driving cars. Path planning, trajectory prediction of detected objects, and motion control are additional essential tasks for a truly autonomous system. Solutions to the subproblems mentioned above exist for varying degrees of scene complexity, underlying sensor sets, and computational resources. However, fundamentally, autonomous navigation remains an open research problem.

Localization and map creation of the (static) environment (known as Simultaneous Localization and Mapping (SLAM)) is among the most well-understood and explored dimensions of the challenge at hand. The main formulations based on Sound Navigation And Ranging (Sonar) and Light Detection And Ranging (LiDAR) sensors were introduced in the 1990s and then further analyzed and developed in the previous two decades [17]. More recently, SLAM systems based on cheaper RGB-camera setups have received increasing attention and several solutions have been proposed, both for dense disparity maps [34] and sparse feature-based maps [33, 85]. While these procedures work well for non-changing environments, they exhibit difficulties in highly dynamic situations where the static world assumption no longer holds. This shortcoming has incited research that detects moving parts in the sensor data and removes these before running "traditional" SLAM culminating in more robust systems [126, 11, 101, 70]. This function was substantially aided with the advent of highly performant Deep Learning (DL) object detectors, e.g., [93, 47, 95, 121].

Detecting objects and associating detections across frames, termed Multiple Object Tracking (MOT), is another subtask for any self-driving system (but also critical for other applications, e.g., surveillance). While considerable research exists on tracking in 2D (i.e. in images), e.g., [63, 105, 104, 71], the requirement of 3D object localization for

autonomous navigation scenarios has recently spawned numerous publications for this task: although many depend on accurate but expensive LiDAR, e.g. [119, 8, 125, 23], there also exist implementations that only utilize image data, e.g., [88, 76, 69, 49, 7, 123, 31].

Of these camera-based 3D multi-object trackers, [69, 49, 7, 123, 31, 76] combine 3D MOT and SLAM. Such a synthesis comes naturally as both systems can benefit from each other: detecting objects enables more robust SLAM in dynamic scenes as well as a higher-level understanding of the environment. On the other hand, accurate robot odometry and static-map knowledge from SLAM aids precise localization of objects in 3D space.

Due to the nascency of 3D MOT, most of the systems mentioned above do not evaluate their performance w.r.t. 3D MOT. Instead, they evaluate their 3D Multiple Object Detection (MOD) capabilities or revert to assessing their 2D MOT ability. Although the work by Weng et al. [119] proposes an extension to 3D MOT evaluation for the widely-used 2D MOT CLEAR [10] metrics, the authors use 3D Intersection-over-Union (IoU) for calculating detection similarity. This measure, however, does not capture 3D tracking performance well as it does not consider imprecise detections. While this is less of a problem for LiDAR systems, detection accuracy for visual-trackers significantly decreases with camera-object distance, and disregarding all of these detections does not reflect the actual tracking performance. Moreover, Luiten et al. [77] recently presented an argument against current MOT metrics (including CLEAR) and proposed an alternative: Higher Order Tracking Accuracy (HOTA).

1.2 Contribution

This thesis focuses on stereo-based 3D MOT that is easily integrable with a static SLAM system. For 2D object detection we employ TrackR-CNN [117]. Utilizing the tried-and-tested Bundle Adjustment (BA) optimization technique abundant in SLAM, we implement object-level BA based on sparse features to perform 3D object localization. We also exploit a priori knowledge of object shapes and dynamics to aggressively cull object landmarks, thus removing outliers and keeping a sparse object representation. We achieve robust association of detections across frames via a multi-step association pipeline: first, we compute similarity based on a novel combined appearance and 3D localization score. For remaining detections, we corroborate the world-space motion of appearance-based association from TrackR-CNN with available 3D information. In a final association stage, only the Euclidean distance encodes the similarity of detections and existing tracks.

We evaluate our system's 3D MOT performance on the KITTI [38] tracking dataset

employing a normalized Generalized Intersection-over-Union (GIoU) similarity measure and demonstrating results w.r.t. the CLEAR metrics as well as the newly introduced HOTA. We compare our results to the LiDAR-based AB3DMOT [119] and demonstrate promising results for rigid objects in motion. We also investigate the impact of various high-level features of our system on overall performance. Additionally, we discuss the capabilities and shortcomings of Bundle Adjustment for Multi-Object Tracking (BAMOT) qualitatively.

2 Notations & Conventions

In this work, we denote scalars as lowercase characters, e.g. a, vectors as bold lowercase characters, e.g. $\mathbf{a}^T = \begin{pmatrix} a_x & a_y & a_z \end{pmatrix}$, and matrices as bold uppercase characters, e.g. $\mathbf{A} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$. The identity matrix of dimension n is expressed as $\mathbf{I}^n$, the 3D zero-vector as $\mathbf{0}$, and the n-dimensional zero matrix (all entries zero) as $\mathbf{0}^n$. $|\mathbf{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$ denotes the length of a vector $\mathbf{a}$. Additionally, $\mathbf{A}^T$ ($\mathbf{a}^T$) is the transpose of a matrix (vector) $\mathbf{A}$ ($\mathbf{a}$) and similarly the inverse of a matrix $\mathbf{A}$ is $\mathbf{A}^{-1}$. A matrix's trace (i.e. the sum of the elements of its main diagonal) is written as $\mathrm{tr}(\mathbf{A})$. The cross-product of two vectors is denoted by "$\times$" and the dot product by "$\cdot$": $\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{pmatrix}$ and $\mathbf{a} \cdot \mathbf{b} = \mathbf{a}^T \mathbf{b}$, respectively. Furthermore, a bold uppercase character with indices in the following manner express transformations (see Chapter 3) from coordinate system Y to coordinate system X: ${}^X_Y\mathbf{T}$. If a time component t is involved, this expression is enhanced: ${}^X_Y\mathbf{T}_t$. We denote a point/vector $\mathbf{p}$ given in a specific coordinate system X as ${}^X\mathbf{p}$. Further, non-recurring additions to this notation are explained as needed. All coordinate systems are right-handed (see Fig. 2.1).

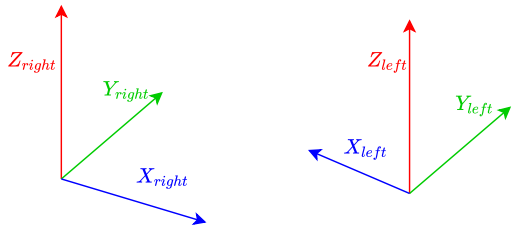


Figure 2.1: Two possible ways to represent coordinate systems. In SLAM one commonly uses the right-handed coordinate system which we adopt for this thesis.

3 Theoretical Background

3.1 Multiple Object Detection and Tracking

Detecting objects in the sensory output of imaging systems (e.g., RGB images or LiDAR scans) has been a cornerstone to Computer Vision (CV) research since its advent in the mid-sixties of the 20th century [90]. It is fundamental to most robotics-related applications today (e.g., surveillance systems or, possibly less dystopian, autonomous driving). Typically, an object detection consists either of a bounding box (2D for RGB images, 3D for RGB-D or LiDAR) around said object or, in more precise methods, of pixel- or voxel-wise (for 2D and 3D input, respectively) detections. If an object detector can detect more than one type of object, each detection is usually accompanied by a class prediction (e.g., cat, dog, or car). Usually, a detector should also be capable of detecting more than one object in its input domain. Furthermore, frequently merely detecting objects does not suffice: being able to associate detections across multiple input frames (e.g., a video stream) is paramount to understanding a dynamic environment. This linking of single detections over time is known as object tracking.

3.1.1 Classical object detection

Some of the earliest research on object detection relates to finding humans (or their faces) in images such as the "Eigenface" approach introduced in [107] and based on Principal Component Analysis (PCA), a Histogram Of Oriented Gradients (HOG) method patented in [80], or a ML-based system proposed in [116] known as the Viola-Jones detector. All of these approaches extract lower-level image features (such as edges, points, or statistical attributes) and then employ varying techniques (often ML-based) to recognize objects based on specific features or clusters thereof. Fig. 3.1 exemplifies representations of feature spaces for some of these procedures. Although these techniques work sufficiently well in specific scenarios (e.g., face detection based on Viola-Jones), they are limited to a single (or very few) object classes, output imprecise bounding boxes for detections, and often require extensive feature-engineering, i.e., how and which low-level characteristics are extracted from the input and fed to a machine-learning algorithm for classification.

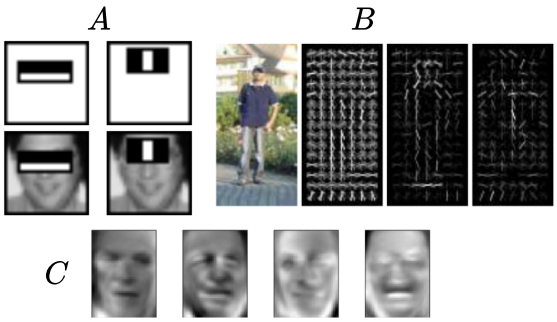


Figure 3.1: *Top Left (A)*: Two example filters/features learned by a Viola-Jones detector for face detection [116]. *Top Right (B, from left to right)*: An example input image followed by its Histogram-of-Oriented-Gradients representation and two results of learned output weighting of the HOGs for people detection [26]. *Bottom (C)*: Various examples of eigenfaces used for face detection and recognition (taken from `https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html`).

3.1.2 Deep object detection

Another early object detection task was recognizing letters, a technique known as Optical Character Recognition (OCR) [103]. These feature-engineering heavy systems were revolutionized by the first automatic gradient-based image-filter Deep Neural Network (DNN) [64], introducing a class of Neural Networkss (NNs) known today as Convolutional Neural Networks (CNNs). CNN are multi-layered NN that utilize learned shift-invariant filters to understand locally-correlated input data (e.g. images) for a plethora of tasks including, first and foremost, object detection [58], [95], [93]). Detailed overviews, explanations and discussions on DL and, more specifically, CNN, can be found in [13], [48], and [40] for example. The above-cited object detection networks produce bounding boxes and their respective class predictions. However, pixel-level object detection networks such as [47] compute so-called segmentation masks and class predictions per mask. Fig. 3.2 shows a comparison of bounding box vs. segmentation mask detections.

This pixel-wise precision of object detections is instrumental and even necessary when objects occlude each other or do not fill-out a rectangle very well (as is mostly the case). 3D object detectors that work on LiDAR or RGB-D data as input have also been proposed [59], [120], [91], [106]. The standard way to describe a 3D object detection is by the seven Degree-of-Freedoms (DOFs) Object Oriented Bounding Box (OOBB). This OOBB consists of a Euclidean location (x, y, z), the object's dimensions (height, width, length), and its rotation around gravity (more commonly referred to as its yaw). Note

Figure 3.2: Rectangular bounding boxes are a poor fit for capturing a person's geometry - segmentation masks do this far better. Additionally, segmentation masks are also superior when objects overlap.

that much like 2D bounding boxes in the image domain, this type of description does not necessarily describe an object well, i.e., it assumes that objects travel along a fixed plane, only rotate around a single axis, and nicely fill out a rectangular box.

3.1.3 Object Tracking

As previously mentioned, detecting objects is frequently accompanied by tracking object instances over time. There exist two fundamental approaches to tracking: offline and online tracking. Offline tracking tries to associate detections after-the-fact: given an entire sequence of i images and n_j detections for a given image, these methods try to find the globally optimal tracks. One possibility to solving such global optimization is to pose the tracking problem as a Minimum-cost Flow Problem (MCFP) [3] where detections constitute nodes and the flow along a collection of nodes makes up a track [71, 104, 63].

On the other hand, online tracking associates detections in a streaming fashion, i.e., by processing images and the detections in these images sequentially. Hence at every step, detections need to be matched with existing object tracks; if the system doesn't match a given detection, it creates a new track. One generally formulates the matching problem as a bipartite graph problem where one collection of nodes are the detections $U_{detections}$ and the other are the existing object tracks V_{tracks} (see Fig. 3.3).

The edges E between nodes $i \in U_{detections}$ and $j \in V_{tracks}$ of these two sets are weighted by some similarity score $S(i, j)$ and the optimization goal is to maximize the total similarity by matching all nodes from the respective sets.

Hungarian Algorithm

The Hungarian Algorithm [60] solves the optimization problem of maximizing the sum of the weights of a matching M between $U_{detections}$ and V_{tracks} for the graph $G =$

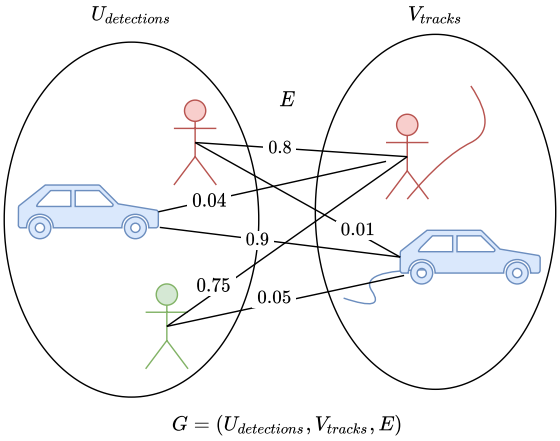


Figure 3.3: The association problem stated as a bipartite graph where one set of nodes are object detections, the other set are object tracks, and the weights of the edges connecting nodes between these two sets are the similarities (or dissimilarities) between each track-detection combination.

$(U_{detections}, V_{tracks}, E)$. If minimizing similarity is the goal, one can trivially accomplish this by multipliying similarities by -1 and then maximizing the total inverse similarity.

The algorithm introduces the notion of labels for nodes (denoted by $l(.)$) where $l(i) + l(j) \leq w(i,j)$ and from this the equality graph $G_l = (U_{detections}, V_{tracks}, E_l)$ where E_l only includes edges whose weights are equal to the sum of the labels of the nodes, i.e. $w_l(i,j) = l(i) + l(j)$. The central theorem of the Hungarian Algorithm is that any *perfect* matching M_l in this equality graph maximizes the edge weights in the original graph, and hence is an optimal matching M. A perfect matching is a matching where all nodes from $U_{detections}$ are matched to a single node in V_{tracks} and vice versa, i.e., a 1-to-1 mapping. Of course, the number of elements of the node sets may be unequal. Introducing "dummy" elements to the smaller set with similarity scores of zero to all other elements in the other set can resolve this discrepancy. To initialize the algorithm, the labels for nodes in $U_{detections}$ are set to the maximum weight of outgoing edges, whereas the labels for nodes in V_{tracks} start at 0. This labeling trivially fulfills $l(i) + l(j) \geq w(i,j)$ and the algorithm now constructs E_l by adding edges for which $l(i) + l(j) = w(i,j)$ holds. Additionally, M_l initializes to the empty set.

The algorithm now improves M_l via two procedures: augmenting paths and improving labels. First, the algorithm searches for an augmented path, e.g., via Breadth-First Search (BFS). An augmented path is a path that starts at a node $u \in U_{detections}$ and ends at a node $v \in V_{tracks}$ both of which are not connected to an edge in M_l. Furthermore, the edges in E_l connecting these nodes must alternately already belong to M_l and not

be part of it. The nodes visited by a path in V_{tracks} are cached as $S \subset V_{tracks}$ and likewise the nodes visited in $U_{detections}$ are tracked in $T \subset U_{detections}$.

If an augmented path is found, the procedure adds the edges of the path that were not in M_l to M_l. It also removes the edges that were previously in M_l from the matching. On the other hand, if an augmented path is not found, the algorithm optimizes the labels $l(.)$ in such a way that existing edges of M_l remain in E_l, i.e. $l(i) + l(j) = w(i, j)$. It then adds new edges to E_l that fulfill this property. The algorithm avhieves this by computing a δ based on visited nodes $s \in S$ and unvisited nodes $y \in U_{detections} \setminus T$ as follows:

$$\delta = \min_{s,y}\{l(s) + l(y) - w(s, y)\} \tag{3.1}$$

From this one constructs a new labeling:

$$l'(n) = \begin{cases} l(n) - \delta \text{ iff } n \in S \\ l(n) + \delta \text{ iff } n \in T \\ l(n) \text{ else} \end{cases} \tag{3.2}$$

This two-step process of finding augmented paths and improving the node labelling repeats until all nodes of both sets are matched.

In scenarios such as object tracking, one doesn't require a one-to-one matching. In fact in many situations such a mapping is not desirable, e.g. when an old object leaves the view and a new object enters it. Thus, a similarity threshold can help: a match between two nodes $u \in U_{detections}, v \in V_{tracks}$ is only valid if the similarity scores between them are greater than an application-specific threshold, i.e. $S(u, v) \geq \alpha$. An alternative implementation for solving the Hungarian Algorithm uses adjacency matrices.

Some examples of online 2D trackers include [117, 12]; instances of object trackers operating in 3D space $\mathbb{E}^3$ (i.e. via OOBB) are [119, 88].

3.2 3D Geometry

3.2.1 Transformations in 3D space

An important part of any robotics-related application is describing rotations and translations of rigid bodies in 3D. These transformations must preserve the following properties [15] w.r.t. two vectors or points $\mathbf{a}, \mathbf{b} \in \mathbb{R}^3$:

1. their distance to each other, i.e. $|\mathbf{a} - \mathbf{b}| = |q(\mathbf{a}) - q(\mathbf{b})|$
2. the orientation between them, i.e. $q(\mathbf{a} \times \mathbf{a}) = q(\mathbf{a}) \times q(\mathbf{b})$

Given a fixed world coordinate frame, a three DOFs translation can be trivially expressed by a 3D vector $\mathbf{t}^T = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}$.

A rotation in 3D Euclidean space also has three DOFs but has multiple representations [79]:

1. as a unit-length rotation axis $\mathbf{u}$ and an angle θ around that axis (axis-angle representation)
2. as a unit quaternion $\mathbf{q} = q_r + q_i\mathbf{i} + q_j\mathbf{j} + q_k\mathbf{k}$ with $|\mathbf{q}| = 1$ and $q_i \in \mathbb{R}$ where i, j, k are imaginary and special mathematical properties apply
3. as $\mathbf{R}$, a 3×3 orthogonal(i.e. $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ [87]) rotation matrix with $\det(\mathbf{R}) = 1$ for right-handed systems (and $\det(\mathbf{R}) = -1$ for left-handed systems)
4. as a combination of three successively applied rotations around three orthogonal axes (Euler angles).

In the following, we will deal with rotations in their axis-angle representation when expressing incremental rotations. Otherwise, we will use their matrix representations as this enables computationally efficient matrix-vector multiplication. Computing the corresponding rotation matrix from a rotation axis $\mathbf{u}$ and an angle θ can be achieved with Rodrigues' formula [87]:

$$\mathbf{R} = \mathbf{I} + (1 - \cos(\theta))\mathbf{u}^\wedge\mathbf{u}^\wedge + \sin(\theta)\mathbf{u}^\wedge \tag{3.3}$$

$\mathbf{u}^\wedge$ in Eq. (3.3) is the skew-symmetric matrix representation of a vector: $\mathbf{u}^\wedge = \begin{pmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{pmatrix}$. This skew-symmetric representation is useful as it linearizes the cross-product between two vectors: $\mathbf{a} \times \mathbf{b} = \mathbf{a}^\wedge\mathbf{b}$. The property giving this representation its name is skew-symmetry: $(\mathbf{u}^\wedge)^T = -\mathbf{u}^\wedge$.

Inversely, one can calculate the angle and rotation axis from a rotation matrix as follows:

1. compute the angle: $\theta = \arccos(\frac{\mathrm{tr}(\mathbf{R})-1}{2})$
2. solve the system of equations given by the following relation: $\mathbf{R}\mathbf{u} = \mathbf{u}$

The combination of rotation and translation is also a rigid body transformation $\mathbf{T}$ and describes arbitrary transformations in $\mathbb{R}^3$. Such a transformation can be expressed as the rotation of a vector/point followed by its translation: $\tilde{\mathbf{a}} = \mathbf{R}\mathbf{a} + \mathbf{t}$. However, ideally, a transformation should be expressed by a single operation.

Fig. 3.4 gives an example of a rigid body transformation from an initial frame A to the final frame B.

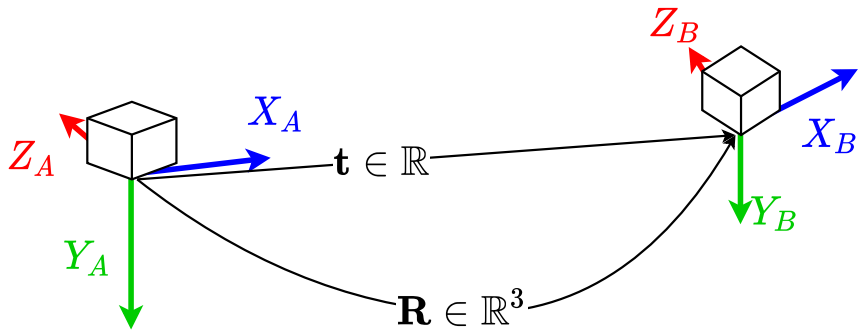


Figure 3.4: A rigid body undergoes a transformation in Euclidean space $\mathbb{E}^3$ from frame A to frame B.

3.2.2 Homogeneous Coordinates

Homogeneous coordinates are a standard tool for transforming points/vectors in a linearized fashion via matrix multiplication [87]. Combining the rotational and translational part of a Euclidean transformation into a single 4-by-4 matrix $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$ and using the homogeneous representation of Euclidean vectors allows for such computationally desired multiplication (see Eq. (3.4)) forgoing the need of rotating a vector first and then translating it as described previously. In homogeneous coordinates a scaling factor w is added as the fourth dimension s.t. the vector $\mathbf{a}^T = \begin{pmatrix} x & y & z \end{pmatrix}$ is represented as $\tilde{\mathbf{a}}^T = \begin{pmatrix} x \cdot w & y \cdot w & z \cdot w & w \end{pmatrix}$. To transform a homogeneous coordinate to Euclidean space, one merely divides the homogeneous vector by w and discards the fourth dimension (which equals one after division). Inversely, to transform a Euclidean vector to its homogeneous counterpart, one can simply multiply all elements by a scaling factor $w \neq 0$ and add this factor as the fourth dimension to the vector. For simplicity, one usually chooses w to be one. Observe that this results in the fact that homogeneous vectors have infinite representations, whereas their Euclidean complements are unique.

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{pmatrix} \mathbf{v} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}\mathbf{v} + \mathbf{t} \\ 1 \end{pmatrix} \tag{3.4}$$

3.2.3 Groups

Group theory is the study of so-called mathematical groups and plays a vital role in algebra, the natural sciences, and engineering. A group is the combination of some set $\mathbf{G}$ and a binary operator $\cdot$ for which the following four axioms must hold [87]:

- *Closure*: combining two elements $a, b \in \mathbf{G}$ with the binary operator produces another element which is part of the group, i.e. $a \cdot b = c \in \mathbf{G}$
- *Associativity*: $\forall a, b, c \in \mathbf{G}$ it must hold that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
- *Identity*: there must exist an identity element $e \in \mathbf{G}$ s.t. $a \cdot e = a \; \forall a \in \mathbf{G}$
- *Invertibility*: for every element a there must be a (unique) inverse element $a^{-1} \in \mathbf{G}$ such that $a \cdot a^{-1} = e$.

A basic example of a group is the set of integers $\mathbb{Z}$ under addition, and typical examples of non-Groups are the set of integers under multiplication or the set of natural numbers $\mathbb{N}$ combined with addition [62].

Lie Groups and Lie Algebras

So-called Lie groups are a subset of groups. These types of groups have the additional unique property that they locally (i.e., close to an element's identity) approximate Euclidean space and, hence, allow one to perform Euclidean calculus such as differentiation. Differentiation implicitly then allows parameterizing an element by the variable being differentiated by, e.g., time.

This informal definition is more correctly described as Lie groups exhibiting the attribute of being a smoothly differentiable manifold, or topological space [1].

One can investigate this local resemblance to Euclidean space via its tangent space. The tangent space close to the identity of a Lie group is called the Lie algebra $\mathfrak{g}$. Its vector space $\mathbb{V}$ (i.e., the space making up its elements) over some field $\mathbb{F}$ (i.e., the field where its elements are defined) and a binary operator, its Lie bracket, $[,]$, characterize a Lie algebra.

This Lie bracket is an alternating bilinear mapping that satisfies the Jacoby identity given by Eq. (3.5) [43].

$$\begin{aligned} &\forall \mathbf{X}, \mathbf{Y}, \mathbf{Z} \in V, \\ &[\mathbf{X}, [\mathbf{Y}, \mathbf{Z}]] + [\mathbf{Z}, [\mathbf{X}, \mathbf{Y}]] + [\mathbf{Y}, [\mathbf{Z}, \mathbf{X}]] = \mathbf{0} \end{aligned} \tag{3.5}$$

An alternating bilinear mapping is defined as a binary operator for which the following assumptions must hold [43]:

- *Closure*: operations are closed under composition, i.e. $\forall \mathbf{X}, \mathbf{Y} \in V \rightarrow [\mathbf{X}, \mathbf{Y}] \in V$.
- *Bilinearity*: the Lie algebra must satisfy the bilinear form as given in Eq. (3.6)

- *Alternativity*: application of the operator on the element and itself must result in the origin vector, i.e. $\forall \mathbf{X} \in V \rightarrow [\mathbf{X}, \mathbf{X}] = \mathbf{0}$.

$$\begin{aligned} &\forall \mathbf{X}, \mathbf{Y}, \mathbf{Z} \in V \wedge u, w \in \mathbf{F} \\ &[u\mathbf{X} + b\mathbf{Y}, \mathbf{Z}] = \\ &u[\mathbf{X}, \mathbf{Z}] + w[\mathbf{Y}, \mathbf{Z}] \\ &\wedge \\ &[\mathbf{Z}, u\mathbf{X} + w\mathbf{Y}] = \\ &= u[\mathbf{Z}, \mathbf{X}] + w[\mathbf{Z}, \mathbf{Y}] \end{aligned} \tag{3.6}$$

These axioms of an alternating bilinear map can be more succinctly denoted with the syntax given in Eq. (3.7).

$$\mathfrak{g} \times \mathfrak{g} \rightarrow \mathfrak{g}, (\mathbf{X}, \mathbf{Y}) \mapsto [\mathbf{X}, \mathbf{Y}] \tag{3.7}$$

A more rigorous and in-depth definition can be found in [43] or [35].

3.2.4 The Special Orthogonal Group SO(3) and its Lie Algebra $\mathfrak{so}(3)$

N-dimensional rotations, or more specifically for the case of 3D space, three-dimensional rotations form a Lie group called the Special Orthogonal Group SO(n) and SO(3), respectively. The Special Orthogonal Group are all elements of $n \times n$ matrices with orthogonal columns and (in right-handed coordinate systems) a determinant of 1 (as previously alluded to in Section 3.2.1). Eq. (3.8) gives the formal definition.

$$SO(n) = \{\mathbf{R} \in \mathbb{R}^{n \times n} | \mathbf{R}\mathbf{R}^{\mathbf{T}} = \mathbf{I}^{\mathbf{n}}, \det(\mathbf{R}) = 1\} \tag{3.8}$$

The proofs of the three axioms for (Lie) groups as described in Section 3.2.3 are outside of this work's scope but are demonstrated in [87] or [43].

We now consider the more specific Special Orthogonal Group in three dimensions SO(3). Rotational matrices have the computationally desirable property that they linearize rotating a vector. Linear operations are very efficient on modern CPUs. However, a "problem" with rotation matrices is that there exists no notion of an infinitesimal rotation matrix. Hence, they can't be parametrized by time - which is crucial in any dynamic (e.g., robotics) application. As mentioned earlier, the combination of a unit-length rotation axis $\mathbf{u}$ and an angle around that axis θ can also describe a rotation. In this representations there exist infinitesimal rotations, i.e. when $d\theta \rightarrow 0$. Thus, we would like a transformation from rotation matrices to their corresponding angle-axis

representation and vice versa. To get there, assume that we rotate some point $\mathbf{p}$ around an axis $\mathbf{u}$ at constant unit velocity:

$$\dot{\mathbf{p}}(t) = \mathbf{u} \times \mathbf{p}(t) = \mathbf{u}^{\wedge}\mathbf{p}(t) \tag{3.9}$$

This gives us the velocity at point $\mathbf{p}$ at time $t_0 = 0$ in the form of a time-invariant Ordinary Differential Equation (ODE) whose solution is

$$\mathbf{p}(t) = e^{\mathbf{u}^{\wedge}t}\mathbf{p}(t_0) \tag{3.10}$$

Here $e^{\mathbf{A}}$ is the matrix exponential given by the Taylor's Series $e^{\mathbf{A}} = \mathbf{I} + \mathbf{A} + \frac{\mathbf{A}^2}{2!} + \frac{\mathbf{A}^3}{3!} \ldots$. Hence, rotating a point for θ units of time produces a functional relationship between the resulting net rotation and the angle and axis of this rotation: $\mathbf{R}(\theta, \mathbf{u}) = e^{\mathbf{u}^{\wedge}\theta} = e^{\boldsymbol{\phi}}$ The skew-symmetric elements $\boldsymbol{\phi}$ representing an angle-axis combination make up the Lie algebra $\mathfrak{so}(3)$ to the Lie group $SO(3)$, formally defined in Eq. (3.11) [87].

$$\mathfrak{so}(3) = \{\boldsymbol{\phi} \in \mathbb{R}^{3\times3} | \boldsymbol{\phi} = \phi^{\wedge}, \phi \in \mathbb{R}^3\} \tag{3.11}$$

The associated Lie bracket is $[\boldsymbol{\phi}_1, \boldsymbol{\phi}_2] = \boldsymbol{\phi}_1\boldsymbol{\phi}_2 - \boldsymbol{\phi}_2\boldsymbol{\phi}_1$. The derived functional relationship between elements of $\mathfrak{so}(3)$ and those in $SO(3)$ is called the exponential map and can be simplified as follows:

$$\mathbf{R}(\boldsymbol{\phi}) = \mathbf{R}(\theta\mathbf{u}^{\wedge}) = e^{\theta\mathbf{u}^{\wedge}} = \mathbf{I} + (1 - \cos(\theta))\mathbf{u}^{\wedge}\mathbf{u}^{\wedge} + \sin(\theta)\mathbf{u}^{\wedge} \tag{3.12}$$

Eq. (3.12) is precisely the Rodrigues formula 3.3 mentioned in Section 3.2.1. The inverse operation, mapping from rotation matrices to the corresponding axis-angle representation $\boldsymbol{\phi}$, is the logarithmic map: $\boldsymbol{\phi}(\mathbf{R}) = \ln(\mathbf{R})$ Eq. (3.13) shows how the angle θ and axis $\mathbf{u}$ can be derived from a rotation matrix $\mathbf{R}$. [87] describes the derivation in full detail.

$$\begin{aligned} \theta &= \arccos(\frac{\mathrm{tr}(\mathbf{R}) - 1}{2}) \\ \mathbf{u} &= \frac{1}{2\sin(\theta)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \end{aligned} \tag{3.13}$$

3.2.5 The Special Euclidean Group SE(3) and its Lie Algebra $\mathfrak{se}(3)$

The Special Euclidean Group SE(3) is informally an extension of SO(3) that covers arbitrary rigid-body transformations, i.e., combining a rotation and a translation. As already noted, 4-by-4 matrices constructed from a rotation matrix $\mathbf{R}$ and a translational

vector $\mathbf{t}$ also represent such transformations. The set of these transformations forms the Special Euclidean Group: $SE(3) = \{\mathbf{T} \in \mathbb{R}^{4\times4} | \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}, \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3\}$ As for SO(3), there is an equivalent generalization to n dimensions.

Similarly to SO(3), we need a different representation that resembles Euclidean space, enabling differentiation. Unsurprisingly, this space, the Lie algebra $\mathfrak{se}(3)$, consists of six-dimensional vectors, where three dimensions represent the translational component $\boldsymbol{\rho}$. The other three represent the rotational element $\boldsymbol{\phi} \in \mathfrak{so}(3)$. Utilizing the $\vee$ (vee) operator, which transforms a skew-symmetric matrix to its vector representation, Eq. (3.14) formally defines this algebra.

$$\mathfrak{se}(3) = \{\xi^\wedge \in \mathbb{R}^{4\times4} | \xi = \begin{pmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi}^\vee \end{pmatrix}, \boldsymbol{\rho} \in \mathbb{R}^3, \boldsymbol{\phi} \in \mathfrak{so}(3)\} \tag{3.14}$$

Analogously to SO(3) and following the example of [87], to get to a mapping between SE(3) and $\mathfrak{se}(3)$ consider rotating a point $\mathbf{p}$ around an axis $\mathbf{u}$ and translating the point by $\mathbf{t}$. We can describe the velocity of said point like so:

$$\dot{\mathbf{p}}(t) = \mathbf{u} \times (\mathbf{p}(t) - \mathbf{t}) = \mathbf{u}^\wedge \mathbf{p}(t) - \mathbf{u}^\wedge \mathbf{t}. \tag{3.15}$$

We now reformulate this using homogeneous coordinates and matrix-vector multiplication as

$$\begin{pmatrix} \dot{\mathbf{p}} \\ 0 \end{pmatrix} = \begin{bmatrix} \mathbf{u}^\wedge & -\mathbf{u}^\wedge \mathbf{t} \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix} \mathbf{p}(t) = \begin{bmatrix} \mathbf{u}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & \mathbf{0} \end{bmatrix} \mathbf{p}(t) = \boldsymbol{\xi}^\wedge \mathbf{p}(t). \tag{3.16}$$

Again, we arrive at a differential equation whose solution is the exponential map: $\mathbf{p}(t) = e^{\boldsymbol{\xi}^\wedge t} \mathbf{p}(0)$. We then perform a rotation for θ units of time t to arrive at

$$\mathbf{T}(\theta, \mathbf{u}, \mathbf{t}) = e^{\boldsymbol{\xi}^\wedge \theta}. \tag{3.17}$$

$\boldsymbol{\xi}^\wedge$ is also known as the twist-parametrization of a rigid-body transformation.

Eq. (3.18) gives the closed-form solution [87].

$$\mathbf{T}(\theta, \mathbf{u}, \mathbf{t}) = e^{\boldsymbol{\xi}^\wedge \theta} = \begin{pmatrix} e^{\mathbf{u}^\wedge \theta} & (\mathbf{I} + \frac{1-\cos(\theta)}{\theta} \mathbf{u}^\wedge + \frac{\theta - \sin(\theta)}{\theta} \mathbf{u}^\wedge \mathbf{u}^\wedge) \boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{pmatrix} = \begin{pmatrix} e^{\mathbf{u}^\wedge \theta} & \mathbf{J}\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{pmatrix}. \tag{3.18}$$

Computation of the angle θ and rotation axis $\mathbf{u}$ from $\mathbf{R}$ is equivalent to the log map of SO(3) (see Eq. (3.13)). Eq. (3.19) describes the recovery of the translational component $\boldsymbol{\rho}$ from the translation $\mathbf{t}$ and the previously computed angle and rotation axis.

$$\boldsymbol{\rho}(\mathbf{t}, \theta, \mathbf{u}) = \mathbf{J}^{-1} \mathbf{t} = (\mathbf{I} - \frac{\theta}{2} \mathbf{u}^\wedge + (1 - \frac{\theta(1 + \cos(\theta))}{2\sin(\theta)}) \mathbf{u}^\wedge \mathbf{u}^\wedge) \mathbf{t} \tag{3.19}$$

These definitions for transformation in Euclidean space for rigid bodies in both representations now allow us to compute transformations of rigid bodies efficiently and likewise utilize their infinitesimal elements to e.g. calculate velocities at a given moment in time. When these transformations describe a coordinate system's relative transformation attached to a rigid body w.r.t. some reference coordinate system, we will interchangeably use the term *pose* and *transformation*.

3.3 Feature Points

Extracting and matching (that is, reidentifying) features or interest points from sensor output - most commonly images, which we will restrict ourselves to here - is a common task for many CV algorithms.

3.3.1 Extracting Features

Detected features should generally be invariant to any transformation of the image, in particular to changes in the following properties:

- brightness and contrast of the image
- the 2D location and orientation of a feature in the image
- the relative scale of the feature in the image
- relative rotational change between image (camera) and 3D space

Feature detectors generally present a trade-off between their computational efficiency and their invariance w.r.t. the properties mentioned above.

Different algorithms require varying features, e.g., a geo-mapping application may want to detect edges in satellite images to infer roads or other structures. Popular edge detectors are [20], [28], and [30].

Other typical features in images are corners. These have proven to be characteristic and re-identifiable points in an image and, hence, are often interchangeably referred to as interest points. Photography software uses such interesting points for stitching images together to form panoramic photos, for example. They are also an essential feature for the type of task at hand, i.e., motion tracking.

One of the most common corner detectors, the Harris-Corner detector [44], extracts corners from images by computing the image gradients I_x and I_y at each pixel (i.e., the change in intensity) along both x and y image dimensions. Large gradients are an indicator of edges in the image where the image intensity changes significantly, whereas large gradients in both dimensions signal an edge at a given pixel. An "edge-response"

function that captures these combined intensity changes together with non-maximum suppression (i.e., filtering only local maxima of edge responses) constitute the Harris-Corner detector.

Rosten and Drummond published a real-time capable detector (~ 20 times faster than Harris) in 2006 [98], titled FAST. This detector looks at a circular area around a pixel and checks whether a specific amount of contiguous pixels in this ring are above or below a certain threshold level indicating a corner. Other "classic" interest point detectors include [75, 74] and [108].

As DL has come to dominate many computer vision tasks, it has also made its way into feature extraction. The main problem for learned features is creating the dataset. Feature detectors typically extract thousands of features from images making the creation of a hand-annotated dataset infeasible.

The current state-of-the-art feature point extractor in Deep Learning, *SuperPoint* [29], combats this issue in a three-step approach. First, a base network, titled *MagicPoint*, is trained to detect corners of various rendered shapes in a synthetic dataset with known corner locations. *MagicPoint* uses an Encoder-Decoder architecture (see [6] for a detailed review of so-called autoencoders) to extract feature points. Images with arbitrary dimensions can pass through the network as it implements non-trained up- and down-sampling. In a second step, the authors generate pseudo-ground truth points for the target dataset of real images. They achieve this through their process of Homographic Adaption - transforming an input image through several (sensible/realistic) homographies and then feeding each image through *MagicPoint*. Finally, the images (and the detected features) are transformed back to their original state via the inverse homographies. The subset of overlapping features are the pseudo-ground truth features. Finally, the *SuperPoint* network is trained on the training dataset using the previously generated features as ground-truth locations. The *SuperPoint* net is similar to *MagicPoint*. However, it has a second loss head for computing the descriptors (see the following subsection).

Another DL-based feature detector is [124]. Although these types of feature detectors can deal with different image sizes by up- or downsampling them before feeding them through their learned network, they have the downside that the number of extracted features is upper-bounded by their output dimension (in contrast to non-DL feature detectors).

3.3.2 Creating Descriptors

As previously mentioned, it is often desirable not only to extract characteristic points or features in an image but also to be able to reidentify the same feature across multiple images (under different illumination, at varying scales, and from numerous

viewing angles). Reidentification requires a matching procedure between features. Such procedures so-called feature descriptors for interest point matching. For non-DL features, the feature extraction methods usually create this descriptor by exploiting information from the neighboring image region. For example, SIFT [74] uses oriented gradients of the pixels in a square region around a feature pixel to compute 128-dimensional descriptors demonstrably invariant to changes in viewpoint, illumination, scale, rotation, and translation. SIFT can then calculate similarities of features by taking the Euclidean distance between these descriptor vectors. The BRIEF descriptor [52] is an alternative 256-bit binary vector based on intensity comparisons between the feature pixel and randomly sampled neighboring pixels. Its main advantage is that the comparison between descriptors is very fast as computing the Hamming distance (vs. the Euclidean distance) is an efficient CPU-instruction-based operation. [99] introduced a popular alternative to SIFT (arguably the most popular, due to the only recently expired patent on SIFT) based on the FAST feature extractor and BRIEF feature descriptors. These so-called ORB features extend the FAST extractor to compute orientations of features alongside their location and, additionally, enhance the BRIEF descriptor to be invariant to rotation of a given feature.

Other feature extraction combined with descriptor creation methods include [9] and [4].

3.3.3 Matching Features

The descriptors of the extracted features need to be matched. As alluded to, this matching is typically based on the appropriate, e.g., Euclidean or Hamming, distance between vectors. The task is then to match two sets of descriptors (i.e., extracted from two different images of the same scene). An obvious solution to this association problem is the Hungarian algorithm described in Section 3.1.3. The similarity measure is the distances between the descriptors (and the distance must be below a certain threshold to be considered a good match). [74] introduced an extension to this matching by requiring the ratio of the distance to the closest feature over the distance to the second-closest feature to be above a certain threshold.

As with extracting features, DL-approaches that learn feature association also exist, e.g., *SuperGlue* [102].

3.4 Stereo Vision

3D reconstruction is an area of CV that concerns itself with extracting 3D information of objects or scenes from sensors, typically in the form of point clouds (i.e., a collection of 3D points) or three-dimensional shapes (i.e., arbitrary surfaces or shapes parametrized

in $\mathbb{E}^3$). Reconstruction requires localization of sensor readings in 3D, i.e., a depth component z in addition to 2D coordinates x and y (plus, optionally, other sensor information such as color). This depth component can be either directly extracted if it's part of the sensor reading (e.g., LiDAR, Sonar, or RGB-D cameras) or be calculated indirectly. Human vision is the underlying inspiration for the latter approach: depth information is inferred from two cameras viewing the same scene. Such a process is called stereo vision, and it is a sub-area of the more generalized method of inferring depth from multiple images of the same scene or object (known as Multi-View Geometry). [46] and [54] offer in-depth discussions and explanations on the varying methods whereas [82] provides a briefer overview.

3.4.1 Camera Calibration

In the case of 2D RGB (or grayscale) cameras, the first step to reconstructing three-dimensional points is understanding the projection of a point in Euclidean space to the two-dimensional image plane. Frequently the Euclidean points are not given (or desired to be known) in a coordinate frame attached to (and possibly moving with) the camera but in a fixed reference coordinate system. Typically, this reference coordinate system is called the world coordinate frame. Hence, transforming the point from world coordinates to image plane coordinates entails two steps:

1. Transform the point ${}^{w(orld)}\mathbf{p}$ to the camera coordinate system via the transformation ${}^{w(orld)}_{c(am)}\mathbf{T} \in SE(3)$ (see Section 3.2 for necessary background on rigid body transformations)

2. project the resulting point ${}^{c(am)}\mathbf{p}$ from the camera coordinate system to the 2D image plane.

The process of finding ${}^{w}_{c}\mathbf{T}$ is known as extrinsic camera calibration and needs to be re-estimated if the relative pose of the camera changes w.r.t. the reference frame. Section 3.4.5 describes a possible procedure to finding such a pose. For now, let's assume 3D points are in camera-coordinates s.t. only the projection function $\pi(.)$ onto the image plane resulting in pixel-coordinates is of interest. The most straightforward projective relationship is given by modeling the camera as a so-called (lense-less) pinhole camera resulting in a linear projection (Fig. 3.5 illustrates this model).

In its simplest form, the only parameters determining this function are the focal point of the camera f (the distance from the image screen to the point where the camera bundles light), and the principal point of the image $\mathbf{c}$ (the point at which the lens axis intersects the image). Note that for digital images, the projected point needs to be mapped to the pixel-grid of the image plane (the coordinate frame spanned by u and

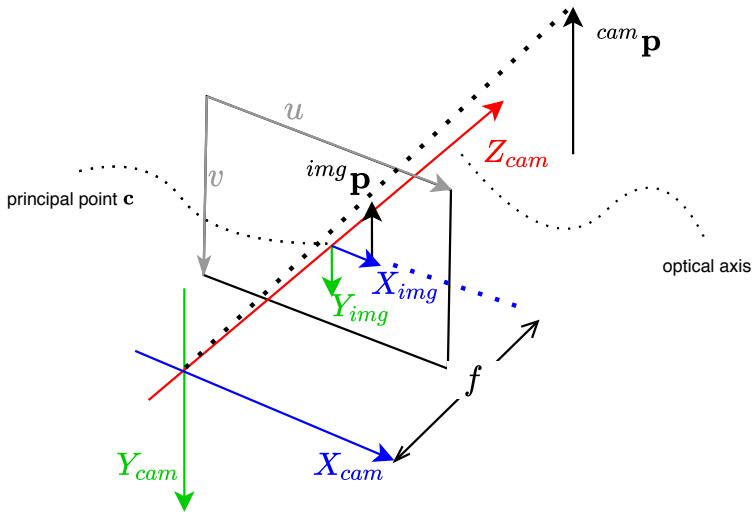


Figure 3.5: The pinhole camera model results in a linear projection function, but typically this simplification comes at the cost of accuracy. The u, v axes are the pixel-coordinate system whereas the X_{img}, Y_{img} coordinate system is in world units and has its origin at the principal point **c** where the optical axis intersects the image plane.

v). The pixel dimensions p_x, and p_y are necessary to compute the focal length and principal point coordinates from world units (i.e., meters) to image plane units (i.e., pixels) and hence discretize the projections. The resulting focal length and principal point components in pixel coordinates are denoted by subscripts of x and y. Eq. (3.20) shows this linearized projection:

$$ {}^{cam}p_z \begin{pmatrix} p_u \\ p_v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{cam}p_x \\ {}^{cam}p_y \\ {}^{cam}p_z \end{pmatrix} \tag{3.20} $$

Note that the z-coordinate of the Euclidean point scales the resulting image point in pixel coordinates. This scaling factor is a result of the fact that all points along the projective line connecting ${}^{cam}\mathbf{p}$ to ${}^{img}\mathbf{p}$ result in the same projected point ${}^{img}\mathbf{p}$. Consequently, this entails that one cannot recover the Euclidean point's depth from a projected point.

A model that allows for non-rectangular pixels introduces a skew-coefficient $s = f_x \tan(\alpha)$ where α is the angle between the x and y axes of the pixels (or, equivalently, the image) - see Fig. 3.6 for a visualization of the skew coefficient. The following equation extends the pinhole model to allow for a non-zero skew-coefficient:

$$ {}^{cam}p_z \begin{pmatrix} p_u \\ p_v \\ 1 \end{pmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} {}^{cam}p_x \\ {}^{cam}p_y \\ {}^{cam}p_z \end{pmatrix} \tag{3.21} $$

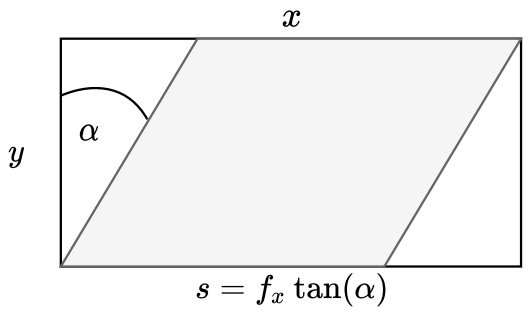


Figure 3.6: The skew coefficient s measures the non-rectangularity of the pixels of a camera. Typically it is assumed to be 1 (i.e. pixels are rectangular). f_x is the x-component of the focal length vector in pixels.

Of course, this pinhole camera model is a highly idealized version of a projection function - real cameras do have lenses that result in various non-linear aberrations. [115] presents an overview of camera models in use in real-world applications today.

We will, however, only discuss the camera model utilized by this work. This model is an extension of the pinhole camera allowing for radial and tangential distortion (see Fig. 3.7) and is implemented by the computer vision library OpenCV[1] which we work with in this thesis.

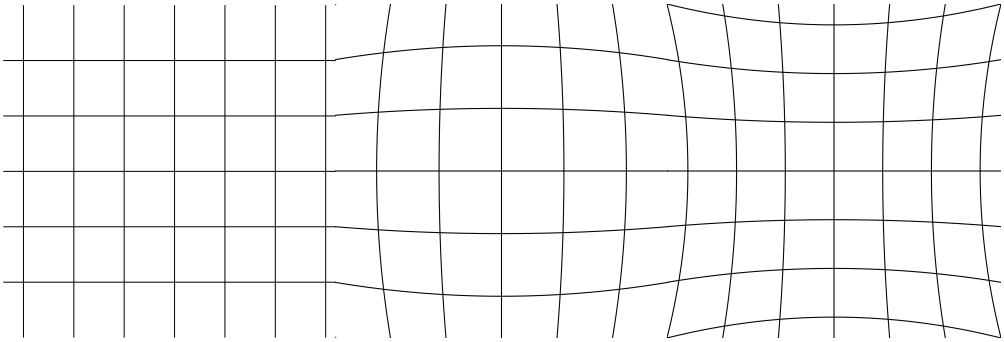


Figure 3.7: *Left*: No distortion - lines in the coordinate grid are equally spaced. *Middle*: Radial distortion - image lines converge outside of the image to form a barrel-like effect in the resulting image. *Right*: Tangential distortion - image lines diverge outside of the image resulting in a "pincushion"-type effect.

As these two types of distortions introduce non-linearity into the projection function $\pi(.)$, they can't be succinctly computed via matrix-vector multiplication but require the following steps:

$$
\begin{aligned}
p'_x &= \frac{{}^c p_x}{{}^c p_z} \\
p'_y &= \frac{{}^c p_y}{{}^c p_z} \\
r^2 &= p'^2_x + p'^2_y \\
p''_x &= p'_x \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2t_1 p'_x p'_y + t_2 (r^2 + 2p'^2_x) \\
p''_y &= p'_y \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2t_2 p'_x p'_y + t_1 (r^2 + 2p'^2_x) \\
p_u &= f_x p''_x + c_x \\
p_y &= f_y p''_y + c_y
\end{aligned}
\tag{3.22}
$$

In Eq. (3.22) the radial distortion is influenced by the parameters $k_i, i \in [1, 6]$ and

[1] `https://www.github.com/opencv/opencv`

the tangential distortion parameters by t_1 and t_2. Overall these parameters can be combined to $\mathbf{d}^T = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & k_5 & k_6 & t_1 & t_2 \end{pmatrix}$.

The process of determining the parameters of the projection function (in our case, the intrinsic parameters **A** - the focal length and principal point - and the distortion parameters **d**) of an assumed camera-model is called camera calibration. As the internal camera parameters typically do not change (for non-photographers), one needs to perform this process only once.

To calibrate a camera, one uses a so-called calibration object or rig. This calibration object is equipped with features easily extracted from and recognized in its projected image representation by pattern matching techniques not further discussed here. The calibration object is typically planar (s.t. it is entirely spanned by two of the three dimensions) with repetitive patterns (see Fig. 3.8 for an example of such an object) where the relative distance between elements of the pattern is known.

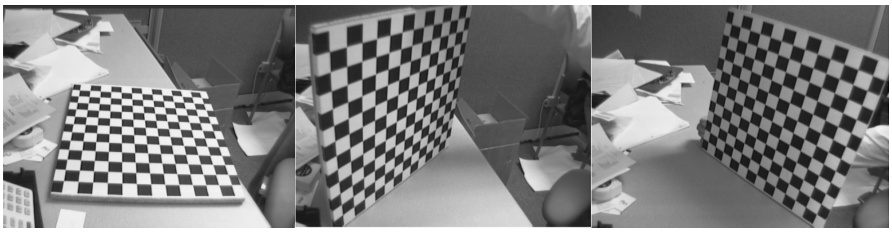


Figure 3.8: Multiple images from different perspectives taken of a chessboard pattern used as a calibration object (images are freely available at `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/calib_example/index.html`)

The object's origin is then either a uniquely recognizable feature or at some unique location w.r.t. the entire pattern. This association process results in 2D-3D correspondences with known Euclidean points in the calibration object frame. One now jointly estimates the extrinsic parameters (the camera pose w.r.t. the object frame), the intrinsic and distortion parameters (of the projection function) by projecting the calibration points to the image plane and minimizing the distance of the projected points to the extracted feature locations. This distance is referred to as the reprojection error. Note that when using such planar calibration objects, at least two images of the object are necessary, and the accuracy of the estimated parameters increases with the number of images.

Also, note that while a closed-form solution exists, noise in the projection and simplifications inherent to the model make a non-linear optimization objective preferrable [128]. This non-linear least-squares objective is as follows and can be solved using an appropriate method, e.g. the Levenberg-Marquadt algorithm [68, 78]:

$$ {}^{w}_{c}\mathbf{T}^{*}_{i}, \mathbf{A}^{*}, \mathbf{d}* = \text{argmin} \frac{1}{2} \sum_{i=0}^{n_{images}} \sum_{j=0}^{n_{features}} || \begin{pmatrix} f_u^{ij} \\ f_v^{ij} \end{pmatrix} - \pi(({}^{w}_{c}\mathbf{T}_i)^{-1} {}^{w}\mathbf{p}^{ij}, \mathbf{d}, \mathbf{A}) ||^2_2 \tag{3.23} $$

$f^{ij}_{\{u,v\}}$ in Eq. (3.23) are the feature coordinates of feature j in image i, ${}^{w}\mathbf{p}^{ij}$ is the associated object point, and ${}^{w}_{c}\mathbf{T}_i$ the relative camera-object pose at image i.

Minimization of this objective then leads to a known projective function necessary for the following steps of recovering 3D scene information. The inverse of the projection function $\pi(.)$ "back-projects" a point from the image plane to camera coordinates and is the ray through all Euclidean points which project to the same image points.

3.4.2 Epipolar Geometry

Now that we have calibrated cameras, the remaining piece enabling 3D reconstruction is the geometry between these two cameras. The intrinsic parameters of both cameras as well as the relative rigid body transformation ${}^{r}_{l}\mathbf{T} \in SE(3)$ from the left to the right camera entirely define this relationship. Fig. 3.9 shows a general configuration of such a stereo setup.

Note that the given configuration and following discussion assumes a pinhole-camera model (as described in the previous section) where both cameras have the same intrinsic matrix $\mathbf{A}$. Additionally, the virtual image plane (upright, between object and camera) is considered instead of the real, inverted image plane behind the camera (as was done in Fig. 3.5).

When projecting some arbitrary scene point ${}^{w}\mathbf{p}$ (seen by both cameras) to the left and right image planes, this results in two 2D points $\mathbf{p}_l$ and $\mathbf{p}_r$, respectively. Due to the linearity of the projection function, the line, or ray, passing through the 3D point and the respective image points also passes through the camera's corresponding optical centers $\mathbf{O}_l$ and $\mathbf{O}_r$. As these rays are (for objects at finite distances) not parallel and, generally, $\mathbf{O}_l \neq \mathbf{O}_r$, all points along these rays through $O_{\{l,r\}}$ project to a single point in the corresponding image plane: $\mathbf{p}^0_{\{l,r\}}$. However, on the respective other image, the points along these rays $\mathbf{r}_{\{l,r\}}$ project to a line $l_{\{r,l\}}$. This line is called the epipolar line. All epipolar lines on one image plane intersect at the epipolar point $\mathbf{e}_{\{l,r\}}$. One can also construct these epipolar points by projecting one camera's optical center to the other's image plane. Furthermore, these two projection rays coincide, i.e., there is a line going through both optical centers and both epipolar points. This line is often referred to as the baseline b. The plane constructed from the baseline and the rays $\mathbf{r}_{\{r,l\}}$ is called the epipolar plane. From the notion of an epipolar line, the so-called epipolar constraint follows [46]: if one knows the geometry between two cameras and wants to find a given

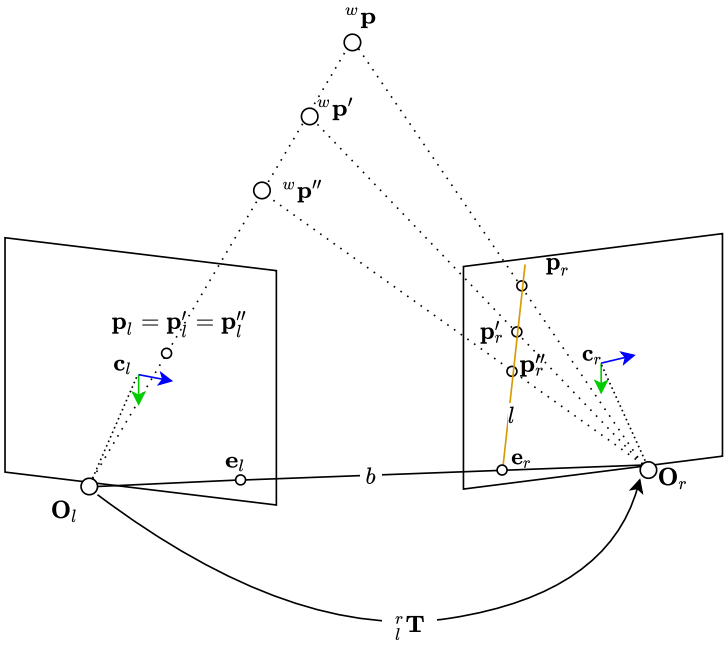


Figure 3.9: The epipolar geometry between two pinhole-cameras. A point projects via the optical centers to the image planes of both cameras.

point from the left image in the right image, one can restricts one's search to the right image's corresponding epipolar line. Put differently, transforming the left ray $\mathbf{r}_l$ to the right coordinate system (i.e. ${}^r_l\mathbf{R}\mathbf{r}_l + {}^r_l\mathbf{t}$) and taking the cross-product of this transformed ray and the baseline (which is identical to ${}^r_l\mathbf{t}$) produces a vector normal to the epipolar plane:

$$ {}^r_l\mathbf{t} \times ({}^r_l\mathbf{R}\mathbf{r}_l + {}^r_l\mathbf{t}) = {}^r_l\mathbf{t} \times ({}^r_l\mathbf{R}\mathbf{r}_l). \tag{3.24} $$

Thus, taking the dot-product of this vector and the right ray must be zero (as these must also be perpendicular):

$$ \mathbf{r}_r \cdot ({}^r_l\mathbf{t} \times {}^r_l\mathbf{R}\mathbf{r}_l) = 0 \tag{3.25} $$

As the projected points are proportional to the rays (and utilizing the cross-product to skew-symmetric transformation introduced in Section 3.2.1) the previous equation simplifies to:

$$ \mathbf{p}_r^T ({}^r_l\mathbf{t}^\wedge {}^r_l\mathbf{R})\mathbf{p}_l = \mathbf{p}_r^T \mathbf{E} \mathbf{p}_l = 0 \tag{3.26} $$

$\mathbf{E}$ in Eq. (3.26) is called the essential matrix and captures the relative transformation between the two cameras.

The epipolar constraint formulated by Eq. (3.26) now allows estimating the essential matrix given sufficient point correspondences between the left and right images. This estimation can be done with the Eight-Point Algorithm [73, 45] given - as the name suggests - at least eight point pairs.

Typically, one calibrates the intrinsic parameters of each camera and the relative extrinsic calibration between cameras conjointly.

A special case of the epipolar geometry is the so-called epipolar standard configuration (see Fig. 3.10): here, both image planes lie in the same plane, s.t. all epipolar lines are parallel to each other (and the baseline), and the epipoles lie at infinity.

This configuration is especially desirable when searching for the corresponding point in the other image by exploiting the epipolar constraint: the configuration constrains the search to a single row in the image plane for which efficient implementations exist. Additionally, if one has a calibrated stereo camera system, one can compute the rotation matrices necessary to transform both image planes to the epipolar standard configuration. From this, one can also transform the images so that they display the desired configuration, and their epipolar lines are parallel. [46] explains this process known as image rectification.

3.4.3 Stereo Triangulation

Knowing the intrinsics of both cameras and the relative transformation between them now allows us to recover the lost depth information from a left-right pair of projected

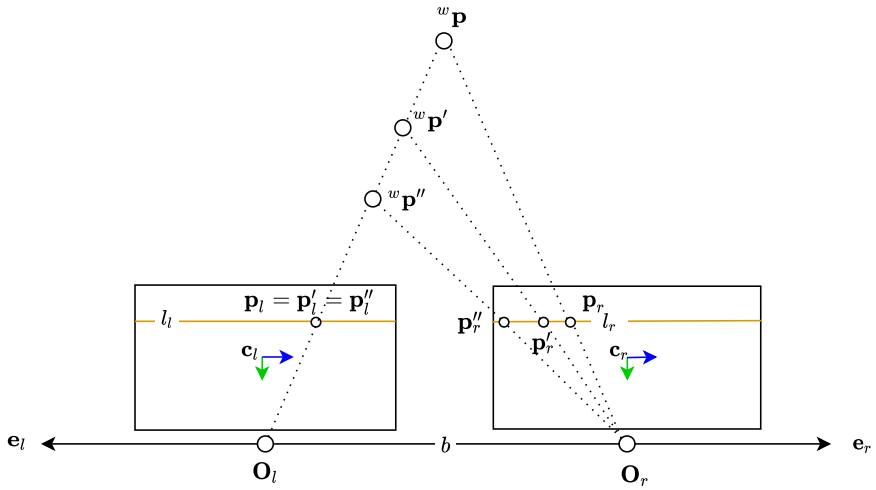


Figure 3.10: In the epipolar standard configuration, both image planes lie in the same plane, making the epipolar lines parallel as the epipoles lie at infinity.

points. This procedure is known as triangulation: given two corresponding points $\mathbf{p}_l$ and $\mathbf{p}_r$ from left and right images, respectively, the point where the "back-projected" rays ${}^r\mathbf{r}$ and ${}^l\mathbf{r}$ intersect is the underlying 3D point ${}^c\mathbf{p}$. However, due to many noise sources (discretized image plane, lenses, etc.), these rays mostly do not intersect (Fig. 3.11 illustrates this). In the most general sense (i.e. when the projection functions $\pi_{\{l,r\}}(.)$) of the cameras are non-linear) one can solve triangulation by posing the problem as a non-linear least squares problem and using e.g. Gauss-Newton [14] to minimize the reprojection errors:

$$\min_{{}^w\mathbf{p}} \sum_{i \in \{l,r\}} ||\mathbf{p}_i - \pi({}^i_w\mathbf{T}\,{}^w\mathbf{p})||_2^2 \tag{3.27}$$

${}^w\mathbf{p}$ is really the point in the left camera coordinate system as we only know the relative transformation ${}^l_r\mathbf{T}$ s.t. ${}^w_l\mathbf{T} = \mathbf{I}^4$ and ${}^w_r\mathbf{T} = {}^l_r\mathbf{T}$.

If the projection functions are linear (i.e. given by the extrinsic camera poses and intrinsic camera matrices), one can also pose triangulation as a regular least-squares problem.

3.4.4 RANSAC

Dealing with noise is a challenge for any task that involves real-world data from imperfect sensors or applies simplified assumptions about said's data underlying model. Estimating a model's parameters using noisy data will especially result in

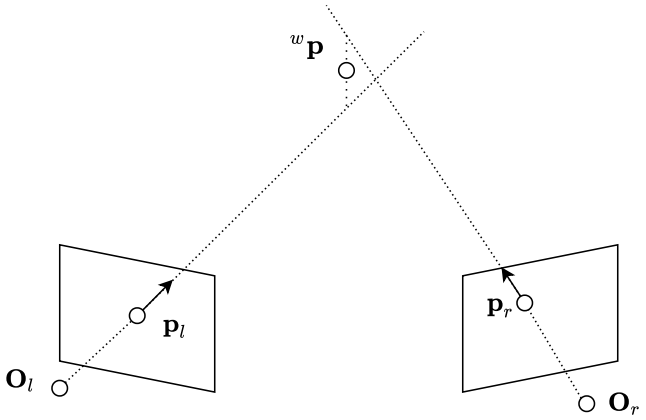


Figure 3.11: In the presence of noise, the two projection rays for a pair of projected points do not intersect. An estimate of the 3D location is then the middle of the shortest path connecting these two rays.

errors if the noise is not uniform across all data points. Under this assumption, one can divide the data points into two subsets: data supporting a model (within a certain noise threshold), dubbed inliers, and outliers - data corrupted by extreme noise which holds no information about the parameters to be estimated. Random Sample Consensus (RANSAC) is a popular method developed in the early 1980s for robust model estimation [36]. RANSAC is an iterative process that consists of the following two steps and terminates when a stopping criterion is met:

1. Randomly select a minimum amount of n points (hypothetical inliers) from the data from which the model parameters can be estimated and fit the model to these

2. Determine the number of points that support this model (the consensus set), i.e., their deviation w.r.t. the model falls within a given threshold

The stopping criteria are: the consensus set must reach a threshold, or the number of iterations must exceed some maximum. The values of these criteria are application-specific. Typically, if the procedure reaches the maximum number of iterations without crossing the inlier ratio (i.e., inliers over total points), the estimation is considered failed. If, however, the process finishes successfully, the model can then be re-estimated on the entire consensus set leading to more robust parameters.

Generally, there is no guarantee that RANSAC finds the inlier set. At every iteration, n points are selected uniformly from the entire dataset. Given a ground truth inlier ratio $t_i = \frac{n_i}{n}$ where n_i are the number of inliers, the probability of selecting n inlier

points in a single iteration is $(t_i)^n$ and, conversely, the probability of selecting at least one outlier is $1 - (t_i)^n$. Assuming for simplicity that a single outlier in the hypothetical inlier set will result in a model that doesn't reach the required inlier ratio, it follows that the probability of selecting non-inlier sets for k consecutive iterations is $(1 - (t_i)^n)^k$. This result now allows one to set the maximum iterations k_{max} by choosing a probability p for successful completion of the process. The converse probability of failure is then $1 - p$ and equating the previously deduced probability of failure after k iterations to this, i.e.

$$(1 - (t_i)^n)^k = 1 - p \tag{3.28}$$

we get

$$k_{max} = \frac{\log(1 - p)}{\log(1 - (t_i)^n)} \tag{3.29}$$

As previously mentioned, this deduction assumes that a single outlier in an inlier set will distort the resulting model to the point of failure. Therefore, k_{max} can be seen as an upper bound. Additionally, the inlier threshold t_i is generally unknown s.t. it makes sense to use a low estimate for this (and consequently compute a larger k_{max}).

An example of a (for simplicity, single iteration) RANSAC scheme for fitting a linear function is given in Fig. 3.12.

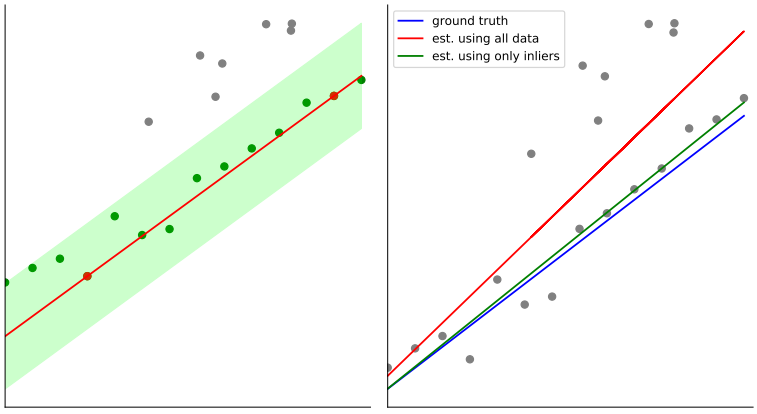


Figure 3.12: *Left:* A single iteration of a RANSAC scheme on a dataset distorted by outliers (gray) assuming a linear model. Two points (red) are used to estimate the model (red line) and the consensus set (green) are all points within a certain threshold (green area). *Right*: The same set and the underlying ground truth function (blue), the estimated function using the entire consensus set (green) and the estimated function using all datapoints (red). *Best viewed in color.*

3.4.5 Perspective-n-Point Camera Pose Estimation

Given a set of known 3D points and their corresponding projected points on an image plane, the extrinsic pose ${}^{w}_{c}\mathbf{T} \in SE(3)$ can be estimated via a process known as Perspective-n-Point (PnP) [72]. Generally, the problem is similar to the triangulation problem described in Section 3.4.3, the difference being that we only have a single camera with an unknown pose and multiple Euclidean points with known coordinates. Still, for n 2D-3D correspondences the generalized non-linear minimization objective looks familiar:

$$\min_{{}^{c}_{w}\mathbf{T}} \sum_{i}^{n} ||\mathbf{p}_i - \pi({}^{c}_{w}\mathbf{T}\,{}^{w}\mathbf{p}_i)||_2^2 \tag{3.30}$$

The desired pose has six DOFs and, hence, a non-linear solver for Eq. (3.30) requires a good initialization for convergence.

However, several linear solutions exist, the earliest dating back as far as 1841 [72]. At a minimum, PnP needs three points to solve the problem. This approach is called P3P [122]. However, using three points results in four feasible solutions. Adding a fourth point can resolve the ambiguity. A linear approach using Direct Linear Transform (DLT) as a solver exists as well.

The arguably most common procedure for solving the PnP problem is Efficient Perspective-n-Point (EPnP) and was published in 2008 by Lepetit et al. [67]. This method solves the non-linear least-squares from Eq. (3.30) via Gauss-Newton but initially estimates the rotational and translational components by expressing the world points as linear combinations. The factors are shared across points and termed virtual points. The resulting problem is that of estimating these control points on the image plane.

As outliers (i.e., incorrect 2D-3D correspondences) are a common problem in the PnP scenario, one commonly applies a RANSAC scheme to the estimation (as detailed in Section 3.4.4).

3.5 SLAM

SLAM is the process by which a robot, car, or even hand-held device uses its sensors to build a model of its surroundings and to localize itself in it simultaneously - without having any a priori information of its environment.

3.5.1 Early SLAM solutions

That is, the SLAM problem consists of an ego or camera trajectory made up of n poses ${}^{w}_{c}\mathbf{T}_i \in SE(3)$ with $i \in [1, n]$ and m Euclidean points (so-called landmarks $\mathbf{l} = \{\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{l}_m\}$) that describe the map.

Each landmark $\mathbf{l}_i$, in turn, has k measurements or observations $\mathbf{o}^i = \{\mathbf{o}^i_1, \mathbf{o}^i_2, ..., \mathbf{o}^i_k\}$ associated with it.

The set of all observations is then $\mathbf{o} = \{\mathbf{o}^1, \mathbf{o}^2, ..., \mathbf{o}^m\}$. Additionally, landmarks may hold supplementary information (e.g. an RGB-color value or a descriptor - see Section 3.3).

Generally, the problem for a single time-step t can be stated probabilistically as the following posteriori probability encompassing the ego-pose and landmarks:

$$P({}^{w}_{c}\mathbf{T}_t, \mathbf{l} | {}^{w}_{c}\mathbf{T}_0, \mathbf{o}_{0:t}) \tag{3.31}$$

In Eq. (3.31), $\mathbf{o}_{0:t}$ denotes all landmark observations made up until the current time-step.

The first SLAM systems were developed by the mobile-robotics community in the 1990s and early 2000s, e.g. [66, 41, 42, 81].

These early approaches solve the SLAM problem either via linear-filtering [66] (i.e. with Extended Kalman Filters (EKFs) - see [16] for a discussion on Kalman filters), non-linear filtering (i.e. with Rao-Blackwellized particle filter [81]), or through Maximum Likelihood Estimates (MLEs) [17]. Generally, these formulations encode the system state (poses & landmarks) in a single vector and extended the formulation given in Eq. (3.31) to include knowledge of a robot's control inputs $\mathbf{u} = \{\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_n}\}$:

$$P({}^{w}_{c}\mathbf{T}_t, \mathbf{l} | {}^{w}_{c}\mathbf{T}_0, \mathbf{o}_{0:t}, \mathbf{u}_{0:t}) \tag{3.32}$$

A central realization of research of the SLAM problem conducted in the wake of the success of these early systems was that the errors of landmarks observations and pose estimates for a single timestep are highly correlated and, hence, result in a much more sparse, and thus numerically superior, problem than initially assumed [17].

3.5.2 Solving SLAM via MAP

The current standard solution (leaving out control inputs for simplicity) to solving SLAM is via Maximum A Posteriori (MAP):

$${}^{w}_{c}\mathbf{T}^*_t, \mathbf{l}* = \underset{{}^{w}_{c}\mathbf{T}_t, \mathbf{l}}{\operatorname{argmax}} P({}^{w}_{c}\mathbf{T}_t, \mathbf{l} | {}^{w}_{c}\mathbf{T}_0, \mathbf{o}_{0:t}) \tag{3.33}$$

Combining poses and landmarks to a single state-vector χ_t, applying Bayes' Theorem and assuming non-correlated measurements between timesteps, we arrive at

$$\chi_t^* = \underset{\chi_t}{\operatorname{argmax}} P(\chi_t) \prod_{k=1}^{t} P(\mathbf{o}_{0:k}|\chi_k) \tag{3.34}$$

Further assuming zero-mean Gaussian distribution of measurement noise, an underlying function $h(.)$ mapping landmarks-poses combinations to the measurement space, and utilizing the fact that maximizing a Gaussian-distributed function is equivalent to minimizing the negative log of it, we arrive at the following least-squares problem (see [17] for a step-by-step derivation):

$$\chi_t^* = \underset{\chi}{\operatorname{argmin}} \sum_{k=0}^{t} ||h(\chi_k) - \mathbf{o}_{0:k}||_2^2 \tag{3.35}$$

As with previously presented least-squares problems (see Section 3.4.1 and Section 3.4.3), non-linear solvers such as Gauss-Newton or Levenberg-Marquardt can solve this. Many implementations exploit the sparsity, as mentioned earlier, of the problem to solve it more efficiently.

Another critical aspect of SLAM is global map consistency. The process of ensuring such consistency is known as loop-closure and entails recognizing previously-seen places and updating the map once such a closure is detected. Loop-closure is necessary for accurate map and pose estimates as even minor estimation errors will accumulate over time to result in large so-called drift errors. Fig. 3.13 illustrates such a drift and the result of a successful loop-closure.

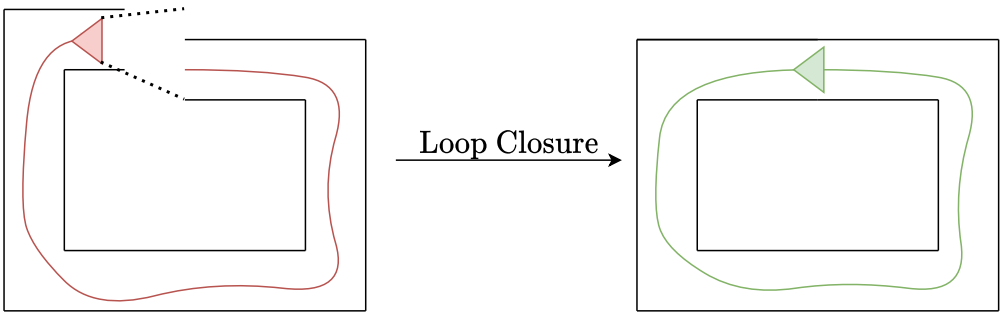


Figure 3.13: Loop closure enables globally consistent maps via place recognition (left) and consequent map and pose adjustments (right).

3.5.3 Visual SLAM

Although SLAM has its origin with robots equipped with LiDAR or SONAR sensors, the success of recreating 3D representations of scenes from a collection of unordered images taken from different viewpoints [2] (known as Structure from Motion (SfM)) motivated the use of RGB-cameras as input for SLAM systems. Different approaches exist here as well: either utilizing pixel-intensity data directly [34, 33] or via indirect feature points (see Section 3.3)[84, 85] extracted from the images. For the later approach this leads to the following updated optimization objective from Eq. (3.35) (where $\mathbf{o}_t^m$ is the image point associated to/measured as landmark $\mathbf{l}_m$ at timestep t):

$$ {}^{w}_{c}\mathbf{T}^*_t, \mathbf{l}^t = \text{argmin} \sum_{t,m} ||\mathbf{o}_t^m - \pi({}^{o}_{c}\mathbf{T}^{to}\mathbf{l}_m)||^2 \tag{3.36} $$

Eq. (3.36) is very similar to the optimization procedure in SfM known as BA. However, it differs in that SfM does this optimization globally, whereas SLAM is inherently iterative - optimizing at each step. Nonetheless, we will refer to this optimization in visual SLAM as BA.

A final important measure for SLAM systems is that they aim to operate in real-time. Although the problem is sparse, the number of variables grows exponentially with the number of frames, and, hence, one must apply techniques to constrain the optimization problem's dimensionality. One typically achieves this by choosing measurements to keep or discard and running the optimization only on a subset of the measurement data, e.g., by removing redundant or outlier poses and landmarks. The arguably most notable addition to visual SLAM systems is ORB-SLAM [84, 85]. This system introduces a so-called covisibility graph to "window" the optimization by only considering local frames. Informally, at some time t only poses and landmarks are included in the optimization if they are part of or see, respectively, the local scene viewed by the camera at time t.

4 Related Work

This work builds upon state-of-the-art DL object detection (see Section 3.1) and the tried-and-tested graph-optimization technique, i.e., BA, from SLAM (see Section 3.5 and Section 3.5.3). Although this work's focus is on 3D MOT, the "bigger-picture" goal is to incorporate the proposed system with a static SLAM system to facilitate dynamic scene understanding. Thus, this chapter will review systems that have incorporated object detection or object tracking to improve SLAM performance without implementing a MOT system. Next, the chapter discusses metrics that try to quantify MOT performance, and finally, existing 3D MOT (with and without SLAM incorporation) are presented.

4.1 Dynamic SLAM

Classical SLAM systems fail to differing degrees in highly dynamic environments based on a static-world assumption. However, the capability of dealing with such surroundings is paramount in moving towards robust autonomous navigation outside of empty factory floors. There exist several approaches which either detect objects, e.g., [126, 11, 101], or more generally, dynamic parts of the environment, e.g., [70], and then remove the associated areas from the sensor input to improve ego-tracking and static map creation.

[101] introduce an "object-oriented" approach to SLAM for environments with a priori knowledge of object types that are likely present in a given scene. The RGB-D system then detects objects and matches them via Iterative-Closest-Point (ICP) to 3D models in a database. The proposed systems removes object areas of the RGB-D frame from the static map pose-graph optimization. The authors claim improved performance compared to previous systems, especially w.r.t. scene geometry; however, they only show qualitative results. Li and Lee [70] presents a system that divides an RGB-D depth map into static and dynamic points. The authors accomplish this by transforming the previous map to its expected representation via an estimated ego-motion. They then perform ICP between this estimated map and the current depth map and discard collections of points whose Euclidean distance is greater than some threshold (they call these "dynamic edges"). The authors evaluate their system on the indoor TUM dataset [111] and compare their ego-tracking to the current state-of-the-art RGB-D-based visual odometry (i.e., no loop-closure) systems [56, 57] and show significant improvement

of ego-tracking in highly dynamic environments. Additionally, they exhibit improved performance compared to the at-the-time state-of-the-art non-real-time RGB-D SLAM system [113].

Another system that excludes objects from the SLAM optimization problem is DynaSLAM [11]. The authors present monocular, stereo and RGB-D flavors of their system. In their proposal, the authors detect objects via the state-of-the-art object detector MaskR-CNN [47]. In the RGB-D case, they additionally use multi-view geometry assumptions to enhance the detected object (e.g., the book a person is holding is included in the final dynamic object although it was not part of the MarkR-CNN detection). The system then excludes those dynamic parts of the frame when optimizing the ego-pose and static map. Furthermore, they estimate the masked frame-parts from the existing static map and output an "inpainted" frame without the dynamic objects. The authors evaluate their system on the TUM [111] and KITTI [38] datasets and compare ego-tracking of their system to ORB-SLAM [84]. DynaSLAM proves to be much more robust than ORB-SLAM in dynamic environments.

DS-SLAM [126] follows a similar approach to DynaSLAM. Yu et al. extend ORB-SLAMv2 [85] to more robustly handle dynamic environments. The authors achieve this by introducing a semantic segmentation network (SegNet [5]) which detects objects. They then exclude the detected objects from the input frame for ORB-SLAMv2. Additionally, for detected features, they perform an optical-flow-based check to determine whether a feature has moved significantly between frames and if so, regard the feature as an outlier. They then compare their system to "vanilla" ORB-SLAMv2 on the RGB-D TUM dataset [111] and show one order-of-magnitude improvement in ego-tracking accuracy for highly dynamic scenes.

The relatively recent ClusterSLAM [50] assumes no a priori knowledge of object types or topologies. Instead, it clusters objects based on groups of landmarks that show similar relative motion between frames. This procedure results in clusters for each object as well as a cluster for the static map. All landmarks clusters are then optimized via BA independently; ego-motion is estimated from the static cluster. The authors show that disjunct optimization of clusters outperforms a single optimization objective that combines all clusters. Although objects are tracked, the authors only show improved ego-tracking and landmark location estimates compared to [85, 11, 92, 86] on synthetic indoor and outdoor scenes created via [110] and [32].

Another issue specific to monocular visual SLAM is that of scale ambiguity. RGB-D systems or stereo cameras do not suffer from this as they either directly (RGB-D) or indirectly (the baseline between stereo cameras) deduce world points in meters. However, [37] shows that detecting objects with a known scale can diminish this scale ambiguity for monocular setups. The presented system detects and tracks cars by detection (based on [127] and [39]) of 2D bounding boxes. It represents objects by a

center point and a fixed dimension (i.e., a cube around a center point). This center is then included in the graph-optimization problem to recover scale. Additionally, points in regions of detected objects are not used as static map features. Note that the system is only able to deal with stationary objects and fails to track moving vehicles. The authors compare performance on the KITTI dataset [38] of their SLAM system with and without including objects in their graph-optimization. They show that the inclusion of this prior scale knowledge immensely reduces the scale error.

The presented research unambiguously demonstrates that a 3D MOT system that can mask sensor input for an existing static SLAM system will result in the latter's improvement.

4.2 Combining SLAM and Multi-Object Tracking

There exist several works which combine visual SLAM and MOT [69, 49, 7, 123, 31].

Dong et al. [31] propose a system that decouples the SLAM and the MOT subsystems. For performing ego-tracking and static map creation, the authors utilize [114] for their non-real-time and [84] for their real-time capable version. Their implementation describes objects by a pose $\in SE(3)$ and an a priori assumed shape. The authors detect and track objects via an EKF, a combination of a DL object detector (SubCNN [121] for the non-real-time version and YOLO [93] otherwise) and the object's class and pose. The resulting system is evaluated on 3D object detection (not tracking) accuracy on the KITTI dataset [38] and compared to "vanilla" SubCNN [121] which it outperforms.

In contrast to the previous system, CubeSLAM [123] proposes a sparse feature-based approach that tightly couples SLAM and MOT s.t. both systems benefit from each other. First, objects are detected via [93] for indoor scenes or [19] for outdoor scenes. Concurrently, a static SLAM system based on ORB-SLAMv2 [85] estimates the ego-pose. Detections are associated with existing object tracks by estimating existing tracks' positions via optical flow and performing 2D IoU on the projected bounding box and the detected object bounding box. The authors then formulate a graph-optimization objective that includes static points and object landmarks. Separately, for each object, a 9 DOFs bounding box is first estimated from its 2D bounding box and viewing point and then regressed towards the optimized object landmarks' locations. The authors evaluate ego-tracking performance as well as 3D object detection on the KITTI [38] and SUN RGBD [109] datasets. They show that in many (dynamic) cases, their system is more robust than existing non-dynamic SLAM implementations (e.g., [85]) and performs comparably or better in 3D object detection to comparable feature-based object detectors. However, the DL 3D detector Deep3D [83] shows superior performance based on 3D IoU.

A non-feature-based system that aims to couple MOT and SLAM to create object tracks, as well as dense scene representation, was presented by Barsan et al. [7]. In their work, the researchers propose a stereo-based system that separates its environment into moving objects, potentially moving objects (e.g., stationary vehicles), and the static map. They achieve this by first computing visual odometry (i.e., the ego pose) from a sparse scene flow. Additionally, they use [25] for multi-object detection and associate detections across frames via simple IoU. Object motion is estimated from scene flow between detections of consecutive frames. The authors mask the input images with the detected objects and use them for the static map. Moreover, for reconstructing objects as well as the map, they compute dense depth maps from the stereo input and then use InfiniTAM [55] for volumetric fusion. To analyze performance, the authors assess the accuracy and completeness of the reconstructed static map and objects on the KITTI dataset [38] using the provided LiDAR point clouds as ground truth. They show that explicitly separating dynamic parts and the static map leads to more accurate reconstruction for both the static map and dynamic objects.

Like our proposal, Li et al. [69] present a system that performs object-level bundle adjustment. They use the Faster R-CNN [95] object detector to extract 2D bounding boxes around objects. The authors then have a network that infers the viewpoint of the object from the 2D bounding box. From this, they calculate the corresponding 3D bounding box based on a priori dimension assumptions. Li et al. associate object detections to tracks via IoU between the projected 3D bounding boxes of existing tracks and the 2D bounding boxes of detections. Feature points are then extracted for each object detection and the remaining static part of the image. The static map and ego-pose are estimated via PnP and BA. They also formulate each object track as a BA problem and introduce a constant motion assumption to the graph-optimization. Finally, the system aligns the inferred 3D bounding box with the optimized 3D object landmarks to improve their estimate. The resulting system is then evaluated on KITTI [38] and Cityscapes [24]. Similar to the works discussed in Section 4.1, the authors show that their resulting ego-tracking is an improvement over ORB-SLAMv2 [85] in these dynamic environments. Additionally, they show their superior object detections compared to the 3D object detector 3DPO [21] based on the IoU accuracy of birds-eye-view projected 2D bounding boxes. They do not analyze object tracking performance or object detection based on 3D bounding boxes (i.e., the height and elevation of bounding boxes is neglected).

4.3 Metrics for Multi-Object Tracking

As explained in Section 3.1, the task of MOT consists of detecting objects in sequential data (e.g., a stream of RGB or LiDAR frames) and associating detections across frames to form tracks. Generally, there are three types of tasks a tracker must accomplish [65]: first, the system must detect ground truth objects in a frame (True Positive (TP) detections) without making incorrect detections (False Positive (FP) detections) or missing any ground truth objects (False Negative (FN) detections). Second, the tracker must localize the TP detections as accurately as possible, and, finally, the detections must be correctly associated across multiple frames, i.e., grouped into tracks. The first two tracker-qualities demonstrate a systems' detection capability, whereas the third encodes the ability to "connect-the-dots" over time. Thus, evaluating MOT systems requires metrics that rank a tracker along these dimensions.

Principally, there exist two approaches to evaluating a tracker: either by associating *detections* per frame, computing detection performance, and then computing *association* performance; or by associating *entire tracks* and then measuring *association* and *detection* performance. Both approaches constitute an assignment problem that the Hungarian Algorithm explained in Section 3.1 can solve. Hence one needs a similarity score S, either between detections for every frame or between trajectories across an entire sequence, respectively. S should be maximal if ground truth and estimate perfectly overlap and minimal if they are maximally dissimilar. Additionally, typically a minimal similarity α is required for an association to be considered valid.

4.3.1 Matching Techniques

Matching By Detection

When matching by detection ground truth and estimated detections need to be associated for every frame. For 2D tracking, a common similarity measure S is the aforementioned IoU between the 2D bounding boxes or, more precisely, via the respective segmentation masks (see [117] for a discussion on the superiority of segmentation masks over bounding boxes).

A significant problem with IoU is that if there is zero overlap, the measure does not differentiate between close and far away detections (both have a similarity score of 0), although intuitively, a far away detection should be penalized more heavily and vice versa. A remedy for this is GIoU as introduced in [96]. GIoU scores similarity between -1 and 1 (compared to 0 and 1 for regular IoU). This is achieved by first computing the minimal surrounding bounding box C around both bounding boxes A and B and then determining the similarity score as follows:

$$\mathrm{GIoU}(A,B) = \mathrm{IoU}(A,B) - \frac{|C \setminus (A \cup B)|}{|C|} \tag{4.1}$$

Consequently, as A and B move further apart, their area w.r.t. C decreases, and similarity decreases as well, converging towards -1. On the other hand, if A and B are tangent to each other, their computed similarity is 50% of the maximum (compared to 0% as in regular IoU). Both these similarity scores can be straightforwardly extrapolated to 3D bounding boxes (i.e., OOBB): IoU is the intersection volume between boxes A^{3D} and B^{3D}. C^{3D} is the minimal surrounding bounding box necessary for GIoU.

Of course, if a similarity score S is required to be between 0 and 1, normalization can easily accomplish this.

Matching By Track

Matching a set of ground truth tracks to a group of estimated tracks requires a similarity score between tracks. Typically, for a given estimated track T_{est} and a ground truth track T_{gt}, similarity is computed for detections in every overlapping frame and then averaged over the length of the ground truth track. Again, IoU or GIoU can achieve this or, if trajectories for tracks are computed, as the Euclidean distance between points on these trajectories. If a ground truth track is split into two or more estimated tracks by the system, track matching procedures usually only keep the most similar part as a correct association.

4.3.2 CLEAR: MOTA and MOTP

[10] introduced the CLEAR metrics to standardize MOT evaluation. Several tracking benchmarks use them, e.g., KITTI [38] and the MOT-Challenge [27]. The CLEAR metrics match detections (not tracks). The authors define a TP detection as an estimated detection matched to a ground truth detection with a similarity score S smaller than some threshold α. A FP is an estimated detection that couldn't be matched to a ground truth detection; a FN is a ground truth detection for which no corresponding estimated detection exists. Finally, a so-called Identity Switch (IDSW) occurs when a tracker associates two TP detections to different tracks while the corresponding ground truth track ID remains the same or vice versa when the tracker does not switch IDs while the ground truth track does.

From these definitions, the authors then introduce several scoring metrics.

First, Multi-Object Tracking Precision (MOTP), the detection precision of a tracker based on the set of TP detections $\{TP\}$ where $S(c)$ is the calculated similarity measure for a given TP detection.

$$\text{MOTP} = \frac{\sum_{c \in \{TP\}} S(c)}{|\{TP\}|} \tag{4.2}$$

Unlike MOTP, Multi-Object Tracking Accuracy (MOTA) tries to capture the tracking capability of a tracker and is defined as follows (where $\{IDSW\}$ is the set of all IDSWs and $\{GT\}$ is the set of all ground truth detections over all frames):

$$\text{MOTA} = 1 - \frac{|\{FN\}| + |\{FP\}| + |\{IDSW\}|}{|\{GT\}|} \tag{4.3}$$

Other miscellaneous metrics belonging to the CLEAR collection are: Mostly Tracked (MT) and Mostly Lossed (ML) (the number of trajectories where the number of TP exceeds or falls below thresholds $\alpha_{MT} = 0.8$ and $\alpha_{MT} = 0.2$, respectively) and the number of fragmentations (i.e., where tracks are "interrupted" or fragmented by missing detections). Partly Tracked (PT) are trajectories that are neither mostly tracked nor mostly lost.

There exist several problems with MOTA and MOTP. First, they do not take the confidence for a given detection into account. This non-varying confidence leads to the fact that authors of trackers that compute confidences evaluate them using different confidence thresholds to filter detections and consequently maximize MOTA. Thus, [119] suggests an enhancement to MOTA and MOTP that evaluates the metrics over various thresholds and averages them. This enhancement amounts to a discretized integration approximation over confidence thresholds from 0 to 1. Second, similar to confidence thresholds, the evaluation is also very dependent on the similarity threshold α, which determines the minimal required similarity for an estimate-ground truth detection pair to be considered valid. As every non-matched detection counts both as FN and a FP, using a single threshold α can have a detrimental effect on perceived performance. Third, the CLEAR metrics provide no single metric which captures all tracker errors equitably and enables straightforward comparison between different methods. Fourth, MOTA is defined in $(-\infty, 1]$ and not in $[0, 1]$ resulting in unintuitiv results. Finally, the CLEAR metrics value detection over association [77].

4.3.3 IDF1 & Track-mAP

IDF1 [97] was initially introduced for multi-camera settings but has gained in popularity for single-camera trackers [77]. Assuming matched ground truth and estimated trajectories, IDF1 introduces the following measures: Identity True Positive (IDTP), the number of detections belonging to a matched trajectory whose similarity score is below some threshold w.r.t. to the associated ground truth trajectory. Similarly, Identity False Positive (IDFP) are detections of the estimated trajectory which do not have a

corresponding ground truth detection, and, finally, Identity False Negative (IDFN) are missed ground truth detections or estimated detections that exceed the similarity threshold. From this, the ID-Recall, ID-Precision, and IDF1 scores follow:

$$\text{ID-Recall} = \frac{|\{IDTP\}|}{|\{IDTP\}| + |\{IDFN\}|} \tag{4.4}$$

$$\text{ID-Precision} = \frac{|\{IDTP\}|}{|\{IDTP\}| + |\{IDFP\}|} \tag{4.5}$$

$$\text{IDF1} = \frac{|\{IDTP\}|}{|\{IDTP\}| + 0.5|\{IDFN\}| + 0.5|\{IDFP\}|} \tag{4.6}$$

Track-mean Average Precision (mAP) [100], on the other hand, matches tracks based on trajectory similarities (notice that, in contrast to IDF1, detections with low similarity that belong to a TP track are not removed). Matched trajectories result in TP tracks, whereas unmatched trajectories create FP tracks. If there are n TP tracks, then the recall and precision for some value $i \leq n$ are (where the length of the associated ground truth track is denoted as $|\text{GT}|$):

$$\text{Pr}_i = \frac{|\{TP\}_i|}{n} \tag{4.7}$$

$$\text{Re}_i = \frac{|\{TP\}_i|}{|\text{GT}|} \tag{4.8}$$

Track-mAP then interpolates the precision score:

$$\text{InterPr}_i = \max_{j \geq i}(Pr_j) \tag{4.9}$$

The final score is the area-under-the-curve of InterPr_i vs. Re_i.

Contrary to the CLEAR metrics, both IDF1 and Track-mAP overvalue association and undervalue detection (see [77] for a rigorous discussion).

4.3.4 HOTA

To combat many of the flaws of the CV community's current metrics (CLEAR, IDF1, and Track-mAP as mentioned above), Luiten et al. [77] recently suggested an improved measure for MOT which the authors call HOTA. HOTA is a single metric (HOTA) capturing the previously mentioned tracking performance dimensions in equal measures. Like MOTA, HOTA implements a per-frame detection-based association. Although HOTA is a single score, it can be broken down into separate metrics which encode the three dimensions of tracking performance: detection, localization, and association.

The authors also address the problem of the dependency between tracker performance and similarity thresholds. Similar to the approach by [119] which integrates performance over confidence thresholds, one integrates HOTA (and the derivative metrics) over the entire range of similarity thresholds α via discretized approximation.

The first metric (not actually captured by the single HOTA score) measures the localization accuracy based on the similarity scores between a ground truth detection and its associated detection over all TP detections integrated over all similarity thresholds α:

$$\text{LocA} = \int_0^1 \text{LocA}_\alpha d\alpha = \int_0^1 \frac{1}{|\{TP_\alpha\}|} \sum_{c \in \{TP_\alpha\}} S(c) d\alpha \tag{4.10}$$

The second metric measures detection accuracy. Again, integrating over all similarity thresholds, detection accuracy is given as follows:

$$\text{DetA} = \int_0^1 \text{DetA}_\alpha d\alpha = \int_0^1 \frac{|\{TP_\alpha\}|}{|\{TP_\alpha\}| + |\{FN_\alpha\}| + |\{FP_\alpha\}|} d\alpha \tag{4.11}$$

This can be further separated into detection precision and detection recall:

$$\text{DetPr} = \int_0^1 \text{DetPr}_\alpha d\alpha = \int_0^1 \frac{|\{TP_\alpha\}|}{|\{TP_\alpha\}| + |\{FP_\alpha\}|} d\alpha \tag{4.12}$$

$$\text{DetRe} = \int_0^1 \text{DetRe}_\alpha d\alpha = \int_0^1 \frac{|\{TP_\alpha\}|}{|\{TP_\alpha\}| + |\{FN_\alpha\}|} d\alpha \tag{4.13}$$

The introduction of the final submetric, association accuracy, requires an explanation of the novel True Positive Association (TPA), False Negative Association (FNA), and False Positive Association (FPA). These measures are all defined for a single TP detection c. $\text{TPA}(c)$ is the set of TP detections with the same track ID as c and are associated with the same ground truth track as c. Similarly, $\text{FPA}(c)$ is the set of FP detections with the same track ID as c and TP detections with the same track ID as c but which are associated with another ground truth track. Lastly, $\text{FNA}(c)$ is the set of FN detections with the same track ID as c and all TP detection with a different track ID as c but associated to the same ground truth track.

From these three values, one can compute the association accuracy, precision, and recall like so:

$$\text{AssA} = \int_0^1 \text{AssA}_\alpha d\alpha = \int_0^1 \frac{1}{|\{TP_\alpha\}|} \sum_{c \in \{TP_\alpha\}} \frac{\text{TPA}(c)}{\text{TPA}(c) + \text{FNA}(c) + \text{FPA}(c)} d\alpha \tag{4.14}$$

$$\text{AssPr} = \int_0^1 \text{AssPr}_\alpha d\alpha = \int_0^1 \frac{1}{|\{TP_\alpha\}|} \sum_{c \in \{TP_\alpha\}} \frac{\text{TPA}(c)}{\text{TPA}(c) + \text{FPA}(c)} d\alpha \tag{4.15}$$

$$\text{AssRe} = \int_0^1 \text{AssRe}_\alpha d\alpha = \int_0^1 \frac{1}{|\{TP_\alpha\}|} \sum_{c \in \{TP_\alpha\}} \frac{\text{TPA}(c)}{\text{TPA}(c) + \text{FNA}(c)} d\alpha \tag{4.16}$$

The single HOTA metric is then defined as the geometric mean of the detection and association accuracies:

$$\text{HOTA} = \int_0^1 \text{HOTA}_\alpha d\alpha = \int_0^1 \sqrt{\text{DetA}_\alpha \cdot \text{AccA}_\alpha} d\alpha \tag{4.17}$$

The authors of HOTA show that the proposed metric balances association and detection more reasonably than the common de-facto standards CLEAR, IDF1, and Track-mAP, while the sub metrics allow for comparison along the different tracking dimensionalities.

4.4 3D Multi-Object Trackers

Having defined metrics for 2D/3D MOT, we now present an overview of 3D MOT trackers.

Ošep et al. [88] present a MOT which combines 2D and 3D information to produce 3D OOBBs. At each step, the proposed system creates 2D object detections based on [39] and [118] from input images. Concurrently, the system generates 3D object proposals from a pair of stereo frames based on their previous work [89]. 2D and 3D detections are then fused via MAP. The authors associate these so-called observations across time via a novel EKF implementation, which uses both 2D and 3D information in its state. As no 3D MOT benchmark was available, they evaluate their porposal on the KITTI [38] 2D tracking benchmark via the CLEAR metrics. The authors show that performance is comparable to both state-of-the-art MOT for pedestrians and cars. Additionally, the authors give quantitative assessments on the improvements of detection precision when 2D information is enhanced by knowledge of the object in 3D space.

Another vision-based 3D MOT was introduced by Luiten et al. [76]. In this approach, the researchers estimate ego motion (and static map creation) via ORB-SLAMv2 [85]. They detect objects by first creating 2D bounding box detections from an out-of-the-box tracker (both [117] and [94]). They feed these bounding boxes through a custom CNN, which outputs segmentation masks. To associate tracks to detections, the authors warp the mask from the previous frame to the current frame using optical flow and then associate these warped masks to the current detection via IoU and the Hungarian Algorithm. They then fuse stereo-based depth maps and the resulting mask to generate OOBBs and finally merge the dynamic tracks with their dense static map reconstruction to create a time-dependent “4D-semantic-map”. Luiten et al. evaluate their proposal on

KITTI [38] via the 2D MOT CLEAR metrics. They demonstrate superior performance to the previously described CIWT [88] and to the up to that point state-of-the-art 2D tracker BeyondPixels [105].

In [119] Weng et al. propose both a LiDAR-based 3D multi-object tracker and tools to evaluate 3D-enabled CLEAR metrics. Additionally, as mentioned in Section 4.3, the authors present adjustments to the metrics that integrate over all possible confidence scores for object detections. Their 3D tracker is straightforward: they use a point cloud-based object detector [106] and then estimate and update the state of existing object tracks via a Kalman filter. Finally, the researchers match tracks to detections using 3D IoU of their OOBBs combined with the Hungarian Algorithm. The authors show that their simple approach is faster than all existing 2D and 3D trackers. Compared to 2D/3D trackers on the 2D tracking challenge of the KITTI dataset [38], the proposed solution is only slightly outperformed by [105], which is, however, two orders-of-magnitude slower. Weng et al. also compare their solution to the ten-times slower 3D tracker FANTrack [8] on the adjusted 3D CLEAR metrics and demonstrate superior performance.

Very recently, Yin et al. proposed a LiDAR-based 3D multi-object tracker, *CenterPoint*, [125], inspired by the successful 2D tracker *CenterTrack* [130]. The system uses a backbone network that encodes the LiDAR point clouds into $M \times N$ f-dimensional feature vectors. This output is then fed through a custom 2D CNN network, which has several regression heads; the first predicts object locations for k classes w.r.t. the bird-view of a map. Additional regression heads learn the velocity, the dimensions, the height-above-ground, as well as the yaw of the object. Features around a given center prediction are then passed through a second-stage Multi-Layer Perceptron (MLP) classifier, which predicts refined bounding boxes as well as confidence scores. To associate detections to existing tracks, the researchers suggest a straightforward L1-distance-based similarity score where the velocity of the detection for a track from the previous frame is used to estimate its location in the current frame. The authors evaluate their system on nuScenes [18] and the Waymo dataset [112] and compare their system w.r.t. 3D object detection as well as 3D multi-object-tracking. The system exhibits state-of-the-art performance in 3D object detection; it also outperforms [119] and the previous state-of-the-art 3D multi-object-tracker [23] using the 3D CLEAR metrics.

4.5 Conclusion

SLAM systems that treat the world's dynamic and static parts separately are more robust in dynamic environments. Additionally, several approaches have been published

that not only exclude dynamic features of the environment but go further and aim to track these objects over time; however, few of these systems evaluate their tracking performance. Instead, they exhibit improved ego-tracking and sometimes show their 3D *detection* capabilities. However, 3D MOT is a highly active research area, although often decoupled from SLAM. Several vision-based 3D trackers exist; however, they are only evaluated on 2D tracking, unable to demonstrate their 3D tracking ability. [119] introduced a method to extend the popular CLEAR metrics to 3D evaluation, and quite a few LiDAR-based trackers have emerged with impressive performance. However, [77] shows that the de-facto standards in 2D/3D MOT evaluation, the CLEAR, IDF1, and Track-mAP metrics, suboptimally evaluate MOT performance. Due to its very recent publication, HOTA has not been adopted as a standard metric in common benchmarks.

5 Method

The goal of MOT in 3D is to detect objects in sequential sensor frames (e.g., RGB images or LiDAR scans) and associate the resulting detections over time to create object tracks.

We use three types of coordinate systems:

1. the static world coordinate system through which the camera (ego vehicle) and other objects move
2. the ego poses at time t w.r.t. the world frame: ${}^{w(orld)}_{c(am)}\mathbf{T}_t$
3. the object poses at time t for object i w.r.t. the world frame: ${}^{w}_{o(bject)_i}\mathbf{T}_t$

The world coordinate system is initialized as the first camera pose at time $t = 0$, i.e. ${}^{w}_{c}\mathbf{T}_0 = \mathbf{I}^4$. Estimating the camera poses is not part of the task at hand s.t. these poses are (alongside the stereo camera frames) the input to the system (coming, e.g., from a separate visual odometry system). The coordinate systems' orientation is based on the KITTI dataset, as this is the basis for evaluation. A visualization of the coordinate systems is given in Fig. 5.1.

As mentioned in Section 3.1, a 3D object is typically expressed by an OOBB This OOBB comprises the object location ${}^{w}\mathbf{t}$ in world coordinates, the object's dimensions (height, width, length) and its yaw (i.e., rotation around its vertical y-axis). We give a visualization of such an OOBB in Fig. 5.2. We estimate the OOBB for each time step. Connecting the object's locations over time forms a 3D trajectory.

5.1 System Overview

The proposed method for 3D MOT, termed BAMOT, combines several techniques from related problems: 2D Object Detection (OD) and tracking (see Section 3.1), and visual SLAM (see Section 3.5). The suggested overarching system is a composition of the

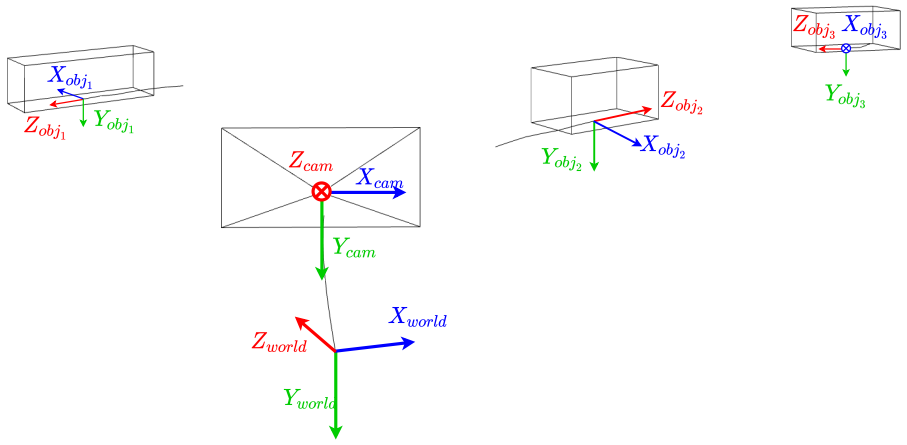


Figure 5.1: An illustration of the different coordinate systems/poses at a given time t. The world frame is initialized to the first camera pose and kept fixed. The other poses are updated every frame.

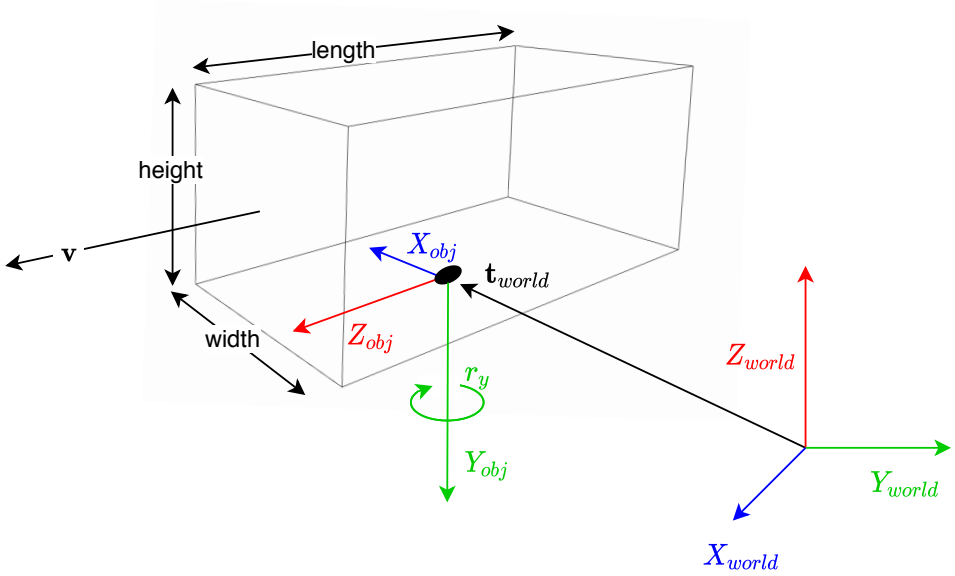


Figure 5.2: The OOBB typically describes objects for the 3D MOT task. Note that this state simplification does not allow a pitch rotation and hence assumes level roads.

following functionalities:

1. stereo multi-object detection
2. detection-track association
3. 3D object tracking

The division of BAMOT into these loosely-coupled subsystems allows easy interchangeability of all subsystems with updated procedures, e.g., an improved 2D object detector, a different method for associating tracks with detections, or other feature point extraction or matching techniques. The boundaries between the subsystems allow these types of changes and facilitate incremental development and improvements along any of these three dimensions. A rough qualitative overview of the proposed system is given in Fig. 5.3.

Note that the sensors used as input to the system potentially create further interdependencies and requirements for the subsystems.

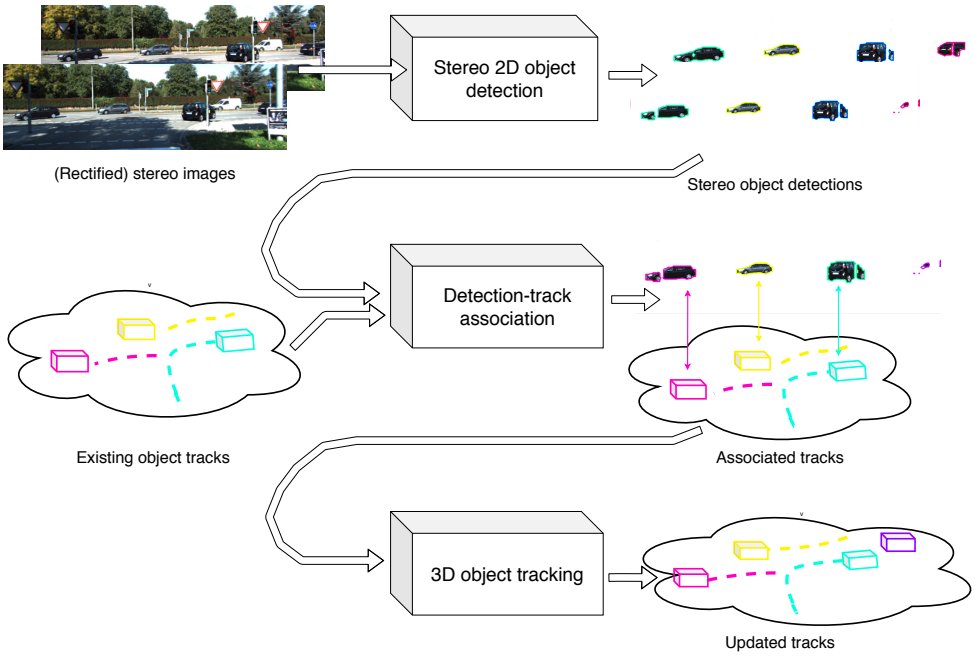


Figure 5.3: A coarse sketch of BAMOT. The proposed process consists of three steps: 2D stereo object detection, detection-track association, and 3D object tracking.

5.2 Sensors

In this work, two stereo-calibrated RGB cameras are the sensory input in form of a stream of rectified images. This rectification is not principally necessary: one can also achieve 2D object detection, feature point extraction and matching, and stereo triangulation using non-rectified (but calibrated) cameras. However, we evaluate this thesis on the popular KITTI dataset (see Section 6.1), which provides rectified images.

5.3 Stereo 2D Multiple Object Detection

The first step in BAMOT is detecting objects from both stereo images, I_l, and I_r, in 2D and associating left and right object detections. The resulting detections are now 2D stereo object detections. The object detector used in BAMOT is the deep-learning-based TrackR-CNN introduced in [117] which extends MaskR-CNN [47] with association capabilities. This 2D tracker outputs pixel-wise segmentation masks and object classes per detection. Both left and right images pass through the detection network. The result is two collections of segmentation masks, M_l^i and M_r^j, of sizes s_l and s_r, respectively.

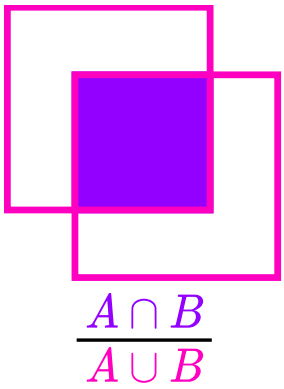


Figure 5.4: The Jaccard Index comes from set theory, but one often uses it as a similarity measure in object detection and association tasks. In this context, it is more commonly known as the Intersection-Over-Union measure.

Since the left and right masks are in different image domains and there is no way to transform a mask from the left image onto the right image without any depth information, a simple area-based similarity score such as the widely-used Jaccard Index [53] or, its more common name in CV tasks, Intersection-over-Union (see Fig. 5.4), will produce erroneous associations, especially with many neighboring objects (see Fig. 5.5 as a qualitative example).

Thus, we use an appearance-based heuristic in addition to the IoU score. Feature points (see Section 3.3) encode appearance and we already use these in the subsequent track-detection association and 3D multi-object tracking steps. Hence, we exploit this redundancy by using such features in multiple parts of the system and caching results in between steps. We compute the appearance score as follows: first, we detect ORB [99] features in both images. This results in two sets of feature points f_l and f_r with sizes n_l and n_r, respectively. Next, we match these features as described in Section 3.3. This then leads to to n_m stereo matches. Finally, the "normalized" matched features score λ_f is:

$$\lambda_f = \frac{n_m}{\max(1, \min(n_l, n_r))} \tag{5.1}$$

This score is zero if there are no feature matches and maximized to 1 if at least one collection of features (left or right) is matched perfectly (unless there are zero matches in either the left or right image). This score puts more weight on the number of matched features per object than on the number of matched features in total and allows to match objects that are occluded or truncated in one image but not (or less so) in the other. Putting more weight on the total number of matches would, for example, result in Eq. (5.2). Note that as an additional measure, detections with different class predictions receive a score of 0.

$$\lambda_f = \frac{n_m}{\max(1, \max(n_l, n_r))} \tag{5.2}$$

Using the former equation vs. the latter empirically proved to result in superior associations. The resulting final score is in the range of $[0, 2]$:

$$\lambda = \lambda_f + IoU(M_l^i, M_r^j) \text{ with } IoU(a, b) = \frac{|a \cap b|}{|a \cup b|} \tag{5.3}$$

After association, there may still be unmatched detections in both the left and the right image. To keep these potentially correct detections, we "transform" the segmentation mask of an unmatched detection in the left or right image to the respective other image. This transformation is simply a dilation of the mask followed by a removal of pixels that they are already part of a different detection. The resulting set of masks are thus non-overlapping. The introduced system dilates the mask by a dynamic number of pixels. This amount is inversely proportional to the mask's size m_s. The notion behind this heuristic is that as the object becomes smaller, the likelihood of correctly "hitting" parts of the object in the other image becomes smaller and vice versa. Setting the proportionality factor to 2% of the image area $A = h \cdot w$ works well empirically. Additionally, we set the dilation to be at least 1px and at most 5% of the smaller image

Figure 5.5: *Left*: Left-right object association is done using the similarity measure from Eq. (5.3). *Right*: Using only IoU as a similarity measure results in bad associations as areas from different image domains cannot be compared. Note that in this qualitative example unmatched detections that are "extrapolated" into the other image domain are left out for simplicity.

dimension $d_s = \min(h, w)$. Again, we determined these values experimentally. The resulting number of pixels p_d the system dialates mask is then:

$$p_d = \min(\lfloor 0.05 \cdot d_s \rfloor, \max(1, \lfloor \frac{0.02 \cdot h \cdot w}{m_s} \rfloor))) \tag{5.4}$$

If the dilated mask is empty (because other detections fully occlude it), the detection remains unmatched, and the system subsequently discards it. A flow diagram of the 2D stereo object detection process is given in Fig. 5.6.

5.4 Detection-Track Association

As the name suggests and as previously mentioned, the object detector we employ in Section 5.3, TrackR-CNN, is in fact a two-stage 2D multi-object tracker that extends the 2D multi-object detector MaskR-CNN. In its first stage, TrackR-CNN detects segmentation masks and encodes the appearance of each object. The association step of TrackR-CNN happens in the second-stage of the network. In this stage, the network learns to associate the encoded appearance vectors of the detected objects from the first stage. For many cases, this works well; however, several scenarios exist where this process fails, e.g., for objects that look similar (resulting in false positives) or for those that change their subjective appearance resulting from a substantially different viewing angle (false negatives). Another error source is partial (or complete) occlusions and truncations and their effect on the encoded appearance.

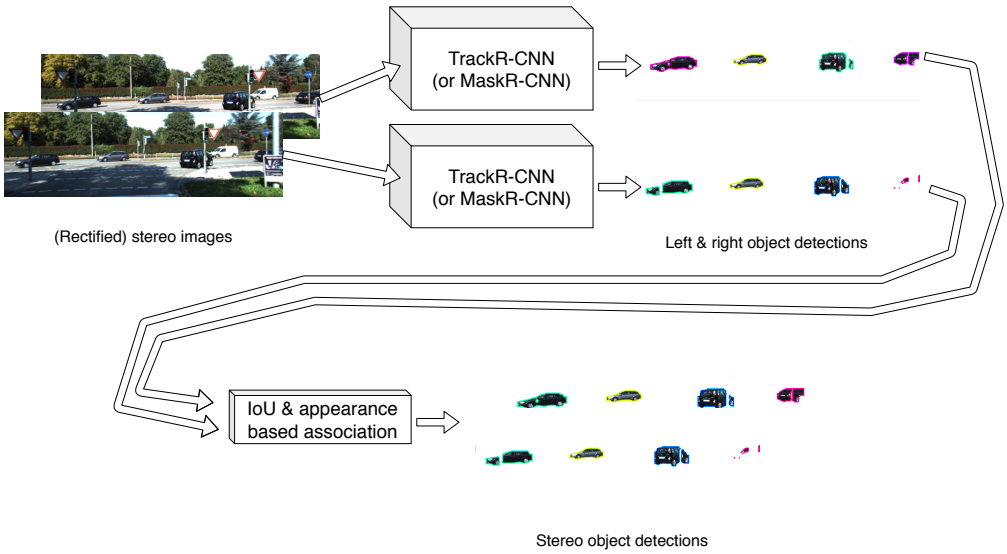


Figure 5.6: First, left and right images pass through a 2D object detector. Then the system associates the resulting detections between the two images using a similarity measure consisting of a proposed combination of the classical IoU and a "normalized" matched feature ratio score.

Additionally, as the network learns this association, it adds another non-explainable component to possibly safety-critical applications (unlike, say, associations based on IoU). Fortunately, our proposed system has additional 3D information that it can utilize to supplement or altogether replace the learned associations from TrackR-CNN. The proposed detection-track association consists of several consecutive association steps:

1. association based on the combination of appearance and 3D location
2. 3D corroborated 2D association (optional, only if the object detector is an object tracker such as TrackR-CNN)
3. association using only 3D location

Note that the proposed system discards track-detection combinations for the first and second step where the track's object class does not match that of the detection. Additionally, we remove out-of-view tracks: track for which no landmarks can be projected to valid left image coordinates.

5.4.1 Association via Combined Appearance & Location

Tracking based either entirely on appearance or the location (both 2D and 3D) of objects is sub-optimal. Appearance changes or is non-unique (to a certain degree), projected 2D positions (e.g., masks or bounding boxes) lose valuable depth information, and both 2D and 3D locations can become ambiguous when objects appear close to one another (especially at greater distances). Therefore, combining these two similarity measures is desirable as it presumably leads to a more robust association less susceptible to either measure's shortcomings. One approach to a unified measure would be to calculate the similarity of appearance and location separately and then merge these two scores in some fashion. However, this merging into a single similarity measure for an association is not straightforward and requires some underlying parameterized model. BAMOT uses such a scheme for stereo object association (see Section 5.3) in 2D which works sufficiently well in practice; however, in the case at hand, associations are not merely snapshot-based (i.e., objects only have to be matched between a single left and a single right image) but detections have to be robustly re-matched to tracks that have potentially become occluded for several frames (and conversely, detections of new tracks should not wrongly be associated to existing tracks which have become occluded). Hence, it is preferable to use a measure that incorporates appearance and 3D location "out-of-the-box".

One such procedure is PnP (see Section 3.4.5): PnP estimates a transformation given a set of 2D-3D correspondences. Typically (i.e., in Visual Odometry (VO) or SLAM), the 3D points are landmarks in a static map, the 2D points are observations of these points on an image plane of a camera, and the resulting transformation is the camera pose w.r.t. the static map. In the proposed implementation, the 3D points are landmarks of an object w.r.t. that object's coordinate frame, and the resulting transformation is the object pose w.r.t. the fixed world frame. The reasoning behind adopting PnP (over, say point cloud alignment via 3D-3D correspondences using, e.g., ICP [22]) is that we already employ this procedure in the following 3D object tracking step. See Section 5.5 for a full description of the adjusted PnP method. Hence, we can reuse the computed poses. PnP uses appearance (feature points-landmark matches) and 3D information (object landmarks and the resulting pose ${}^{cam}_{o}T_t \in \mathbb{R}^3$). As mentioned above, we cache the feature points per object detection from the previous 2D stereo object detection step. We match these with the landmarks of each existing track. The following Section on 3D object tracking (Section 5.5) explains our features-landmark matching method.

PnP doesn't directly result in a similarity measure that the system can employ for associations. However, it does output a normalized ratio of success: the inlier ratio $\lambda_i = \frac{n_i}{n}$ from the underlying RANSAC scheme (see Section 3.4.4) where n_i is the number of inliers, and n is the total number of points used in the estimation (i.e., the number of

2D-3D matches). Nonetheless, we cannot use the inlier ratio directly as a measure of similarity: take, as a concrete example, an object-track combination with five 2D-3D correspondences and an inlier ratio of 1.0 vs. an object-track combination with 100 2D-3D matches and an inlier ratio of 0.99. Adding a third object-track combination with 200 2D-3D correspondences and an inlier ratio of 0.5 illustrates that using the raw number of inliers also doesn't work as a proxy for similarity. Thus, we need to normalize the number of inliers in some other way. Two factors limit the maximum number of inliers n_{max}, or, equivalently, the maximum number of 2D-3D correspondences. First, the number of landmarks n_l of an object, and second, the number of features n_f in a detection. Consequently, $n_{max} = \min(n_l, n_f)$. This allows for both occluded objects (n_l will be high, but n_f small) and objects coming into view (n_l will be low, but n_f high). Thus, the resulting similarity measure is:

$$\lambda = \frac{n_i}{n_{max}} \tag{5.5}$$

If the pose estimation fails, we set the measure to 0.

As an additional criterion to check the success of the pose estimation, our system assesses the validity of the translational component. We consider a pose estimate valid if the relative object translation w.r.t. the previous pose at time $t-1$ is within a certain distance d_{max}:

$$|\mathbf{t}_{rel}| <= d_{max} \text{ with } ({}^{c}_{o}T_t)^{-1}\,{}^{c}_{o}T_{t-1} = {}^{o_t}_{o_{t-1}}T = \begin{bmatrix} \mathbf{R}_{rel} & \mathbf{t}_{rel} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{5.6}$$

If this "motion" is deemed invalid, we set the similarity measure to 0.

Three parameters influence the heuristical function behind d_{max}:

1. the object type (i.e., a pedestrian is slower than a car)
2. the distance of the object from the camera d_c (i.e., the accuracy of pose estimation decreases with distance to the camera)
3. the number of consecutive frames that BAMOT was not able to associate a track to a detection (see the following Section 5.5)

Each object-type i has a fixed maximum expected speed of v_i. This speed multiplied by the frame rate T gives a baseline maximum translation $\tilde{d}_{max} = T \cdot v_i$. This baseline defines the radius of a sphere around the previous location that an object is expected to maximally travel between two frames. We multiply this radius by a factor which conceptually acts as a measure of uncertainty (as determined by parameters two and three of the preceding listing) of the previous location, e.g., if the ambiguity of the last

position is high, this factor is greater than 1. The uncertainty arising from the distance to the camera d_c stems from fewer features detected on a smaller object (hence the pose estimation can use fewer 2D-3D associations as well). Additionally, triangulation accuracy decreases with distance. The authors of ORB-SLAM [84, 85] empirically determined that up until a distance of ~20 times the baseline b of the stereo cameras (see Section 3.4.2) triangulation is accurate. For this reason the distance component f_d is as given in Eq. (5.7) where it is 1 for distances beneath the threshold $d_b = 20 \cdot b$ and the distance divided by d_b for greater distances.

$$f_d(c_d) = \begin{cases} 1, \text{ if } d_c < d_b \\ \frac{d_c}{d_b}, \text{ else} \end{cases} \tag{5.7}$$

The third parameter, the number of frames n_{bad} for which no 2D stereo detections exist, is another source of uncertainty of the object location. BAMOT incorporates this into the final multiplicative factor by multiplying n_{bad} by a factor f_{bad} and adding one to the fraction (s.t. the multiplier doesn't become zero when the object's location uncertainty is due to its distance from the camera and not due to missing detections). Additionally, the multiplicative factor is limited by a constant f_{max} to restrict the possible locations and avoid false-positive associations. This cutoff essentially gives an upper bound to the maximally acceptable uncertainty of an object's location when associating it to detections. Overall, the resulting allowed distance is:

$$d_{max} = f \cdot \tilde{d}_{max} = \min(f_{max}, f_d(c_d) \cdot (f_{bad} \cdot n_{bad} + 1)) \cdot T \cdot v_i \tag{5.8}$$

A complete flow diagram of this process is presented in Fig. 5.7.

5.4.2 3D Corroborated 2D Assocations

This next step is employed when the object detector is, in fact, a 2D object tracker such as TrackR-CNN. The idea behind this association step is to supplement the given 2D information (by the 2D tracker) with the available 3D information. We achieve this merging by corroborating (or discarding) 2D associations with the 3D location of a given detection and the associated object track as follows: First, for every stereo detection, our system uses the stereo feature matches from the stereo detection association step (see Section 5.3) to localize the object in 3D space. The proposed system accomplishes this localization by triangulating each stereo feature (see Section 3.4.3) to compute a point cloud for a given detection. If we cannot successfully triangulate any points (and, hence, have no 3D information), the 2D association is discarded. Otherwise, the median point of the resulting point cloud is the location estimate. The median is more robust to outliers s.t. the accuracy of the resulting location will not suffer from few badly

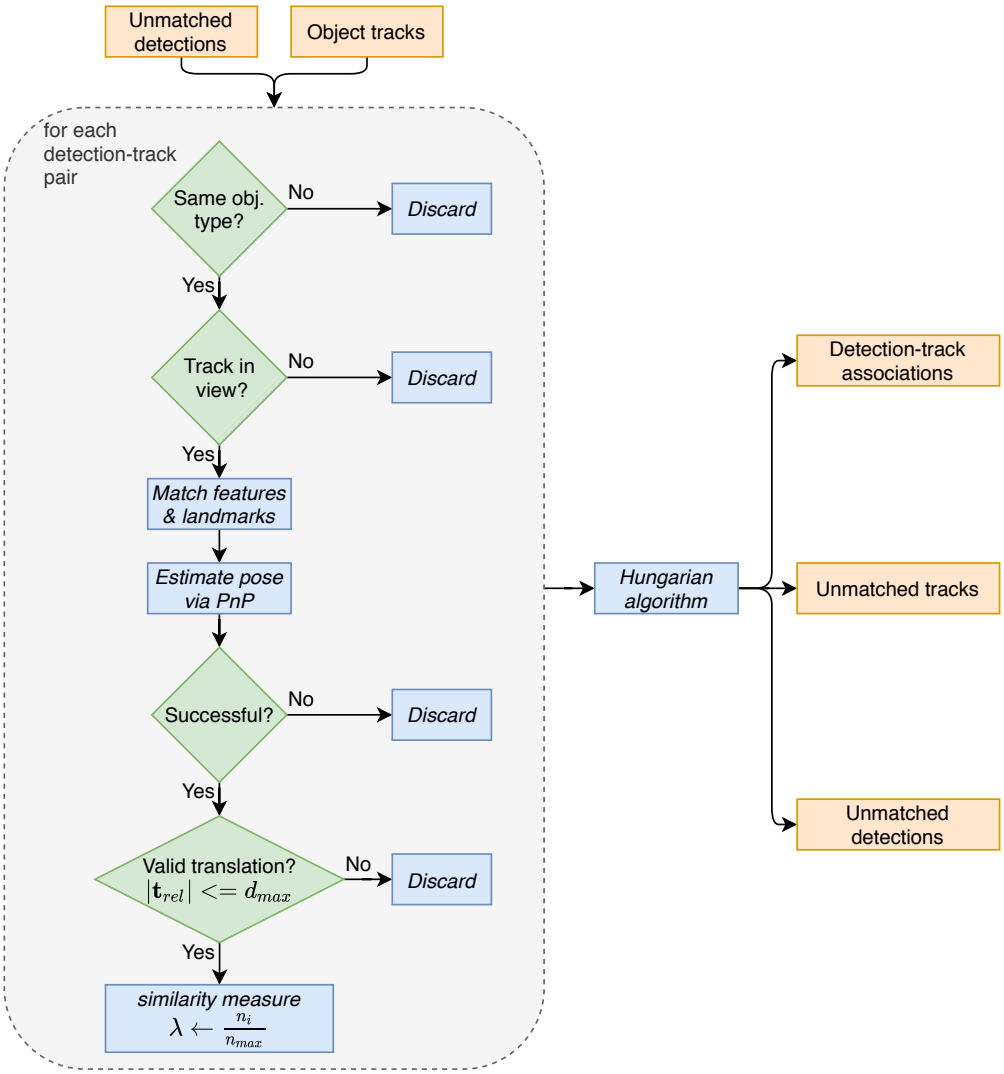


Figure 5.7: A flowchart of the detection-track association step based on PnP as described in Section 5.4.1.

triangulated (because badly matched) features. We then check whether the relative translation between the current location $\mathbf{t}_0$ and the estimated location $\mathbf{t}_{-1}$ from the previous frame (see Section 5.5) is valid:

$$|\mathbf{t}_{rel}| = |\mathbf{t} - \mathbf{t}_{-1}| \leq d_{max} \tag{5.9}$$

The preceding association step does an equivalent validation when it verifies the PnP-estimated pose's resulting translational component. If the translation is acceptable, we adopt the association emanating from the 2D tracker. Lastly, if the 2D tracker initializes a new track, we respect this, and the system creates a new track. Again, we provide a flowchart of this process in Fig. 5.8.

5.4.3 Association via 3D location

The association pipeline's final step is to associate tracks and detections using only their respective locations in 3D space. For each detection-track combination, the similarity measure used by the Hungarian algorithm (see Section 3.1.3) is the 3D distance between the estimated track and the detection. We estimate the location for a given detection via its point cloud's median point. As in the previous step, we create the point cloud by triangulating the stereo matches. In line with the preceding stages, the distance between a detection-track pair needs to fall within a threshold d_{max} (see Eq. (5.8)) to be considered a successful association. Fig. 5.9 shows an overview of this final phase.

BAMOT creates new tracks from detections that remain unmatched after this three-step association pipeline.

5.5 3D Object Tracking

The proposed 3D object tracking amounts to object-level SLAM with modifications rooted in a priori knowledge of the object types' geometry and dynamics. Landmarks of a given object are defined w.r.t. its coordinate system, i.e., its current pose. Every landmark can have multiple observations, i.e., 2D feature points matched to this landmark. For each frame, we estimate an object's pose w.r.t. the world frame. Initially (i.e., when BAMOT initializes an object track), the object pose coincides with the current camera pose. Additionally, each object has a location associated with it. As in the association pipeline, an object's location is its median landmark. This location is always given in the current frame at time t w.r.t. the world coordinate frame. The previous locations make up an object's trajectory and we use it to compute an object's OOBB.

Three track categories exist after associating 2D detections to existing tracks: unmatched tracks, matched tracks, and new tracks.

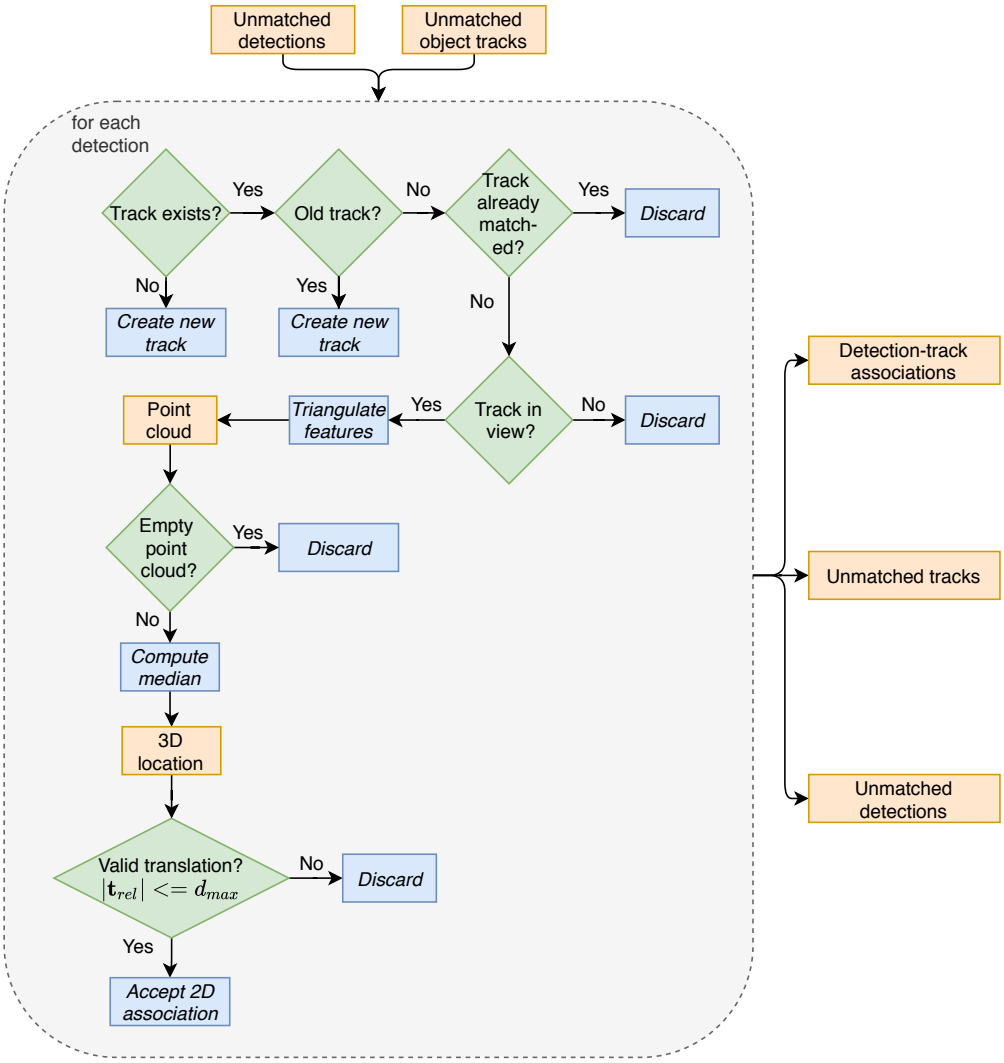


Figure 5.8: If the 2D object detector is a tracker (such as TrackR-CNN), the second step in the association method corroborates the 2D associations using 3D information. This flow diagram depicts this procedure.

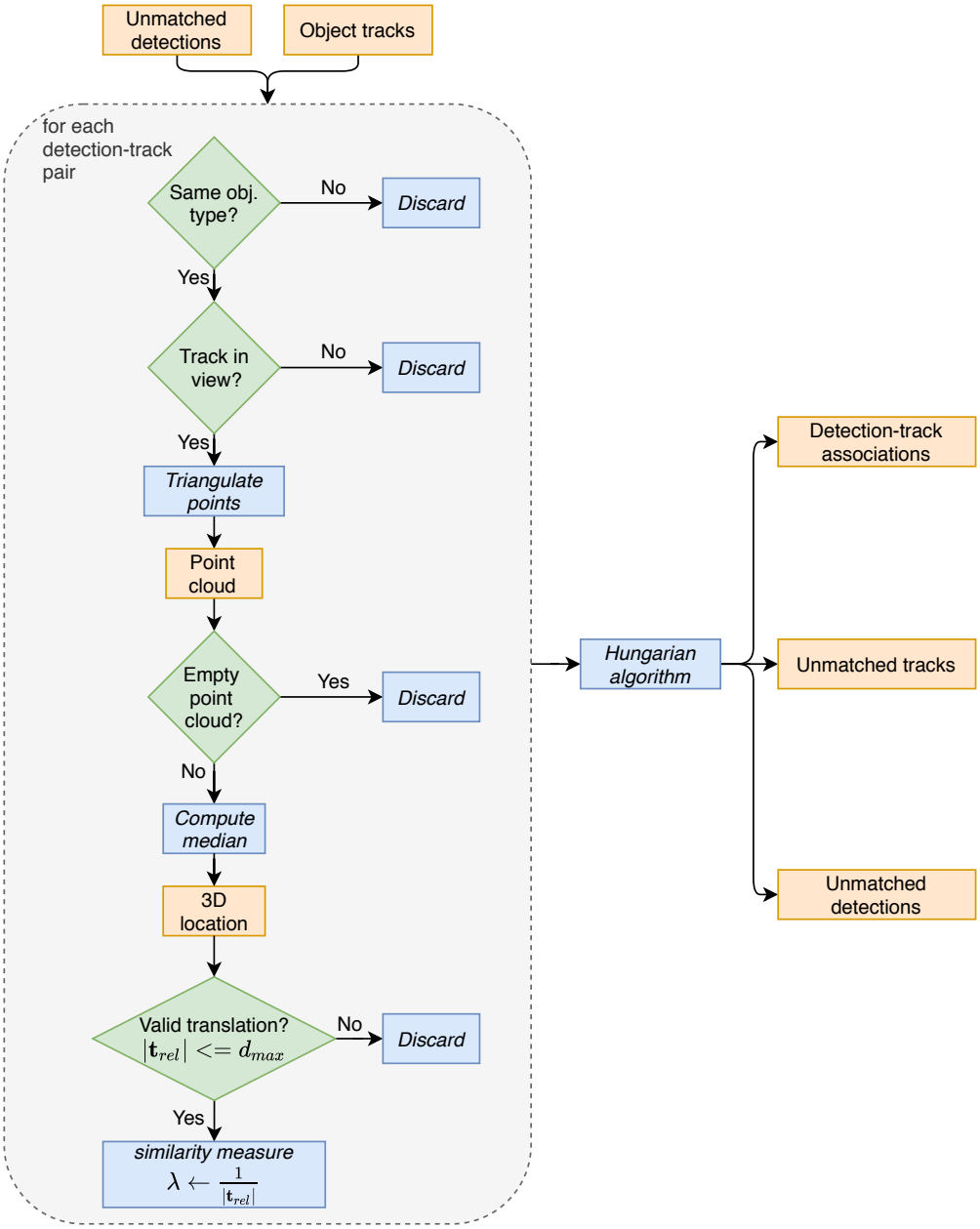


Figure 5.9: The final stage in the association pipeline: detections and tracks are associated solely by their similarity in 3D space. Note that BAMOT already performs some of these steps (e.g., creating point clouds) in a previous stage. However, the steps are shown here for completeness.

5.5.1 Extrapolating Motion For Unmatched Tracks

For unmatched tracks, i.e., tracks with no corresponding 2D detection, BAMOT extrapolates the locations. This extrapolation rests on a constant (local) motion assumption across a span of n_{rel} frames. That is, for a track with n_p poses we take the previous poses ${}^{w}_{o}\mathbf{T}_{t-1}$ and ${}^{w}_{o}\mathbf{T}_{t_0}$ from time $t-1$ and $t_0 = t - \min(n_{rel}, n_p)$, and then determine the average relative translation between these:

$$\bar{\mathbf{t}}_{rel} = \frac{\mathbf{t}_{t-1} - \mathbf{t}_{t_0}}{\min(n_{rel}, n_p)} \tag{5.10}$$

We then compute the next pose ${}^{w}_{o}T_t$ by keeping the rotational component ${}^{w}_{o}\mathbf{R}_{t-1}$ equivalent and adding $\bar{\mathbf{t}}_{rel}$ to the previous translational vector $\mathbf{t}_{t-1}$:

$${}^{w}_{o}\mathbf{T}_t = \begin{pmatrix} {}^{w}_{o}\mathbf{R}_{t-1} & (\mathbf{t}_{t-1} + \bar{\mathbf{t}}_{rel}) \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{5.11}$$

For tracks with only a single pose we have no translational estimate so the extrapolated pose is kept equivalent to the previous pose (${}^{w}_{o}\mathbf{T}_t = {}^{w}_{o}\mathbf{T}_{t-1}$). Additionally, we consider these unmatched tracks as poorly tracked and record the number of consecutively poorly tracked frames per object. If this counter goes above a certain threshold t_{bad}, we consider the track inactive, remove it from future association steps, and do not record its OOBB.

5.5.2 Object Pose Estimation

The tracking procedure is broadly equivalent for new and matched tracks and we henceforth describe it for a single track. First, the system resets the badly tracked frames counter to zero. Then it estimates the current pose. For matched tracks, the pose extrapolation procedure explained in Section 5.4 is the initial pose guess. We then refine this guess using PnP estimation in a RANSAC scheme (see Section 3.4.5 and Section 3.4.4, respectively). However, we reframe the typical PnP problem as we are interested in the object and not in the camera motion. Note that our system assumes the camera pose ${}^{w}_{c}\mathbf{P}_t$ to be given. In addition to the camera pose, we require the feature-landmark matches for this. These are available from the preceding detection-track association step (see Section 5.4). Also, notice that as the system performs four-point PnP (and not, e.g., the underdetermined P3P [122]), our initial pose guess remains unchanged if there are fewer than four matches. Usually, PnP estimates the camera pose w.r.t. landmarks of a static map. However, in this application, each object is its own static map, i.e., landmarks are in object coordinates. The poses that BA optimizes are the relative poses between the camera and the object. Eq. (5.12) gives the corresponding optimization objective of minimizing the reprojection loss function.

$$\min_{{}^{c}_{o}\mathbf{T}} \sum_{i}^{n} ||\mathbf{p}_i - \pi({}^{c}_{o}\mathbf{T}\,{}^{o}\mathbf{p}_i)||_2^2 \tag{5.12}$$

If PnP successfully estimates a transformation, and the inlier ratio is greater than λ_{pnp}, the system re-runs the procedure on the entire inlier set from the initial RANSAC optimization. Additionally, as in the first stage of the detection-track association procedure (see Section 5.4), the translational component of the optimized pose is verified to be less than d_{max} from Eq. (5.8). If the optimization is unsuccessful, has an inlier ratio below λ_{pnp}, or the relative translation is too great, the initial guess (extrapolated pose) remains unchanged.

5.5.3 Adding Landmarks and Observations

We utilize stereo matches (see Section 5.3) and interest point-landmark matches (see Section 5.4) to create new landmarks of an object and add observations of existing landmarks, respectively. As explained in Section 3.3, we require a descriptor for each landmark to perform descriptor matching between landmarks and (left) features. Our descriptor calculation for a landmark with n observations is inspired by ORB-SLAM [84, 85]: for each observation, we compute the distance of its descriptor to all other observations. Then, we determine the median of those distances per observation. Finally, we choose the observation's descriptor with the smallest median distance to all other observations as the landmark's representative descriptor. Unlike ORB-SLAM, there is no concept of "keyframes": BAMOT considers all frames for a given track. Consequently, landmarks may have many more observations than in a SLAM setting. Since the descriptor computation mentioned above has a runtime of $O(n^2)$, the proposed system limits the number of descriptors used for comparison $\tilde{n}$ to n_{max}. We sample the $\tilde{n} \leq n_{max}$ descriptors uniformly from all available n observations. Once (left) features have been matched to landmarks, we can add the features as observations. However, these features must fulfill two criteria: first, when transforming the matched object landmark to the current camera pose, the point must not be behind the camera, and, second, its distance must not exceed a given threshold d^c_{max}. The system adds a stereo observation of the landmark if there exist a corresponding right feature point (i.e. the feature is also a stereo match) and if the left-right feature correspondence does not violate the epipolar constraint (as detailed in Section 3.4.2).

After the preceding step, the system triangulates new landmarks (see Section 3.4.3) from stereo matches (feature-feature matches from the left and right camera frames) that it did not yet add as observations to existing landmarks in the preceding step. Several conditions must hold for these newly created landmarks to be valid as well: they need to fulfill the epipolar constraint, the triangulated point must not lie behind

the cameras, and finally, the distance of the point to the cameras may not exceed the threshold of d^c_{max}. Fig. 5.10 shows the process of adding observations and landmarks.

The set of newly created landmarks and observed landmarks, termed the current landmarks, are then used to remove stale landmarks in the following manner: first, the center of the current landmarks is their median. Then, we mark all landmarks that are further from this center than an object-specific threshold as invalid and consequently delete them. This object-specific threshold is the maximum expected object dimension for a given type (e.g., the length for cars and height for pedestrians). Such a threshold ensures that even if the current landmarks are at the very edge of an object, the process does not remove existing landmarks at the object's opposing edge. This thresholding-based culling process proves to be a simple and powerful way to remove outlier landmarks based on known object dimensions. This is a deviation from regular SLAM systems as they do not have such geometric knowledge of the static map a priori.

As mentioned above, we set the object's initial pose to the current camera pose. However, once the object contains landmarks, we translate the coordinate system, so its origin lies at the median of the object's point cloud. This translation is necessary to limit the effect of camera rotation on motion estimates for far away objects. Finally, if the system did not create more than l_{min} landmarks, it discards the newly created track. This conservative track creation ensures more robust initialization at the potential cost of later than necessary initialization.

5.5.4 Object-level Bundle Adjustment

The classical SLAM problem consists of simultaneously finding the world positions of a collection of landmarks and the world poses of a set of camera poses where each pose observes some subset of the landmarks. One approach to solve this problem is through Bundle Adjustment (see Section 3.5), a graph-based solution.

In the proposed system, each object k has n_k landmarks that are fixed w.r.t. the object's coordinate frame under the rigidity assumption. Also, each object has t_k pose estimates w.r.t. the fixed world frame. In order to simultaneously optimize the landmark positions and the object poses, we need to make some mathematical adjustments s.t. the problem can be solved via Bundle Adjustment. Since the pose of the camera at time t is given as ${}^{w}_{c}\mathbf{T}_t$, estimating the relative pose between the camera and the object k at time t ${}^{o_k}_{c}\mathbf{T}_t$ results in the desired pose:

$$ {}^{w}_{o_k}\mathbf{T}_t = {}^{w}_{c}\mathbf{T}_t \left({}^{o_k}_{c}\mathbf{T}_t\right)^{-1} \tag{5.13} $$

Thus, we introduce a Bundle Adjustment scheme where both landmarks and camera poses are optimized w.r.t. the object frame. Since the camera poses w.r.t. the world frame are known, the system can deduce the needed object poses and landmarks w.r.t.

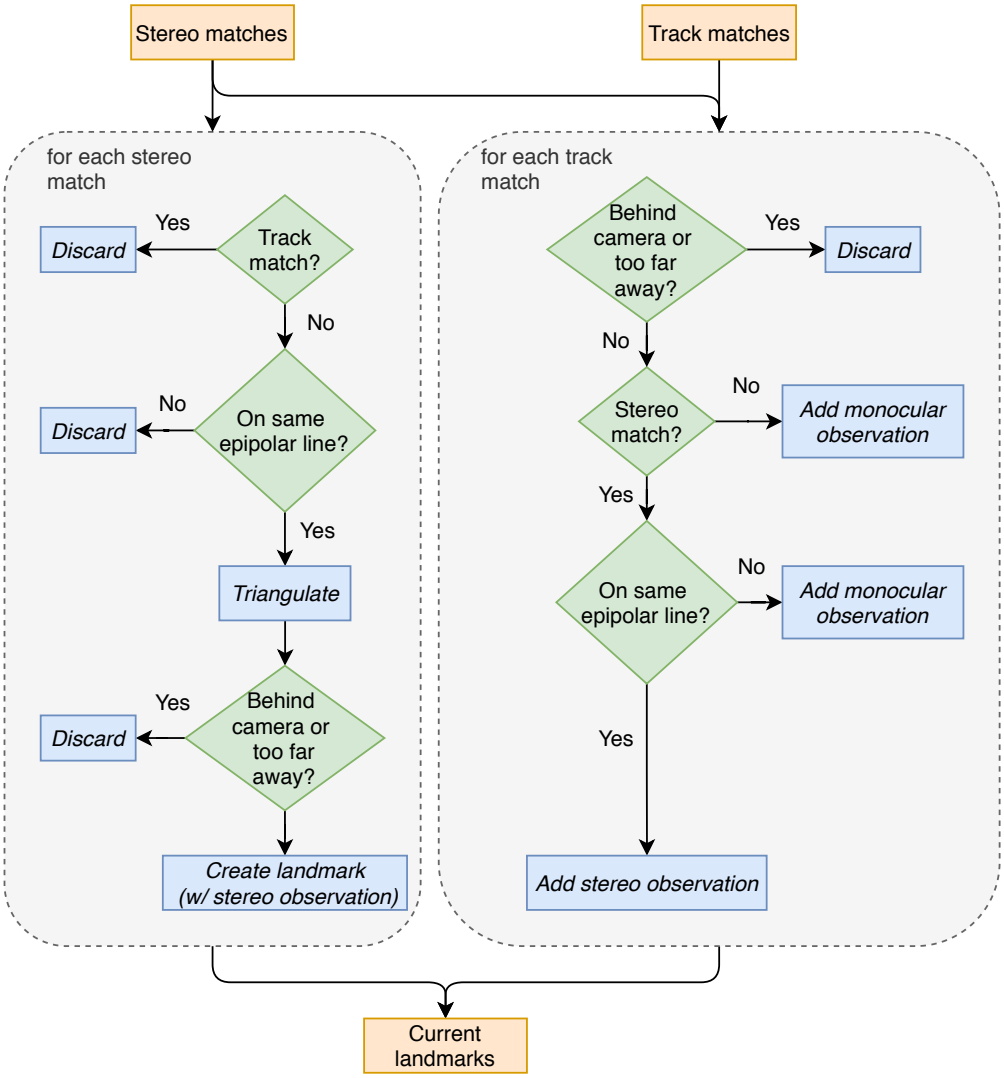


Figure 5.10: The qualitative process of adding new observations to existing landmarks from feature matches between the left camera and landmarks and creating new landmarks from matches between the left and right camera frames. Section 5.5.3 explains the procedure in detail.

the world frame. Eq. (5.14) displays the adjusted objective function for a single object k with n landmarks tracked over t frames with known camera poses over these frames. Note that $\pi(.)$ is the known projection function, p_t^n is the observed feature point for the landmark n at frame t, and $\rho(.)$ is the robust Huber kernel [51]. The Huber kernel limits outliers' influence: their residual is linear instead of quadratic.

$$\min \sum_{n,t} \rho(||\mathbf{p}_t^n - \pi({}_c^o\mathbf{T}^{to}\mathbf{P}_n)||^2) \tag{5.14}$$

We further adjust this objective function via the addition of an error term that penalizes non-smooth trajectories. The idea is that objects are expected to have constant velocity, at least locally, and further that adding this requirement has a positive regularizing effect on the optimization problem. We express this constant motion requirement as follows: for three poses at times t, $t-1$, and $t-2$, the relative object motion from $t-2$ to $t-1$ and from $t-1$ to t must be identical. The equality from Eq. (5.15) shows that the necessary poses are ${}_c^o\mathbf{T}_{\tilde{t}}$ and ${}_c^w\mathbf{T}_{\tilde{t}}$ for $\tilde{t} \in \{t-2, t-1, t\}$. ${}_c^o\mathbf{T}_{\tilde{t}}$ are the poses that the BA procedure optimizes and ${}_c^w\mathbf{T}_{\tilde{t}}$ are the known camera poses that we use as constants in the graph-optimization.

$$\begin{aligned}
{}_{o^{t-2}}^{o^{t-1}}\mathbf{T} &= {}_{o^{t-1}}^{o^t}\mathbf{T} \\
{}_w^o\mathbf{T}_{t-1}({}_w^o\mathbf{T}_{t-2})^{-1} &= {}_w^o\mathbf{T}_t({}_w^o\mathbf{T}_{t-1})^{-1} \\
{}_c^o\mathbf{T}_{t-1}({}_c^w\mathbf{T}_{t-1})^{-1}({}_c^o\mathbf{T}_{t-2}({}_c^w\mathbf{T}_{t-2})^{-1})^{-1} &= {}_c^o\mathbf{T}_t({}_c^w\mathbf{T}_t)^{-1}({}_c^o\mathbf{T}_{t-1}({}_c^w\mathbf{T}_{t-1})^{-1})^{-1} \\
{}_c^o\mathbf{T}_{t-1}({}_c^w\mathbf{T}_{t-1})^{-1}{}_c^w\mathbf{T}_{t-2}({}_c^o\mathbf{T}_{t-2})^{-1} &= {}_c^o\mathbf{T}_t({}_c^w\mathbf{T}_t)^{-1}{}_c^w\mathbf{T}_{t-1}({}_c^o\mathbf{T}_{t-1})^{-1}
\end{aligned} \tag{5.15}$$

To arrive at an error term from the equality condition given in Eq. (5.15), a notion of infinitesimal deviation must exist. As explained in Section 3.2 for Lie groups (such as the transformations at hand from the Special Euclidean Group SE(3)), a Lie group's tangent space at the identity, its Lie algebra, represents elements in a Euclidean-like space where differentiation is possible, and a notion of distance between two transformations exists. As shown in Section 3.2.5, the so-called log map transforms an element $\mathbf{T}$ from $SE(3)$ to its element in $\mathfrak{se}(3)$ $\boldsymbol{\xi}$. The resulting representation can be parametrized by $\boldsymbol{\xi} \in \mathbb{R}^6$. Transforming the identity transformation to its Lie algebra representation results in the zero vector and thus the length of the vector can be seen as the residual e:

$$e_t = |\boldsymbol{\epsilon}_t| = |\log(({}_{o^{t-1}}^{o^t}\mathbf{T})^{-1}1_{o^{t-2}}^{o^{t-1}}\mathbf{T})| \tag{5.16}$$

The first three entries of $\boldsymbol{\xi}$ encode translation and rotation whereas the last three entries are linked to rotation only (see Section 3.2.5 for details). This is an important property as the error stemming from rotation is bounded by $2 \cdot \pi$ but the translational

error is prinicpally unbounded. Furthermore, the velocity of an object influences the translational error. Let us assume two objects traveling at speeds $\mathbf{v}_1$ and $\mathbf{v}_2 > \mathbf{v}_1$, respectively. If both objects deviate by x% from the constant motion assumption (i.e. they exhibit non-zero acceleration between frames) this will result in two translational errors e_1^t and $e_2^t > e_1^t$. However, if we want to treat both errors as identical (to their identical *relative* deviation), we need to employ some form of normalization. [1] The proposed system uses the ratio of the estimated object velocity $\mathbf{v}_{est}^i$ for object i over the maximum expected speed of a given object type j, v_{max}^j, for normalization. Thus, the normalized velocity equals one when the object is moving at maximum speed and 0 when it is standing still. We derive the estimated object velocity by computing the median translation between frames over the previous n_v frames and multiplying this by the camera system's frame rate. To avoid this ratio reaching either zero or exceeding one (if an object is moving faster than the maximum expected velocity), it is smoothed to arrive at a normalizing factor λ_c for the translational residual as given in Eq. (5.17) and visualized in Fig. 5.11.

$$\lambda_c = \frac{1}{2}(\tanh(\frac{4 \cdot |\mathbf{v}_{est}^i|}{v_{max}^j} - 2) + 1) \tag{5.17}$$

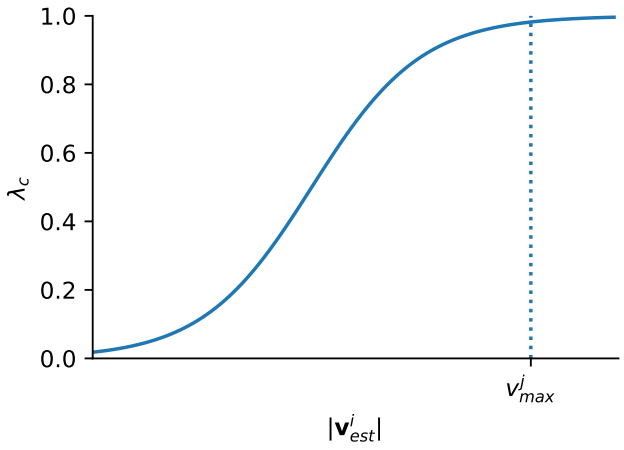


Figure 5.11: The result of evaluating this function for an estimated object velocity is used as the normalizing factor for the translational residual from the constant motion assumption.

[1] On the other hand, one may also argue that a deviation at a higher velocity must be penalized more heavily than a relatively identical deviation at a lower speed.

We make another adjustment to the objective function: as every landmark-feature pair creates a residual for a given frame, but there exist only a single constant motion residual, we scale the constant motion residual proportionally to the number of observations in a frame (or vice versa, the residuals of each observation needs to be inversely scaled by the number of total observations o_t in frame t). Finally, we multiply both the translational and the rotational residual of the constant motion assumption by fixed object-specific (for object i) constants c_t^i and c_R^i, respectively. Eq. (5.18) gives the final objective function for the Bundle Adjustment optimization.

$$\min \sum_{n,t} \frac{1}{o_t} \rho(||\mathbf{p}_t^n - \pi({}_c^o\mathbf{T}^{to}\mathbf{P}_n)||^2) + \lambda_c \cdot c_t^i \cdot |\begin{bmatrix} \mathbf{I}^3 & \mathbf{0}^3 \\ \mathbf{0}^3 & \mathbf{0}^3 \end{bmatrix} \boldsymbol{\epsilon}_t| + c_R^i \cdot |\begin{bmatrix} \mathbf{0}^3 & \mathbf{0}^3 \\ \mathbf{0}^3 & \mathbf{I}^3 \end{bmatrix} \boldsymbol{\epsilon}_t| \quad (5.18)$$

We employ the open-source framework g2o [61] for our object-level graph optimization. BAMOT adapts a sliding window approach to limit the number of parameters in the optimization problem: the proposed system only performs BA on the past n_w frames for a given object while it is active. The optimization keeps the oldest pose in this sliding window fixed s.t. the resulting trajectory remains connected. Limiting the number of parameters, in turn, decreases runtime.

5.5.5 Estimating OOBBs

At every step, we estimate OOBBs in an online fashion, which we use for later performance evaluation. As previously mentioned, an OOBB consists of a 3D location, the dimensions of an object, and the object's orientation w.r.t. its height (assumed to be parallel to gravity). To estimate an object's location, we compute the entire object point cloud median and the current landmarks' median. We then take the mean of these two medians. We use the object pose and landmark locations after BA to benefit fully from this optimization result. For objects that the system did not detect in a given frame, we only use the median of all existing landmarks as no notion of "current" landmarks exists. To calculate the orientation, we use the direction of an estimated velocity vector. We base this velocity vector on the previous n_v frames: similar to our motion extrapolation method, we compute the relative translation in the current ego frame between the current frame and frame at time $t_0 = t - \min(n_v, n_p)$ where n_p are the number of detections of an object (i.e., the length of the track). We then project the resulting velocity vector onto the x,z-plane of the current camera coordinate system. Finally, we calculate the angle between the projected velocity vector $\tilde{\mathbf{v}}$ and the x-axis of the ego frame:

$$r_y = \arccos(\begin{pmatrix} 1 & 0 \end{pmatrix} \cdot \tilde{\mathbf{v}}) \quad (5.19)$$

This angle r_y is the OOBB orientation.

We do not estimate object dimensions. Instead, we employ constant dimensions per object class. We set these dimensions to roughly match the median dimensions of objects in the dataset we evaluate on.

6 Results

The Appendix tabulates the system's hyperparameters' configuration (as discussed in Chapter 5) for the following results. The code is publicly available.[1]

6.1 KITTI Dataset

The KITTI benchmark suite [38][2] encompasses datasets and evaluation procedures for various computer vision applications, including odometry, 3D object detection, and 2D multi-object tracking. The datasets are created by driving around a car through Karlsruhe (Germany), surrounding suburban areas, and highways. The result is a variety of real-world situations with alternating amounts of other traffic participants (e.g., cars, pedestrians, and cyclists). The vehicle is equipped with two grayscale cameras, two RGB cameras, a GPS sensor, an IMU sensor, and a LiDAR (see Fig. 6.1 for the setup). The authors calibrate the car's cameras and provide both raw and rectified (see Section 3.4.2) image data and the corresponding calibration parameters. Additionally, the datasets include the LiDAR data and the ego vehicle's ground truth poses (derived from the GPS and IMU sensors). Different datasets provide further data specific to the task at hand.

We evaluate the proposed system on the multi-object tracking dataset.[3] The dataset encompasses 21 training scenes, each of which includes ground truth object detections for every frame. Each detection consists of a track ID (s.t. ground truth tracks are available), the OOBB for the detection, the class of the detected object, the 2D bounding box around the detection, and (in its multi-object tracking and segmentation extensions) 2D segmentation masks. There are also 29 scenes for testing for which only the input data (i.e., images and LiDAR) are openly available. Performance on this test set can only be evaluated by uploading results to the official KITTI benchmark servers.

[1] `https://github.com/AnselmC/bamot`
[2] `http://www.cvlibs.net/datasets/kitti/index.php`
[3] `http://www.cvlibs.net/datasets/kitti/eval_tracking.php`

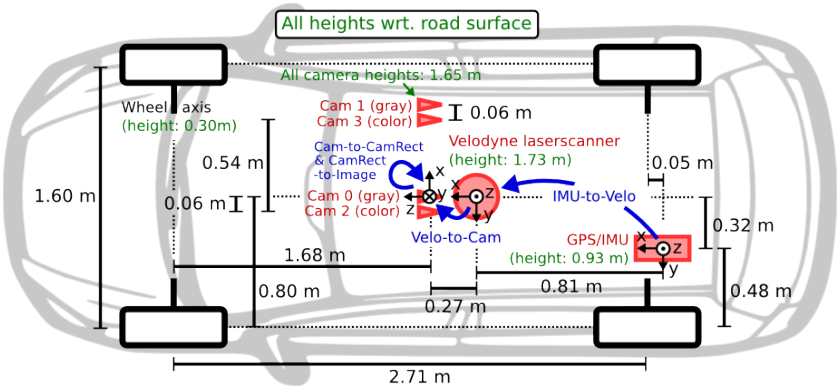


Figure 6.1: The car setup for the KITTI dataset [38].

6.2 Quantitative results

Although KITTI provides OOBB for detections and includes a benchmark for 3D MOD and 2D MOT, evaluation for 3D MOT is notably absent. As mentioned in Chapter 4, [119] introduced a benchmark for this purpose which is based on KITTI and uses adjusted CLEAR metrics (see Section 4.3.2) with a 3D IoU similarity score. Nonetheless, as such a non-generalized IoU doesn't take distance of non-overlapping bounding boxes into account (see Section 4.3.1 for an explanation) we adopt a normalized 3D GIoU [96] to calculate similarity:

$$S(d_{gt}, d_{est}) = \frac{1 + IoU(d_{gt}, d_{est}) - \frac{|C \setminus (d_{gt} \cup d_{est})|}{|C|}}{2} \tag{6.1}$$

In Eq. (6.1), d_{gt} is a ground truth detection in a given frame, d_{est} is an estimated detection, and C is the minimal OOBB which includes both d_{gt} and d_{est}.

We evaluate system performance w.r.t. cars and pedestrians on both HOTA [77] and CLEAR [10]. However, we set the system parameters to optimize for HOTA. We then compare the results of BAMOT to the LiDAR-based method AB3DMOT [119] as the authors readily provide 3D tracking results. We compare performance on the validation scenes of the learned system AB3DMOT from KITTI: $\{1, 6, 8, 10, 12, 13, 14, 15, 16, 18, 19\}$. Because AB3DMOT depends on confidence threshold filtering, we evaluate using the confidence threshold that maximizes performance for the given metric.

We evaluate via the official scripts of HOTA[4] with a custom implementation for 3D OOBB with the above mentioned normalized 3D GIoU.[5] The official 2D KITTI

[4] `https://github.com/JonathonLuiten/TrackEval`

[5] based on work from Nikita Korobov (`https://github.com/nekorobov/HOTA-metrics`)

	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA
[119] (car, $\alpha \geq 0.5$)	74.4	72.5	**76.9**	79.1	83.8	**79.9**	**91.0**	**89.0**
[119] (car, $\alpha \geq 0.25$)	**74.5**	**72.6**	76.8	79.1	**83.9**	**79.9**	90.9	**89.0**
[119] (car, $\alpha \geq 0.1$)	74.3	72.5	76.6	**79.2**	83.7	**79.9**	90.7	88.9
[119] (ped., $\alpha \geq 0.5$)	**55.5**	55.3	**55.9**	**59.9**	**76.0**	**57.5**	**84.4**	**82.5**
[119] (ped., $\alpha \geq 0.25$)	**55.5**	55.3	**55.9**	**59.9**	**76.0**	**57.5**	**84.4**	**82.5**
[119] (ped., $\alpha \geq 0.1$)	**55.5**	**55.4**	**55.9**	**59.9**	**76.0**	**57.5**	84.3	82.4

Table 6.1: Adjusting the preprocessing similarity threshold α for filtering FP to match the new setting (i.e., normalized 3D GIoU vs. 2D IoU) also has a slightly positive effect on AB3DMOT

.

evaluation removes "acceptable" FP detections in crowded areas or smaller than 25 pixels. The script determines these FP detections by matching estimated and ground-truth detections and considering all matches with IoU scores below 0.5 as unmatched and potential FP. This removal arguably makes sense in 2D (but it also originates from the fact that the original CLEAR metric evaluation only counts a detection with an IoU greater than 0.5 as a TP). However, for small (i.e., far away) detections, the depth estimates are less accurate than their 2D projections. Thus, considering detections with normalized 3D GIoU similarities below 0.5 as unmatched and then deeming them as FPs if they fall slightly below the 25-pixel height harms performance. Since this preprocessing step is supposed to be a positive contribution to overall performance in its original application, and this is not the case by one-to-one application in 3D, we adjust the threshold to be 0.25 vs. 0.5. Table 6.1 shows that this has a positive, if negligible, effect on the method we compare against w.r.t. to the primary evaluation metric HOTA.

The confidence thresholds determined to maximize AB3DMOT performance on 3D GIoU are 2.34 for cars (both HOTA and MOTA) and 1.50 (HOTA)/2.0 (MOTA) for pedestrians (note that AB3DMOT does not normalize confidences).

6.2.1 HOTA

Table 6.2 tabulates a performance comparison between AB3DMOT and BAMOT. As to be expected, the LiDAR-based method outperforms our vision-based method. This superiority arises from the fact that the underlying depth data from LiDAR is far more accurate than triangulated landmarks from RGB data. Additionally, the discrepancy in accuracy increases with distance. Finally, the LiDAR input density compared to feature-based landmarks' sparsity allows for more accurate location and shape estimates

and thus inferred bounding boxes. This not only affects the location accuracy LocA (Eq. (4.10)) but also on the detection accuracy DetA (Eq. (4.11)), association accuracy AssA (Eq. (4.14)), and thus on HOTA (Eq. (4.17)). This widespread effect comes from the similarity score (in our case given in Eq. (6.1)) upon which all metrics are based. The maximally achievable similarity S_{max} of a given MOT system gives an upper bound for HOTA and its derivative metrics. This is easily shown. For a given threshold α,

$$\text{LocA}_\alpha = \frac{\sum_{c \in \{TP_\alpha\}} S(c)}{|\{TP_\alpha\}|} \leq S_{max}. \tag{6.2}$$

Additionally, for $\alpha > S_{max}$,

$$\text{LocA}_{\alpha > S_{max}} = 0. \tag{6.3}$$

This inequality follows from assuming (at best) constant similarities S_{max} for all TP detections. Integrating over α to calculate LocA, then yields

$$\text{LocA} = \int_0^1 \text{LocA}_\alpha d\alpha \leq S_{max}^2. \tag{6.4}$$

Similarly, for DetA the upper-bound for a single threshold α is 1:

$$\text{DetA}_\alpha = \frac{|\{TP_\alpha\}|}{|\{TP_\alpha\}| + |\{FN_\alpha\}| + |\{FP_\alpha\}|} \leq 1. \tag{6.5}$$

Again, assuming a maximally attainable similarity score S_{max}, it follows that

$$\text{DetA}_{\alpha > S_{max}} = 0 \tag{6.6}$$

and thus

$$\text{DetA} = \int_0^1 \text{DetA}_\alpha d\alpha \leq S_{max}. \tag{6.7}$$

The same inequality holds for AssA and HOTA, i.e. AssA $\leq S_{max}$ and HOTA $\leq S_{max}$.

In practice, LocA is larger than the other measures as it disregards FNs and FPs. Nonetheless, the presented inequalities give a sense of how high similarity scores strongly influence all evaluation metrics' dimensions.

When comparing our system for the car object class to AB3DMOT, one also sees that while the detection accuracy decreases by $\frac{65.8-35.8}{65.8} \approx 46\%$, association accuracy merely decreases by $\frac{77.8-48.2}{77.8} \approx 38\%$. This indicates that BAMOT comparatively improves association. Another key insight is that tracking performance w.r.t. pedestrians is considerably worse. This inferiority has two reasons. First, our sparse feature-based approach detects fewer features on smaller objects which negatively affects both localization accuracy and robust association. Second, people violate our model's non-rigidity

	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA
BAMOT (car)	40.7	35.8	48.2	40.9	49.0	53.8	59.9	66.6
[119] (car)	**71.3**	**65.8**	**77.8**	**83.1**	**71.5**	**81.5**	**90.0**	**88.5**
BAMOT (ped.)	19.7	22.7	18.4	27.2	39.6	21.2	46.0	65.0
[119] (ped.)	**55.5**	**55.3**	**55.9**	**59.9**	**76.0**	**57.5**	**84.4**	**82.5**

Table 6.2: As expected, the LiDAR-based AB3DMOT is superior to our vision-based approach. However, the superiority especially stems from more accurate localization. Because we implement a feature-based approach for both 3D localization and association, the resulting sparsity makes it harder to track smaller, non-rigid pedestrians than cars.

assumption, i.e., landmarks detected on arms and legs change their location w.r.t. the fixed object coordinate-frame. Non-static landmarks lead to either estimating wrong object poses via PnP or removing such landmarks, thus increasing pedestrian point clouds' sparsity. Hence, unless pedestrians are fully visible (i.e., landmarks on the more-or-less rigid torso can be redetected and used for localization and association) or stationary, our system has trouble tracking them over time. It often loses track of the pedestrian and reinitializes a new object track. This instability results in a lower association accuracy vs. detection accuracy for pedestrians. For cars the opposite is true: association accuracy is higher than detection accuracy. Additionally, the localization accuracy (i.e., average similarity score for TP detections at a given similarity threshold) for pedestrians is only slightly below that of cars (65.0 vs. 66.6).

Table 6.3 tabulates how various parts of BAMOT influence overall performance. We alternately "switch-off" the following characteristics:

- improving TrackR-CNN's association with our three-step association pipeline detailed in Section 5.4
- initializing tracks only when the number of successfully triangulated landmarks surpasses a threshold
- the local constant motion error term in object-level BA (see Section 5.5.4)
- keeping tracks in memory and thus, "re-matchable", after not being successfully localized for a specified number of frames

For cars, our association pipeline's effect is a ~16% improvement in association accuracy, translating to a ~9% increase in the resulting HOTA score. This improvement comes at a slight (arguably negligible) cost in localization accuracy and, thus, in detection accuracy.

However, our association pipeline hurts pedestrian tracking. While association accuracy increases slightly (the number of FPAs significantly decreases, leading to much higher association recall), detection accuracy decreases. This is because we are often unable to successfully associate a detection to a track in 3D and, thus, forgo adding the detection. Using the image-based associations from TrackR-CNN will not result in this behavior leading to a performance increase of ~5% for HOTA. This is not surprising and another effect of our sparse object representation approach.

The constant motion error term in object-level BA penalizes non-smooth trajectories. This has a significant effect on pedestrians since they contain a larger outlier ratio w.r.t. feature-landmark associations because of their non-rigidity. This mainly manifests itself in the resulting detection accuracy, which increases by ~97%. Thus, they benefit from this regularizing error term. For cars, the effect is negligible.

The removal of robust initialization (i.e., initializing new objects once the system triangulates a single landmark) results in overall more but inaccurate detections. Comparatively, for low similarity thresholds, this increases TPs and decreases FNs leading to higher detection recall. However, for higher similarity thresholds, such inaccurate detections lead to FPs and FNs, resulting in lower detection precision and lower recall (compared to robust initialization). Thus, although both metrics are affected, the effect on detection precision is more severe in total as it is not counter-balanced by the improved recall at lower α. We plot the impact of the different number of minimum landmarks for initializing cars in Fig. 6.2.

Lastly, retaining tracks after they are lost has a more significant effect on association vs. detection. This comparative improvement in associating detections is because BAMOT can reassociate such lost tracks to detections in the future, increasing the TPAs for a given detection (i.e., avoiding an identity switch). Fig. 6.3 shows the effect of retaining a car track for a differing number of frames before removing it from the system. While the outcome is positive for both cars and pedestrians overall, one can see that the detection and association precision, as well as the localization accuracy improves when we delete tracks quickly. This is because we have less false positive detections that result from not being able to track an existing track and, thus, estimating its position in a given frame incorrectly. However, if we continuously reinitialize tracks, we can more accurately localize each detection as the localization is unconstrained from past detections (and associated landmarks and object poses).

6.2.2 MOTA

For completeness, we also evaluate 3D MOTA. Table 6.4 and Table 6.5 show results based on the CLEAR [10] metrics (see Section 4.3.2). The 2D CLEAR metrics deem a detection a TP if its IoU is at least 0.5. Unlike HOTA, the CLEAR metrics only evaluate

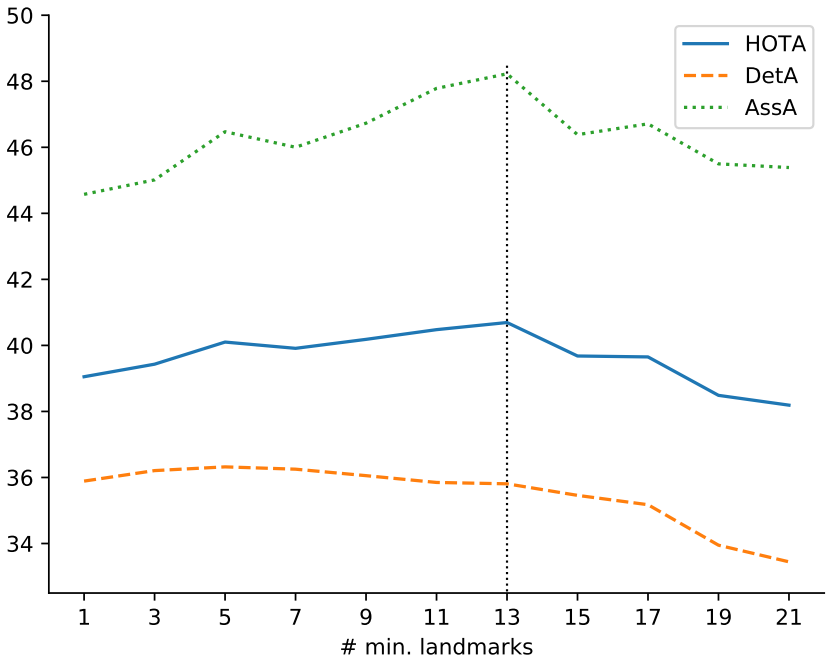


Figure 6.2: Robustly initializing objects with a minimal number of required landmarks leads to improved association and, thus, improved HOTA. However, as the number increases, the TP detection count decreases along with performance. This plot shows how the number of minimum landmarks affects our system's car tracking performance. *Best viewed in color*.

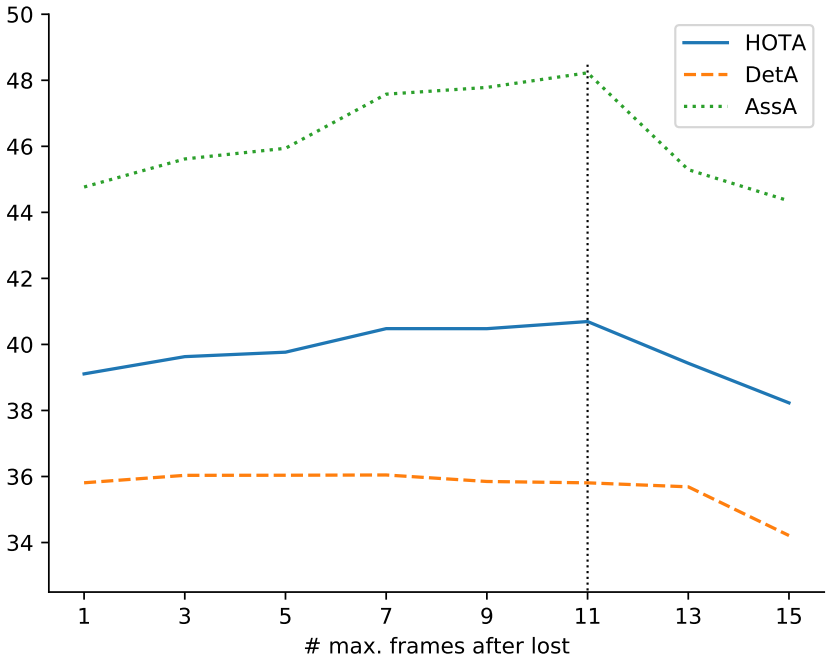


Figure 6.3: Keeping tracks after the system cannot associate a 2D detection to it (e.g., because the object is occluded), allows us to associate future detections to a track. Again, there is a trade-off: if we keep tracks too long in memory, we increase the likelihood of erroneously matching detections of new tracks to old tracks. This plot shows how this trade-off manifests itself in the performance for 3D multi-object tracking for cars. *Best viewed in color*.

	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA
BAMOT (car)	**40.7**	35.8	**48.2**	40.9	49.0	**53.8**	59.9	66.6
no impr. assoc. (car)	37.2 ↓	**35.9** ↑	41.2 ↓	41.6 ↑	47.7 ↓	52.6 ↓	52.5 ↓	66.4 ↓
no const. motion (car)	40.4 ↓	36.1 ↑	47.1 ↓	40.7 ↓	50.3 ↑	52.7 ↓	60.4 ↑	66.9 ↑
no track retention (car)	39.1 ↓	35.8 -	44.8 ↓	39.8 ↓	**51.2** ↑	49.2 ↓	**61.8** ↑	**66.9** ↑
no robust init. (car)	39.1 ↓	35.9 ↑	44.6 ↓	**42.7** ↑	46.0 ↓	50.2 ↓	57.9 ↓	66.0 ↓
none of the above (car)	35.0 ↓	34.9 ↓	37.3 ↓	40.2 ↓	46.3 ↓	49.0 ↓	48.5 ↓	65.6 ↓
BAMOT (ped.)	19.7	22.7	18.4	27.2	39.6	21.2	46.0	65.0
no impr. assoc. (ped.)	**20.7** ↑	**24.3** ↑	18.1 ↓	**30.7** ↑	36.4 ↓	**29.4** ↑	27.6 ↓	64.2 ↓
no const. motion (ped.)	13.7 ↓	11.5 ↓	17.0 ↓	12.9 ↓	35.6 ↓	19.3 ↓	44.4 ↓	63.6 ↓
no track retention (ped.)	11.6 ↓	11.4 ↓	12.5 ↓	12.0 ↓	**45.0** ↑	13.3 ↓	**59.0** ↑	**65.4** ↑
no robust init. (ped.)	14.6 ↓	12.0 ↓	18.6 ↑	13.4 ↓	36.0 ↓	20.9 ↓	45.9 ↓	63.5 ↓
none of the above (ped.)	15.5 ↓	12.4 ↓	**19.8** ↑	13.7 ↓	37.0 ↓	28.8 ↑	30.7 ↓	64.0 ↓

Table 6.3: An ablation study comparing how different high-level notions of our system influence its overall performance. Overall BAMOT benefits from a $\sim$ 17% (cars) and $\sim$ 27% (pedestrians) performance increase from using improved associations, robustly initializing new tracks, assuming constant local motion in object-level BA, and keeping lost tracks in memory s.t. BAMOT can redetect them.

a single similarity threshold. If we keep this threshold for 3D GIoU, our vision-based method performs very poorly as many estimated detections "fall under the table" and get counted as both a FP and a FN. However, if we count detections with a similarity score of at least 0.25 ($S \geq \alpha = 0.25$), this paints a very different picture. For cars and pedestrians, the MOTA score increases by large margins: from -0.0954 to 69.4 for cars and from $-18.5 to 31.6$ for pedestrians. This is because, at a lower minimally required similarity, many more estimated detections can be matched to their ground truth counterparts, thus increasing TPs. At the same time, both MOTP scores drop. This is because MOTP measures the average similarity of TPs detections. At a high minimum similarity threshold, this is necessarily higher, as only very precise detections are considered. Pedestrians exhibit fewer such precise OOBBs compared to cars. This leads to both lower MOTA and MOTP performance. Disabling the minimum similarity score altogether still increases MOTA; however, it also decreases MOTP to the same degree, arguably rendering that final step inconsequential. The effect for AB3DMOT is less severe as its OOBB estimates are decidedly more accurate due to the precision of LiDAR.

	MOTA	MOTP	MT (%)	PT (%)	ML (%)	IDSW	FRAG
BAMOT ($\alpha = 0.5$)	-0.0954	62.2	4.9	54.6	40.5	21	421
[119] ($\alpha = 0.5$)	81.0	88.1	71.9	27.0	1.08	22	149
BAMOT ($\alpha = 0.25$)	69.4	52.1	53.5	38.4	8.11	39	92
[119] ($\alpha = 0.25$)	81.1	87.6	71.9	27.1	1.08	29	147
BAMOT ($\alpha = 0$)	76.6	48.9	61.1	34.1	4.86	67	72
[119] ($\alpha = 0$)	83.0	81.9	77.8	22.2	0	129	159

Table 6.4: Comparison of cars' performance on the KITTI dataset via the detection-biased CLEAR metrics using 3D GIoU as a similarity measure. When setting the minimum threshold α to 0.5, the evaluation disregards many detections. The CLEAR metrics then count each unmatched detection *both* as a TP and as a FN, resulting in a low MOTA score.

	MOTA	MOTP	MT (%)	PT (%)	ML (%)	IDSW	FRAG
BAMOT ($\alpha = 0.5$)	-18.5	61.0	4.23	45.8	50	259	497
[119] ($\alpha = 0.5$)	66.7	80.5	44.4	40.8	14.8	85	235
BAMOT ($\alpha = 0.25$)	31.6	48.2	21.1	60.6	18.3	366	355
[119] ($\alpha = 0.25$)	66.9	79.9	45.1	40.8	14.1	88	233
BAMOT ($\alpha = 0$)	58.7	33.1	33.1	53.5	13.4	326	242
[119] ($\alpha = 0$)	68.8	69.6	48.6	39.4	12.0	156	251

Table 6.5: For pedestrians, the effect of setting the minimum threshold α below 0.5 has a similar effect as it does for cars. Overall, pedestrian tracks are often incorrectly tracked, as can be seen in the large number of IDSWs.

6.3 Qualitative results

This section discusses and illustrates the qualitative results of this work. We create visualizations with Open3D [129] [6].

As stated above, BAMOT can track cars significantly better than small, non-rigid people. However, as our system orients bounding boxes based on the object's velocity vector, the orientation and detection accuracy increase as an object continues to be tracked for multiple frames and has an estimated velocity. Additionally, the localization depends on the accuracy of triangulated landmarks and the object's distance from the camera. Fig. 6.4 shows how the orientation estimate improves for both close and far away objects as the system tracks them for a longer time; the figure also demonstrates how the camera's distance influences localization accuracy.

The dependency of orientation estimates on accurate velocity vectors results in a problem for stationary objects: here, the estimated movement of the object in 3D space is small but present, s.t. orientation estimates are erroneous, resulting in imprecise OOBBs. Aligning OOBBs based on the geometry of the point cloud could result in improved alignment in such cases. Fig. 6.5 gives examples of the difficulties BAMOT has with parked cars. Incorrect velocity vectors are less of an issue for pedestrians since OOBB orientation consists only of the yaw (i.e., rotation w.r.t. gravity) and pedestrians have similar dimensions in the relevant axes.

Fig. 6.6 illustrates how the system can handle varying degrees of difficulties in MOT. Objects in the image domain (and potentially also detected by the object detector) are not initialized by the system if not enough landmarks can be successfully triangulated. Even for initialized objects, one can see the landmark count's significant influence on its localization accuracy. Furthermore, for objects with several frames-long trajectories, the orientation of the sliding-window velocity vector accurately estimates the ground truth orientation. Additionally, even if objects leave the image domain, BAMOT continues to know their estimated locations in 3D space. This memorization has the negative side effect of FP detections. Although we do not record not-in-view detections, objects leave our field-of-view typically after KITTI considers them gone.

Fig. 6.7 visualizes several other examples of our system's tracking capabilities w.r.t. different traffic scenarios for cars.

As explained above, the sparsity of the proposed system and pedestrians' non-rigidity make it more difficult for BAMOT to accurately track pedestrians compared to cars. To track pedestrians well, enough of their nearly-rigid torso must be visible across frames to precisely localize and associate them. Compared to cars, single frame localization is also less precise since BAMOT creates fewer landmarks on smaller objects.

[6] `http://open3d.org`

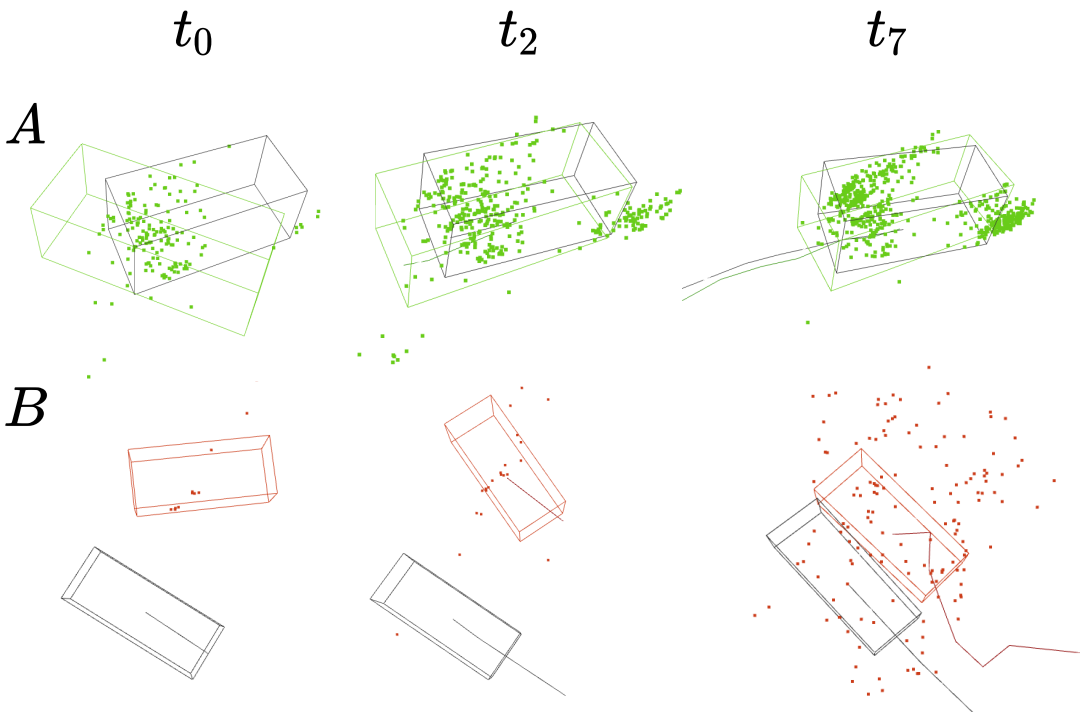


Figure 6.4: This figure demonstrates how the length of an object track and its distance to the camera influences the accuracy of OOBBs. t_0 is the first frame containing the object detection. t_2 and t_7 show the object two and seven frames after initialization, respectively. Ground truth OOBBs and ground truth trajectories are drawn in black, estimated OOBB and trajectories in color. *Top (A):* For close objects (approx. up to 20 times the stereo camera baseline), the localization of triangulated landmarks is precise. Additionally, as the system observes the object for several frames, the orientation of the bounding box improves. *Bottom (B):* For far-away objects (~30m in this example), BAMOT triangulates fewer and less-precise landmarks resulting in an inaccurate OOBB. Still, the estimate improves as more observations of the object are added. *Best viewed in color*.

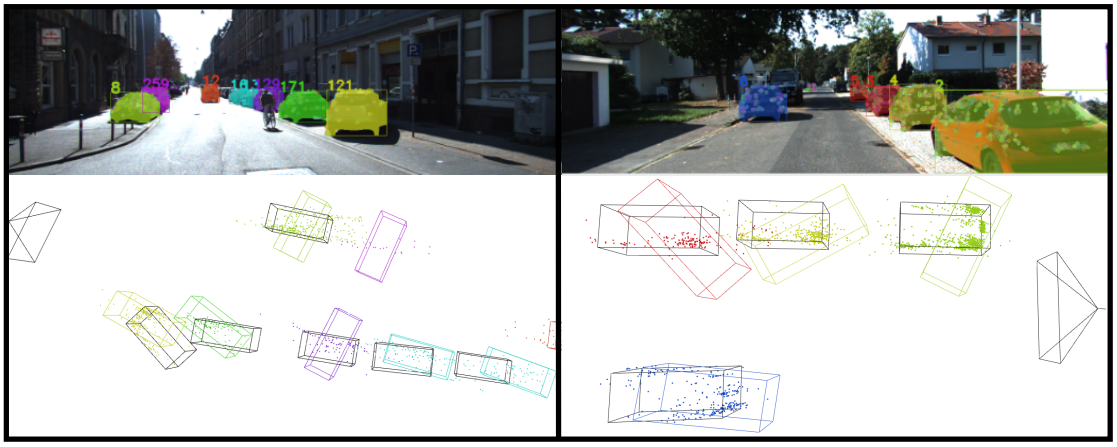


Figure 6.5: Estimating orientation from a velocity vector does not work for stationary objects. Ground truth OOBBs and trajectories are in black; their estimated counterparts are in color. *Best viewed in color*.

Fig. 6.8 shows some examples where the system is capable of localizing and tracking pedestrians. Fig. 6.9 displays weak tracking ability scenarios; instead, pedestrians are either localized imprecisely or lost and created as new tracks.

The combination of appearance and 3D information for associating existing tracks to stereo detections improves association in several situations compared to the appearance-only procedure from TrackR-CNN. One such example is shown in Fig. 6.10.

Despite the discussed problems, BAMOT works well in many cases, especially when considering localization only (i.e., object trajectories vs. OOBB). Fig. 6.11 and Fig. 6.12 illustrate such cases of accurately estimated trajectories. They also depict difficult situations such as parked cars or scenarios where IDSWs can occur. Fig. 6.13 showcases how localization accuracy decreases with distance from the camera. The Figure, however, also shows how this is less of an issue if the track is first localized close to the camera and then moves away from it. Localization accuracy further declines when the landmark count decreases. For the trajectory figures, we match estimated trajectories to ground truth trajectories based on the average distance between detections weighted by the number of frames that tracks overlap.

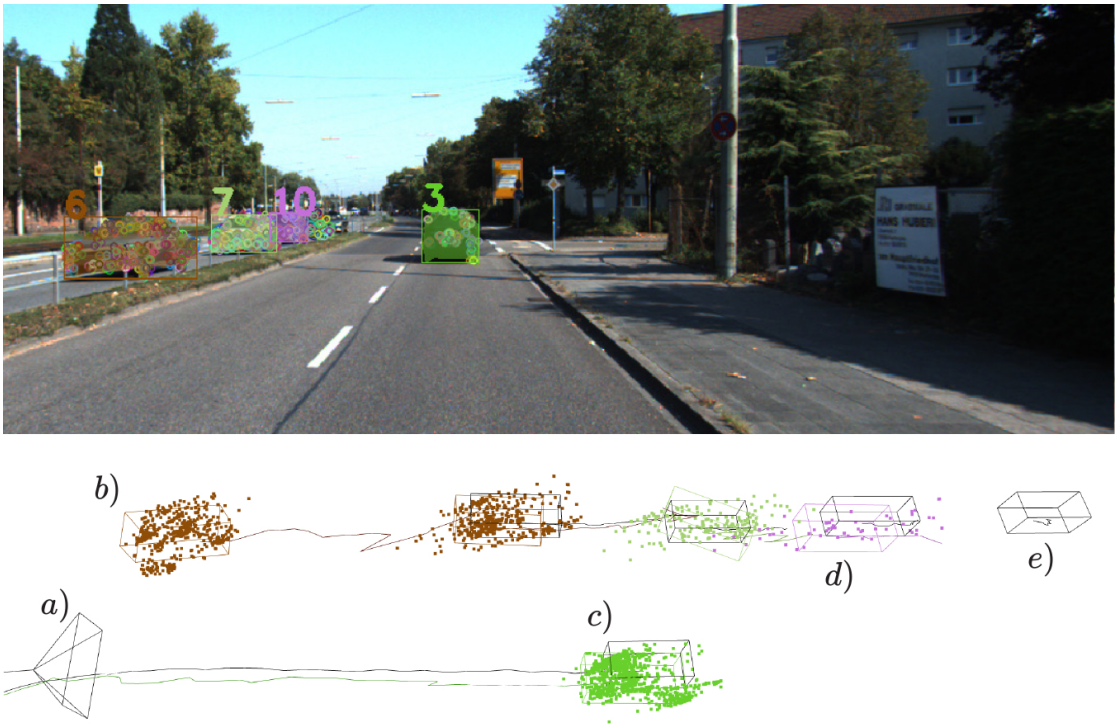


Figure 6.6: *Top*: The current frame including the detected object segmentation masks and the associated object track IDs. Additionally, we draw the extracted ORB features onto the image. *Bottom*: A 3D visualization of the ego-vehicle (a)), the ground truth OOBBs and trajectories (black), and the estimated object tracks (colorized). b) Even if the object leaves the image domain (no ground truth box exists) BAMOT continues to track it for several frames and can reassociate it. c) A non-occluded object that isn't too far away and has been tracked for several frames is accurately (for vision-based systems) estimated, whereas objects with fewer landmarks exhibit less precise localization (d)). e) If we cannot triangulate enough landmarks, the system does not initialize an object even if detected in the image domain. *Best viewed in color.*

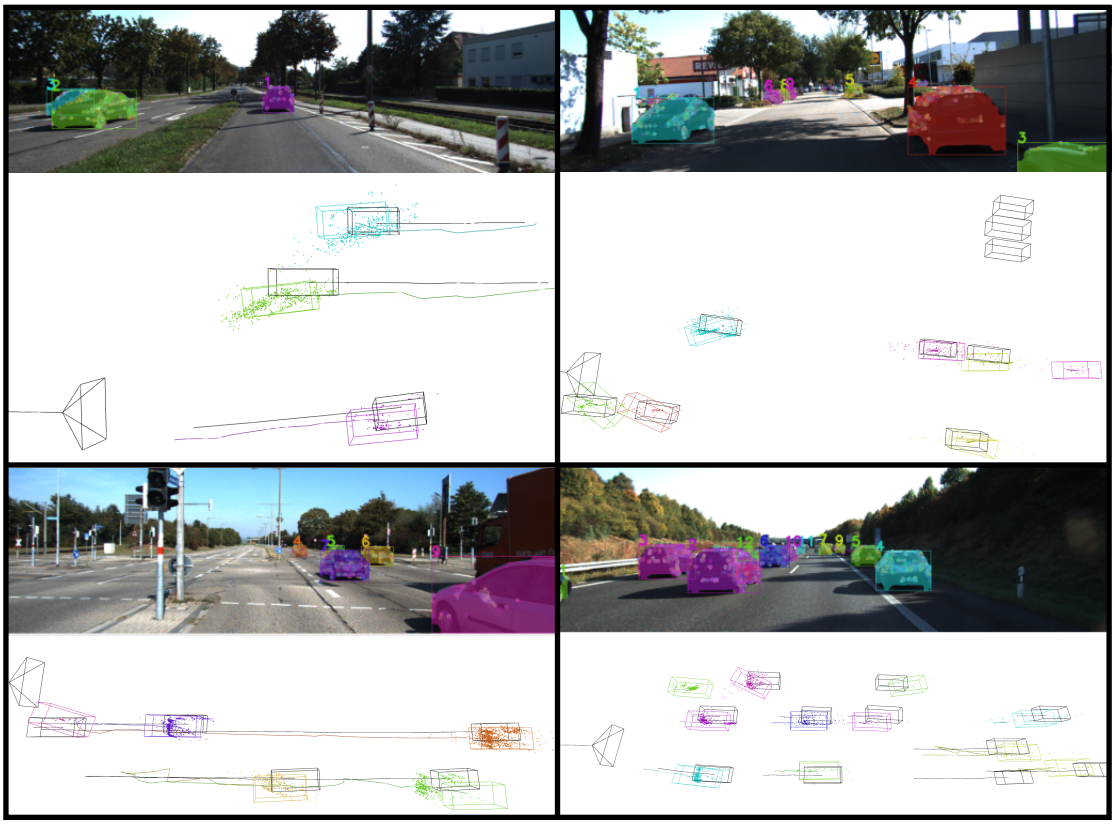


Figure 6.7: Several examples of BAMOT's ability to track cars. Each example consists of the RGB image and 2D detections above the 3D visualization of object tracks. Ground truth objects are in black; estimated tracks in color. *Best viewed in color.*

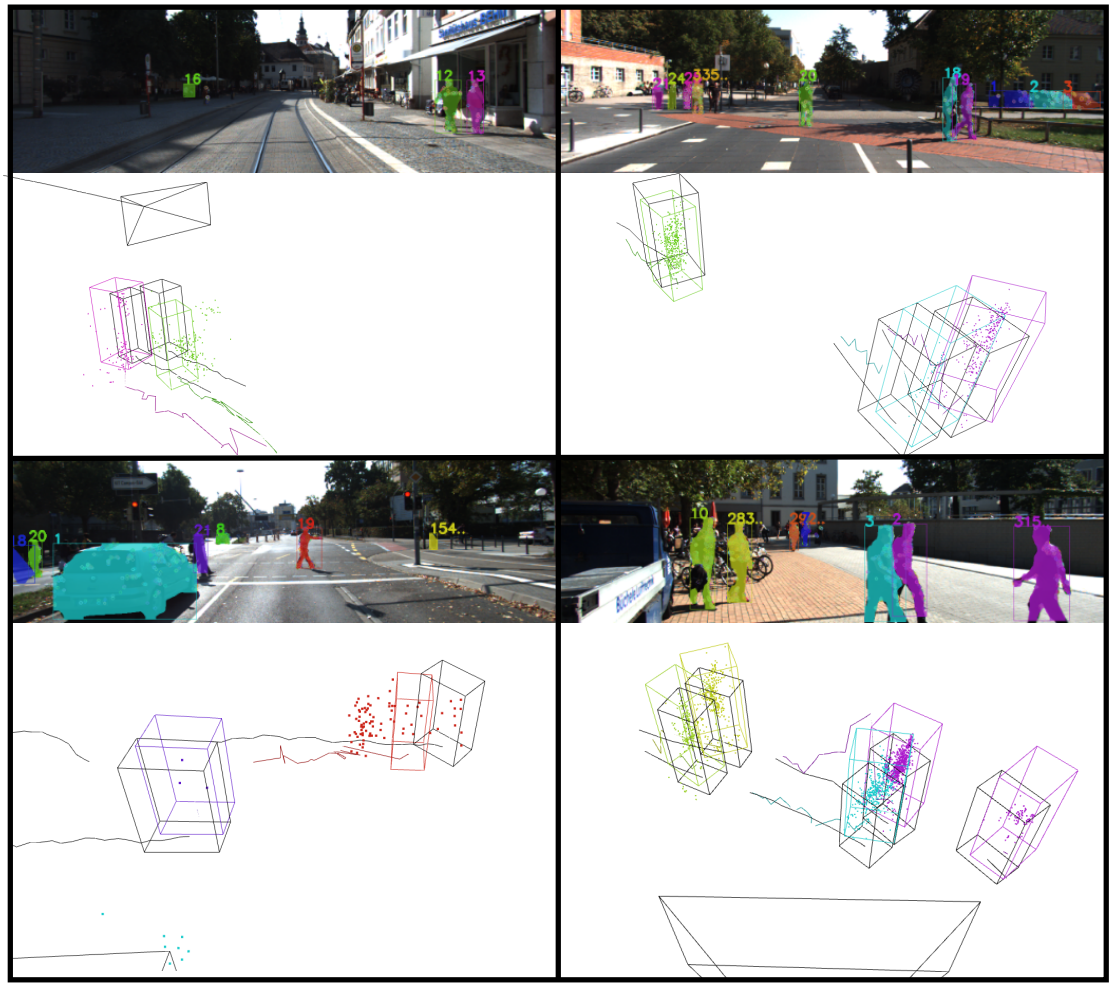


Figure 6.8: When pedestrians are not occluded, BAMOT can create enough landmarks to localize them accurately. Additionally, if they remain unoccluded for several frames, the system can track landmarks on their torso over time. *Best viewed in color*.

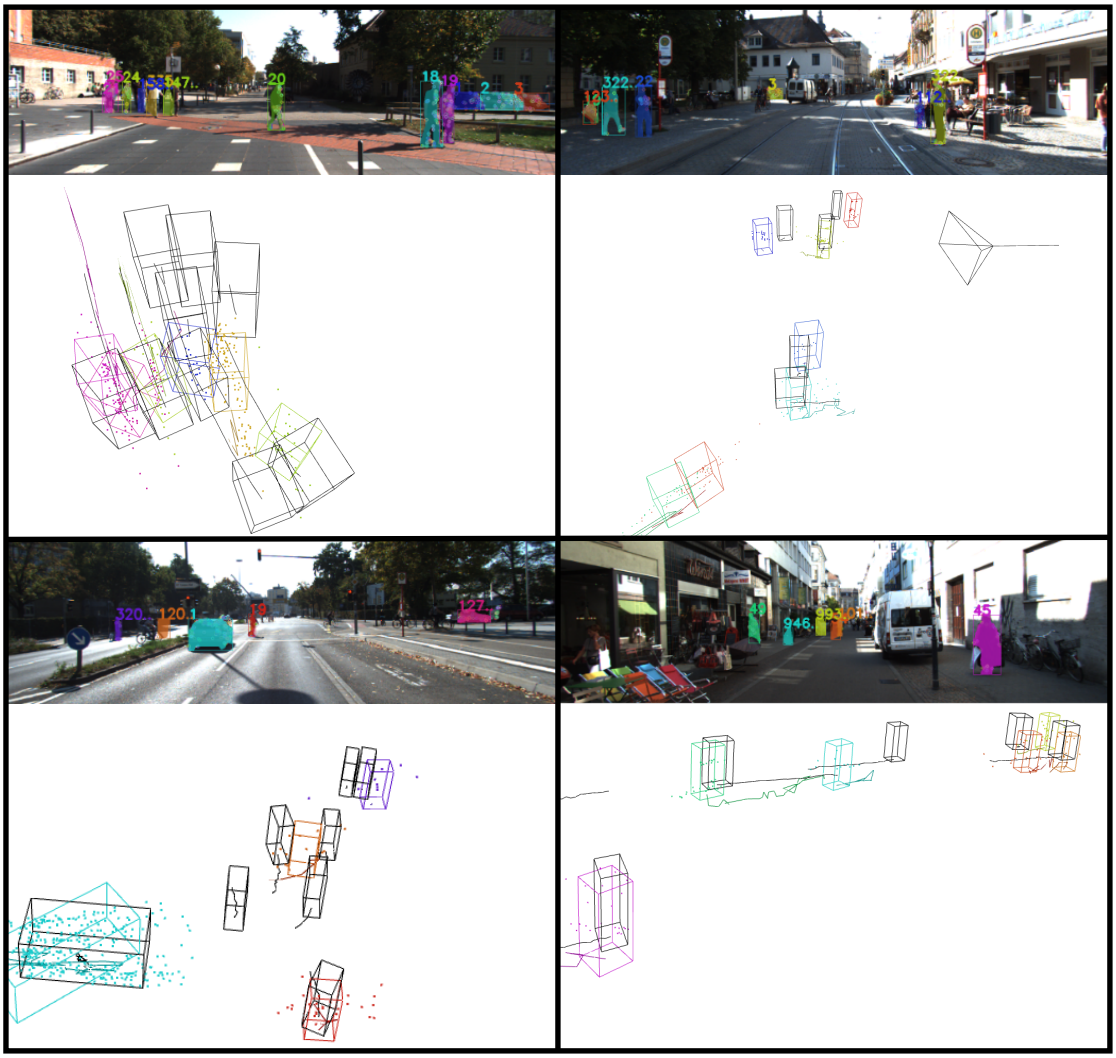


Figure 6.9: The non-rigidity of pedestrians combined with our sparse approach makes association difficult. Often, pedestrian tracks are also not initialized because they are heavily occluded, and hence, our proposal cannot match enough features between the left and right image. This figure illustrates some difficult examples. *Best viewed in color*.

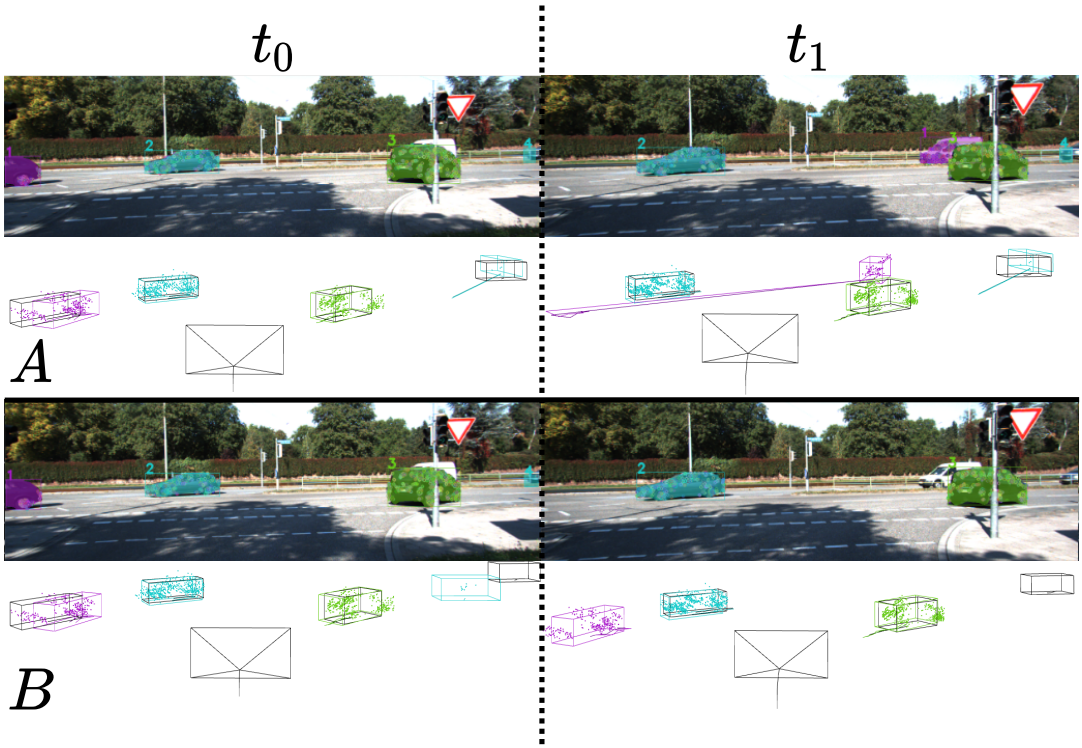


Figure 6.10: *Top (A)*: Associations by TrackR-CNN are only appearance-based. Hence, associations between two frames t_0 and t_1 that make no sense in 3D are possible (far-left, purple bounding box). *Bottom (B)*: Our association pipeline does take 3D information into account and, hence, the erroneous association from TrackR-CNN does not occur. Also, note the motion (and absence thereof) of the far-left (blue) bounding box: our system filtered a second incorrect association from TrackR-CNN in previous frames. *Best viewed in color*.

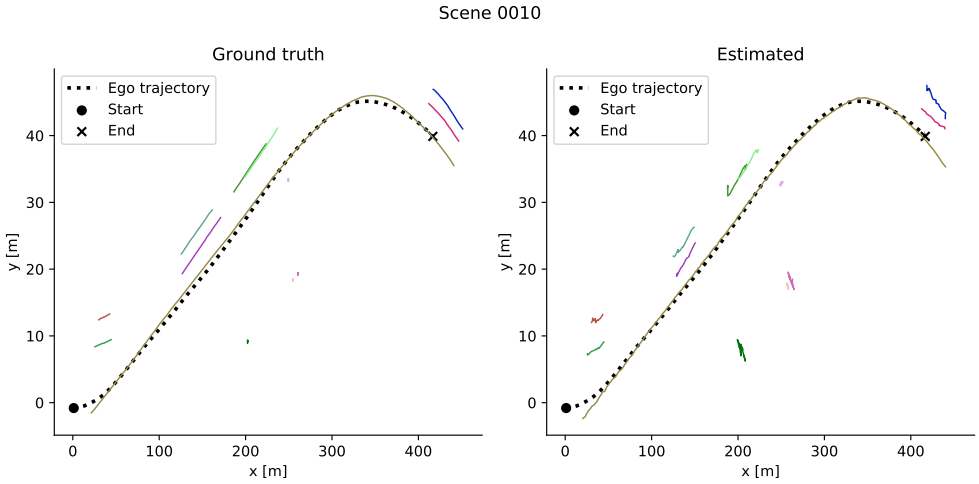


Figure 6.11: Our system performs well for moving cars as exemplified by these trajectories. However, parked cars at a distance often lead to jagged trajectories rather than points. One can also see how estimates become more accurate (and trajectories smoother) as we track the object for a longer period of time. *Best viewed in color.*

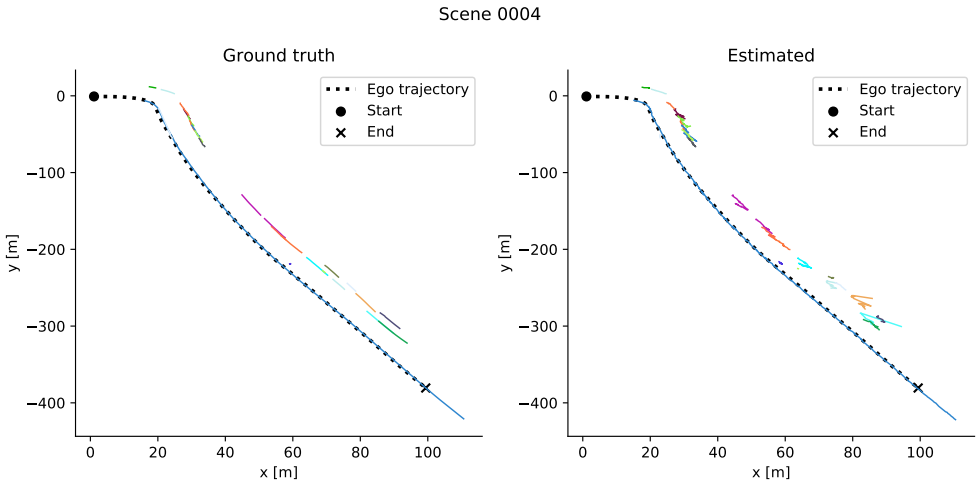


Figure 6.12: When an object is not tracked for several frames and far from the camera, our system allows for larger 3D motions. Generally, this improves performance. However, in some cases this can lead to IDSWs as exemplified by some of the inconsistent trajectories. *Best viewed in color.*

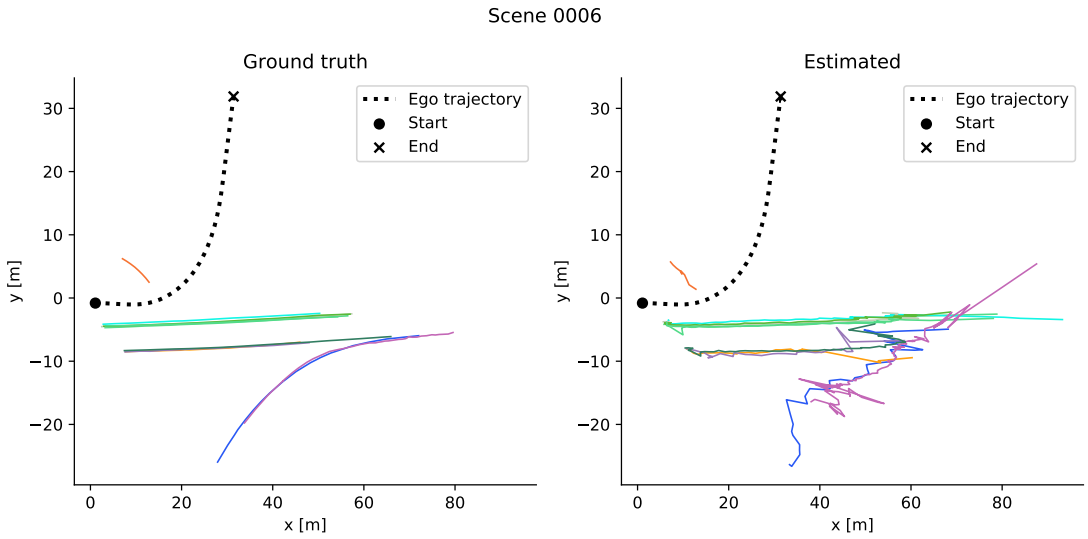


Figure 6.13: For most of the time, the ego-vehicle is at the left of this plot. This figure shows how the distance of objects w.r.t. the camera affects localization accuracy. Notice that the system can track trajectories longer than KITTI expects them to. However, in this scenario this does not count negatively towards the evaluation as both HOTA and CLEAR ignore objects that are smaller than 25 pixels in the image domain. *Best viewed in color.*

7 Conclusion & Future Work

7.1 Conclusion

In this thesis, we present a stereo-based approach for 3D MOT. Employing object-level BA on sparse point clouds, a DL-based object detector, a novel hierarchical association pipeline that combines the objects' appearance and 3D information, and a priori assumptions about objects' shapes and dynamics, we show encouraging results w.r.t. moving cars: object localization benefits from multi-frame tracking, as does orientation estimation leading to accurate OOBBs. Additionally, our association pipeline removes erroneous appearance-based associations by exploiting the available 3D knowledge our system keeps of objects.

We compare our system to the LiDAR-based AB3DMOT [119] on the KITTI dataset [38]. Since there exists no official 3D MOT benchmark for KITTI, and the adjustments presented in [119] use the CLEAR [10] metrics with the flawed similarity measure of 3D IoU, we evaluate using a normalized 3D GIoU on the recently introduced HOTA [77] metrics. We also present the corresponding CLEAR results.

While [119] can more accurately localize objects due to the superior depth estimates of LiDARs vs. stereo-cameras, we show a possible comparative advantage when it comes to tracking. Furthermore, we explore the impact of various implementation aspects of BAMOT. Due to our sparse approach and pedestrians' non-rigidity, BAMOT demonstrates its capability to a greater degree for larger, rigid objects, namely cars. The system's ability is most profound when the cars are moving (vs. parked) and initialized at not too-great distances (below ~20 times the stereo setup baseline).

7.2 Future Work

Several approaches are possible to improve performance further. First, orientation estimates for slow objects or those that haven't been detected in sufficiently many frames to estimate velocity accurately should be revised. Applying PCA to determine the dominant landmark point cloud axes may achieve this improvement in the before-mentioned situations. Second, there exists considerable research on extracting OOBBs from point clouds, e.g., [91, 106]. While these approaches assume dense LiDAR point

clouds, their results hint at the possibility of successfully training a similar network on sparse-point clouds. Adding our current OOBB estimates as initial guesses from which the network regresses may prove invaluable.

Also, comparing our approach to a dense stereo-based system, e.g., an adjusted MOTSFusion [76] that stores OOBBs, should be insightful. Ultimately, KITTI adding a 3D MOT benchmark will allow us to compare against a wide variety of proposals.

As discussed in Chapter 4, static SLAM systems benefit from removing dynamic parts of the sensor input, so integrating our work with an existing visual SLAM implementation is another possible direction of work. For such integration to make the most sense, it is necessary to make BAMOT real-time capable first.

8 Appendix

Variable name	Description	Value	SI Unit
h^{ped}	fixed height of pedestrians	1.9	m
w^{ped}	fixed width of pedestrians	0.7	m
l^{ped}	fixed length of pedestrians	0.9	m
h^{car}	fixed height of cars	1.6	m
w^{car}	fixed width of cars	1.8	m
l^{car}	fixed length of cars	4.3	m
k^{car}_{min}	min. number of landmarks required to initialize track (car)	13	-
k^{ped}_{min}	min. number of landmarks required to initialize track (ped)	3	-
c^{car}_t	const. motion weight for translation in Bundle Adjustment (car)	0.0015	-
c^{car}_R	const. motion weight for rotation in Bundle Adjustment (car)	50	-
c^{ped}_t	const. motion weight for translation in Bundle Adjustment (ped)	0.0005	-
c^{ped}_R	const. motion weight for rotation in Bundle Adjustment (ped)	0.005	-
v^{car}_{max}	max. expected velocity (car)	40	m/s
v^{ped}_{max}	max. expected velocity (ped)	8	m/s
t_{bad}	max. number of consecutive frames w/ no detection before track is deleted	11	-
d^c_{max}	max. distance to current camera frame for newly triangulated landmark	55	m
n_w	sliding window size for Bundle Adjustment	15	-
n_v	sliding window size for calculating velocity vector	10	-
n_{rel}	sliding window size for calculating mean relative translation	10	-
n_{max}	size of uniform sample for computing landmark descriptor	50	-
f_{max}	constant in max. dist d_{max}	0.75	-

List of Figures

List of Tables

Acronyms

BA Bundle Adjustment

BAMOT Bundle Adjustment for Multi-Object Tracking

BFS Breadth-First Search

CNN Convolutional Neural Network

CV Computer Vision

DL Deep Learning

DLT Direct Linear Transform

DNN Deep Neural Network

DOF Degree-of-Freedom

EKF Extended Kalman Filter

EPnP Efficient Perspective-n-Point

FN False Negative

FNA False Negative Association

FP False Positive

FPA False Positive Association

GIoU Generalized Intersection-over-Union

HA Hungarian Algorithm

HOG Histogram Of Oriented Gradients

HOTA Higher Order Tracking Accuracy

ICP Iterative-Closest-Point

IDFN Identity False Negative

IDFP Identity False Positive

IDSW Identity Switch

IDTP Identity True Positive

IoU Intersection-over-Union

LiDAR Light Detection And Ranging

MAP Maximum A Posteriori

mAP mean Average Precision

MCFP Minimum-cost Flow Problem

ML Machine-Learning

ML Mostly Lossed

MLE Maximum Likelihood Estimate

MLP Multi-Layer Perceptron

MOD Multiple Object Detection

MOT Multiple Object Tracking

MOTA Multi-Object Tracking Accuracy

MOTP Multi-Object Tracking Precision

MT Mostly Tracked

NN Neural Networks

OCR Optical Character Recognition

OD Object Detection

ODE Ordinary Differential Equation

OOBB Object Oriented Bounding Box

ORB Oriented FAST and Rotated BRIEF

PCA Principal Component Analysis

PnP Perspective-n-Point

PT Partly Tracked

RANSAC Random Sample Consensus

SfM Structure from Motion

SLAM Simultaneous Localization and Mapping

Sonar Sound Navigation And Ranging

SVD Singular Value Decomposition

TP True Positive

TPA True Positive Association

VO Visual Odometry

Bibliography

[1] J. Frank Adams, Zafer Mahmud, and M. Mimura. *Lectures on Exceptional Lie Groups*. Chicago Lectures in Mathematics Series. Chicago: University of Chicago Press, 1996. ISBN: 978-0-226-00526-3 978-0-226-00527-0.

[2] Sameer Agarwal et al. "Building Rome in a Day." en. In: *Communications of the ACM* 54.10 (Oct. 2011), pp. 105–112. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2001269.2001293.

[3] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, N.J: Prentice Hall, 1993. ISBN: 978-0-13-617549-0.

[4] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces." en. In: *Procedings of the British Machine Vision Conference 2013*. Bristol: British Machine Vision Association, 2013, pp. 13.1–13.11. ISBN: 978-1-901725-49-0. DOI: 10.5244/C.27.13.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (Dec. 2017), pp. 2481–2495. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2016.2644615.

[6] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders." In: *arXiv:2003.05991 [cs, stat]* (Mar. 2020). arXiv: 2003.05991 [cs, stat].

[7] Ioan Andrei Bârsan et al. "Robust Dense Mapping for Large-Scale Dynamic Environments." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)* (May 2018), pp. 7510–7517. DOI: 10.1109/ICRA.2018.8462974. arXiv: 1905.02781.

[8] Erkan Baser et al. "FANTrack: 3D Multi-Object Tracking with Feature Association Network." In: *arXiv:1905.02843 [cs]* (May 2019). arXiv: 1905.02843 [cs].

[9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features." en. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417. ISBN: 978-3-540-33832-1 978-3-540-33833-8. DOI: 10.1007/11744023_32.

[10] Keni Bernardin and Rainer Stiefelhagen. "Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics." en. In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–10. ISSN: 1687-5176, 1687-5281. DOI: 10.1155/2008/246309.

[11] Berta Bescos et al. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes." In: *arXiv:1806.05620 [cs]* (Aug. 2018). DOI: 10.1109/LRA.2018.2860039. arXiv: 1806.05620 [cs].

[12] Alex Bewley et al. "Simple Online and Realtime Tracking." In: *2016 IEEE International Conference on Image Processing (ICIP)* (Sept. 2016), pp. 3464–3468. DOI: 10.1109/ICIP.2016.7533003. arXiv: 1602.00763.

[13] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. en. Information Science and Statistics. New York: Springer, 2006. ISBN: 978-0-387-31073-2.

[14] Åke Björck. *Numerical Methods for Least Squares Problems*. Philadelphia: SIAM, 1996. ISBN: 978-0-89871-360-2.

[15] O. Bottema and Bernard Roth. *Theoretical Kinematics*. New York: Dover Publications, 1990. ISBN: 978-0-486-66346-3.

[16] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering: With MATLAB Exercises and Solutions*. 3rd ed. New York: Wiley, 1997. ISBN: 978-0-471-12839-7.

[17] Cesar Cadena et al. "Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age." en. In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1309–1332. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2016.2624754. arXiv: 1606.05830.

[18] Holger Caesar et al. "nuScenes: A Multimodal Dataset for Autonomous Driving." In: *arXiv:1903.11027 [cs, stat]* (May 2020). arXiv: 1903.11027 [cs, stat].

[19] Zhaowei Cai et al. "A Unified Multi-Scale Deep Convolutional Neural Network for Fast Object Detection." In: *arXiv:1607.07155 [cs]* (July 2016). arXiv: 1607.07155 [cs].

[20] John Canny. "A Computational Approach to Edge Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (Nov. 1986), pp. 679–698. ISSN: 0162-8828. DOI: 10.1109/TPAMI.1986.4767851.

[21] Xiaozhi Chen et al. "3D Object Proposals for Accurate Object Class Detection." In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[22] Yang Chen and Gérard Medioni. "Object Modelling by Registration of Multiple Range Images." en. In: *Image and Vision Computing* 10.3 (Apr. 1992), pp. 145–155. ISSN: 02628856. DOI: 10.1016/0262-8856(92)90066-C.

[23] Hsu-kuang Chiu et al. "Probabilistic 3D Multi-Object Tracking for Autonomous Driving." en. In: *arXiv:2001.05673 [cs]* (Jan. 2020). arXiv: 2001.05673 [cs].

[24] Marius Cordts et al. "The Cityscapes Dataset for Semantic Urban Scene Understanding." In: *arXiv:1604.01685 [cs]* (Apr. 2016). arXiv: 1604.01685 [cs].

[25] Jifeng Dai, Kaiming He, and Jian Sun. "Instance-Aware Semantic Segmentation via Multi-Task Network Cascades." In: *arXiv:1512.04412 [cs]* (Dec. 2015). arXiv: 1512.04412 [cs].

[26] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." en. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. San Diego, CA, USA: IEEE, 2005, pp. 886–893. ISBN: 978-0-7695-2372-9. DOI: 10.1109/CVPR.2005.177.

[27] Patrick Dendorfer et al. "MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking." en. In: *arXiv:2010.07548 [cs]* (Dec. 2020). arXiv: 2010.07548 [cs].

[28] Rachid Deriche. "Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector." en. In: *International Journal of Computer Vision* 1.2 (1987), pp. 167–187. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/BF00123164.

[29] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. "SuperPoint: Self-Supervised Interest Point Detection and Description." In: *arXiv:1712.07629 [cs]* (Apr. 2018). arXiv: 1712.07629 [cs].

[30] Jules R. Dim and Tamio Takamura. "Alternative Approach for Satellite Cloud Classification: Edge Gradient Application." en. In: *Advances in Meteorology* 2013 (2013), pp. 1–8. ISSN: 1687-9309, 1687-9317. DOI: 10.1155/2013/584816.

[31] Jingming Dong, Xiaohan Fei, and Stefano Soatto. "Visual-Inertial-Semantic Scene Representation for 3-D Object Detection." In: *arXiv:1606.03968 [cs]* (Apr. 2017). arXiv: 1606.03968 [cs].

[32] Alexey Dosovitskiy et al. "CARLA: An Open Urban Driving Simulator." In: *arXiv:1711.03938 [cs]* (Nov. 2017). arXiv: 1711.03938 [cs].

[33] Jakob Engel, Vladlen Koltun, and Daniel Cremers. "Direct Sparse Odometry." en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (Mar. 2018), pp. 611–625. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2017.2658577.

[34] Jakob Engel, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM." en. In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Vol. 8690. Cham: Springer International Publishing, 2014, pp. 834–849. ISBN: 978-3-319-10604-5 978-3-319-10605-2. DOI: 10.1007/978-3-319-10605-2_54.

[35] Karin Erdmann and Mark J. Wildon. *Introduction to Lie Algebras*. Springer Undergraduate Mathematics Series. London: Springer, 2006. ISBN: 978-1-84628-040-5.

[36] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography." en. In: *Communications of the ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/358669.358692.

[37] Duncan Frost, Victor Prisacariu, and David Murray. "Recovering Stable Scale in Monocular SLAM Using Object-Supplemented Bundle Adjustment." In: *IEEE Transactions on Robotics* 34.3 (June 2018), pp. 736–747. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2018.2820722.

[38] A Geiger et al. "Vision Meets Robotics: The KITTI Dataset." en. In: *The International Journal of Robotics Research* 32.11 (Sept. 2013), pp. 1231–1237. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364913491297.

[39] Andreas Geiger et al. "3D Traffic Scene Understanding From Movable Platforms." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.5 (May 2014), pp. 1012–1025. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2013.185.

[40] *Guide to Convolutional Neural Networks*. New York, NY: Springer Berlin Heidelberg, 2017. ISBN: 978-3-319-57549-0.

[41] Jose Guivant, Eduardo Nebot, and Stephan Baiker. "Localization and Map Building Using Laser Range Sensors in Outdoor Applications." In: *Journal of Robotic Systems* 17.10 (2000), pp. 565–583. DOI: 10.1002/1097-4563(200010)17:10<565::AID-ROB4>3.0.CO;2-6.

[42] J.-S. Gutmann and K. Konolige. "Incremental Mapping of Large Cyclic Environments." In: *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)*. Monterey, CA, USA: IEEE, 1999, pp. 318–325. ISBN: 978-0-7803-5806-5. DOI: 10.1109/CIRA.1999.810068.

[43] Brian C. Hall. *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*. Second edition. Graduate Texts in Mathematics 222. Cham ; New York: Springer, 2015. ISBN: 978-3-319-13466-6.

[44] C. Harris and M. Stephens. "A Combined Corner and Edge Detector." en. In: *Procedings of the Alvey Vision Conference 1988*. Manchester: Alvey Vision Club, 1988, pp. 23.1–23.6. DOI: 10.5244/C.2.23.

[45] R.I. Hartley. "In Defense of the Eight-Point Algorithm." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (June 1997), pp. 580–593. ISSN: 01628828. DOI: 10.1109/34.601246.

[46] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Second. Cambridge University Press, Mar. 2004. ISBN: 978-0-521-54051-3 978-0-511-81168-5. DOI: 10.1017/CBO9780511811685.

[47] Kaiming He et al. "Mask R-CNN." In: *arXiv:1703.06870 [cs]* (Jan. 2018). arXiv: 1703.06870 [cs].

[48] Jeff Heaton. "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep Learning: The MIT Press, 2016, 800 Pp, ISBN: 0262035618." en. In: *Genetic Programming and Evolvable Machines* 19.1-2 (June 2018), pp. 305–307. ISSN: 1389-2576, 1573-7632. DOI: 10.1007/s10710-017-9314-z.

[49] Mina Henein et al. "Dynamic SLAM: The Need For Speed." en. In: *arXiv:2002.08584 [cs]* (Feb. 2020). arXiv: 2002.08584 [cs].

[50] Jiahui Huang et al. "ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation." en. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019, pp. 5874–5883. ISBN: 978-1-72814-803-8. DOI: 10.1109/ICCV.2019.00597.

[51] Peter J. Huber. "Robust Estimation of a Location Parameter." en. In: *The Annals of Mathematical Statistics* 35.1 (Mar. 1964), pp. 73–101. ISSN: 0003-4851. DOI: 10.1214/aoms/1177703732.

[52] David Hutchison et al. "BRIEF: Binary Robust Independent Elementary Features." en. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Vol. 6314. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 778–792. ISBN: 978-3-642-15560-4 978-3-642-15561-1. DOI: 10.1007/978-3-642-15561-1_56.

[53] Paul Jaccard. "THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1." en. In: *New Phytologist* 11.2 (Feb. 1912), pp. 37–50. ISSN: 0028-646X, 1469-8137. DOI: 10.1111/j.1469-8137.1912.tb05611.x.

[54] Bernd Jähne and Horst Haussecker, eds. *Computer Vision and Applications: A Guide for Students and Practitioners*. San Diego: Academic Press, 2000. ISBN: 978-0-12-379777-3.

[55] Olaf Kahler et al. "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices." In: *IEEE Transactions on Visualization and Computer Graphics* 21.11 (Nov. 2015), pp. 1241–1250. ISSN: 1077-2626, 1941-0506, 2160-9306. DOI: 10.1109/TVCG.2015.2459891.

[56] Christian Kerl, Jurgen Sturm, and Daniel Cremers. "Robust Odometry Estimation for RGB-D Cameras." en. In: *2013 IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany: IEEE, May 2013, pp. 3748–3754. ISBN: 978-1-4673-5643-5 978-1-4673-5641-1. DOI: 10.1109/ICRA.2013.6631104.

[57] Deok-Hwa Kim and Jong-Hwan Kim. "Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment." In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1565–1573. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2016.2609395.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." en. In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3065386.

[59] Jason Ku et al. "Joint 3D Proposal Generation and Object Detection from View Aggregation." In: *arXiv:1712.02294 [cs]* (July 2018). arXiv: 1712.02294 [cs].

[60] H. W. Kuhn. "The Hungarian Method for the Assignment Problem." en. In: *Naval Research Logistics Quarterly* 2.1-2 (Mar. 1955), pp. 83–97. ISSN: 00281441, 19319193. DOI: 10.1002/nav.3800020109.

[61] Rainer Kummerle et al. "G^2o: A General Framework for Graph Optimization." en. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 3607–3613. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5979949.

[62] Serge Lang. *Algebra*. Rev. 3rd ed. Graduate Texts in Mathematics 211. New York: Springer, 2002. ISBN: 978-0-387-95385-4.

[63] Laura Leal-Taixe, Gerard Pons-Moll, and Bodo Rosenhahn. "Everybody Needs Somebody: Modeling Social and Grouping Behavior on a Linear Programming Multiple People Tracker." en. In: *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. Barcelona, Spain: IEEE, Nov. 2011, pp. 120–127. ISBN: 978-1-4673-0063-6 978-1-4673-0062-9 978-1-4673-0061-2. DOI: 10.1109/ICCVW.2011.6130233.

[64] Y. Lecun et al. "Gradient-Based Learning Applied to Document Recognition." In: *Proceedings of the IEEE* 86.11 (Nov./1998), pp. 2278–2324. ISSN: 00189219. DOI: 10.1109/5.726791.

[65] I. Leichter and E. Krupka. "Monotonicity and Error Type Differentiability in Performance Measures for Target Detection and Tracking in Video." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.10 (Oct. 2013), pp. 2553–2560. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2013.70.

[66] J.J. Leonard and H.F. Durrant-Whyte. "Simultaneous Map Building and Localization for an Autonomous Mobile Robot." In: *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*. Osaka, Japan: IEEE, 1991, pp. 1442–1447. ISBN: 978-0-7803-0067-5. DOI: 10.1109/IROS.1991.174711.

[67] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. "EPnP: An Accurate O(n) Solution to the PnP Problem." en. In: *International Journal of Computer Vision* 81.2 (Feb. 2009), pp. 155–166. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-008-0152-6.

[68] Kenneth Levenberg. "A Method for the Solution of Certain Non-Linear Problems in Least Squares." en. In: *Quarterly of Applied Mathematics* 2.2 (July 1944), pp. 164–168. ISSN: 0033-569X, 1552-4485. DOI: 10.1090/qam/10666.

[69] Peiliang Li, Tong Qin, and Shaojie Shen. "Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving." In: *arXiv:1807.02062 [cs]* (Nov. 2018). arXiv: 1807.02062 [cs].

[70] Shile Li and Dongheui Lee. "RGB-D SLAM in Dynamic Environments Using Static Point Weighting." In: *IEEE Robotics and Automation Letters* 2.4 (Oct. 2017), pp. 2263–2270. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2017.2724759.

[71] Li Zhang, Yuan Li, and Ramakant Nevatia. "Global Data Association for Multi-Object Tracking Using Network Flows." en. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. Anchorage, AK, USA: IEEE, June 2008, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587584.

[72] Long Quan and Zhongdan Lan. "Linear N-Point Camera Pose Determination." en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.8 (Aug./1999), pp. 774–780. ISSN: 01628828. DOI: 10.1109/34.784291.

[73] H. C. Longuet-Higgins. "A Computer Algorithm for Reconstructing a Scene from Two Projections." en. In: *Nature* 293.5828 (Sept. 1981), pp. 133–135. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/293133a0.

[74] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." en. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[75] D.G. Lowe. "Object Recognition from Local Scale-Invariant Features." en. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Kerkyra, Greece: IEEE, 1999, 1150–1157 vol.2. ISBN: 978-0-7695-0164-2. DOI: 10.1109/ICCV.1999.790410.

[76] Jonathon Luiten, Tobias Fischer, and Bastian Leibe. "Track to Reconstruct and Reconstruct to Track." In: *arXiv:1910.00130 [cs]* (Apr. 2020). DOI: 10.1109/LRA.2020.2969183. arXiv: 1910.00130 [cs].

[77] Jonathon Luiten et al. "HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking." en. In: *International Journal of Computer Vision* 129.2 (Feb. 2021), pp. 548–578. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-020-01375-2.

[78] Donald W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters." en. In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (June 1963), pp. 431–441. ISSN: 0368-4245, 2168-3484. DOI: 10.1137/0111030.

[79] J. M. McCarthy. *An Introduction to Theoretical Kinematics*. Cambridge, Mass: MIT Press, 1990. ISBN: 978-0-262-13252-7.

[80] Robert K. McConnell. "Method of and Apparatus for Pattern Recognition." Pat. 4,567,610 (Arlington, Mass.).

[81] Michael Montemerlo et al. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem." In: *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, 2002, pp. 593–598. ISBN: 0-262-51129-0.

[82] Theo Moons. "3D Reconstruction from Multiple Images Part 1: Principles." en. In: *Foundations and Trends® in Computer Graphics and Vision* 4.4 (2008), pp. 287–404. ISSN: 1572-2740, 1572-2759. DOI: 10.1561/0600000007.

[83] Arsalan Mousavian et al. "3D Bounding Box Estimation Using Deep Learning and Geometry." en. In: *arXiv:1612.00496 [cs]* (Apr. 2017). arXiv: 1612.00496 [cs].

[84] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System." In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2015.2463671. arXiv: 1502.00956.

[85] Raul Mur-Artal and Juan D. Tardos. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras." In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2017.2705103. arXiv: 1610.06475.

[86] Varun Murali et al. "Utilizing Semantic Visual Landmarks for Precise Vehicle Navigation." en. In: *arXiv:1801.00858 [cs]* (Jan. 2018). arXiv: 1801.00858 [cs].

[87] Richard M. Murray, Zexiang Li, and Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. Boca Raton: CRC Press, 1994. ISBN: 978-0-8493-7981-9.

[88] Aljosa Osep et al. "Combined Image- and World-Space Tracking in Traffic Scenes." en. In: *arXiv:1809.07357 [cs]* (Sept. 2018). arXiv: 1809.07357 [cs].

[89] Aljosa Osep et al. "Multi-Scale Object Candidates for Generic Object Tracking in Street Scenes." en. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm: IEEE, May 2016, pp. 3180–3187. ISBN: 978-1-4673-8026-3. DOI: 10.1109/ICRA.2016.7487487.

[90] S. Papert. *The Summer Vision Project*. AI Memo. Massachusetts Institute of Technology, Project MAC, 1966.

[91] Charles R. Qi et al. "Frustum PointNets for 3D Object Detection from RGB-D Data." en. In: *arXiv:1711.08488 [cs]* (Apr. 2018). arXiv: 1711.08488 [cs].

[92] N Dinesh Reddy, Minh Vo, and Srinivasa G. Narasimhan. "CarFusion: Combining Point Tracking and Part Detection for Dynamic 3D Reconstruction of Vehicles." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT, USA: IEEE, June 2018, pp. 1906–1915. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00204.

[93] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection." In: *arXiv:1506.02640 [cs]* (May 2016). arXiv: 1506.02640 [cs].

[94] Jimmy Ren et al. "Accurate Single Stage Detector Using Recurrent Rolling Convolution." In: *arXiv:1704.05776 [cs]* (Apr. 2017). arXiv: 1704.05776 [cs].

[95] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (June 2017), pp. 1137–1149. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2016.2577031.

[96] Hamid Rezatofighi et al. "Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression." In: *arXiv:1902.09630 [cs]* (Apr. 2019). arXiv: 1902.09630 [cs].

[97] Ergys Ristani et al. "Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking." In: *arXiv:1609.01775 [cs]* (Sept. 2016). arXiv: 1609.01775 [cs].

[98] Edward Rosten and Tom Drummond. "Machine Learning for High-Speed Corner Detection." en. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof, and Axel Pinz. Vol. 3951. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443. ISBN: 978-3-540-33832-1 978-3-540-33833-8. DOI: 10.1007/11744023_34.

[99] Ethan Rublee et al. "ORB: An Efficient Alternative to SIFT or SURF." en. In: *2011 International Conference on Computer Vision*. Barcelona, Spain: IEEE, Nov. 2011, pp. 2564–2571. ISBN: 978-1-4577-1102-2 978-1-4577-1101-5 978-1-4577-1100-8. DOI: 10.1109/ICCV.2011.6126544.

[100] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge." In: *arXiv:1409.0575 [cs]* (Jan. 2015). arXiv: 1409.0575 [cs].

[101] Renato F. Salas-Moreno et al. "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects." en. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. Portland, OR, USA: IEEE, June 2013, pp. 1352–1359. ISBN: 978-0-7695-4989-7. DOI: 10.1109/CVPR.2013.178.

[102] Paul-Edouard Sarlin et al. "SuperGlue: Learning Feature Matching with Graph Neural Networks." en. In: *arXiv:1911.11763 [cs]* (Mar. 2020). arXiv: 1911.11763 [cs].

[103] Herbert F. Schantz. *The History of OCR, Optical Character Recognition*. Manchester Center, Vt.: Recognition Technologies Users Association, 1982. ISBN: 978-0-943072-01-2.

[104] Samuel Schulter et al. "Deep Network Flow for Multi-Object Tracking." en. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 2730–2739. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.292.

[105] Sarthak Sharma et al. "Beyond Pixels: Leveraging Geometry and Shape Cues for Online Multi-Object Tracking." en. In: *arXiv:1802.09298 [cs]* (July 2018). arXiv: 1802.09298 [cs].

[106] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud." en. In: *arXiv:1812.04244 [cs]* (May 2019). arXiv: 1812.04244 [cs].

[107] L. Sirovich and M. Kirby. "Low-Dimensional Procedure for the Characterization of Human Faces." en. In: *Journal of the Optical Society of America A* 4.3 (Mar. 1987), p. 519. ISSN: 1084-7529, 1520-8532. DOI: 10.1364/JOSAA.4.000519.

[108] Stephen M. Smith and J. Michael Brady. "SUSAN - a New Approach to Low Level Image Processing." In: *International Journal of Computer Vision* 23.1 (1997), pp. 45–78. ISSN: 09205691. DOI: 10.1023/A:1007963824710.

[109] Shuran Song, Samuel P. Lichtenberg, and Jianxiong Xiao. "SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite." en. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA: IEEE, June 2015, pp. 567–576. ISBN: 978-1-4673-6964-0. DOI: 10.1109/CVPR.2015.7298655.

[110] Shuran Song et al. "Semantic Scene Completion from a Single Depth Image." In: *arXiv:1611.08974 [cs]* (Nov. 2016). arXiv: 1611.08974 [cs].

[111] Jrgen Sturm et al. "A Benchmark for the Evaluation of RGB-D SLAM Systems." In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vilamoura-Algarve, Portugal: IEEE, Oct. 2012, pp. 573–580. ISBN: 978-1-4673-1736-8 978-1-4673-1737-5 978-1-4673-1735-1. DOI: 10.1109/IROS.2012.6385773.

[112] Pei Sun et al. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset." In: *arXiv:1912.04838 [cs, stat]* (May 2020). arXiv: 1912.04838 [cs, stat].

[113] Yuxiang Sun, Ming Liu, and Max Q.-H. Meng. "Improving RGB-D SLAM in Dynamic Environments: A Motion Removal Approach." en. In: *Robotics and Autonomous Systems* 89 (Mar. 2017), pp. 110–122. ISSN: 09218890. DOI: 10.1016/j.robot.2016.11.012.

[114] Konstantine Tsotsos, Alessandro Chiuso, and Stefano Soatto. "Robust Inference for Visual-Inertial Sensor Fusion." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, WA, USA: IEEE, May 2015, pp. 5203–5210. ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139924.

[115] Vladyslav Usenko, Nikolaus Demmel, and Daniel Cremers. "The Double Sphere Camera Model." en. In: *2018 International Conference on 3D Vision (3DV)* (Sept. 2018), pp. 552–560. DOI: 10.1109/3DV.2018.00069. arXiv: 1807.08957.

[116] Paul Viola and Michael J. Jones. "Robust Real-Time Face Detection." In: *International Journal of Computer Vision* 57.2 (May 2004), pp. 137–154. ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000013087.49260.fb.

[117] Paul Voigtlaender et al. "MOTS: Multi-Object Tracking and Segmentation." en. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, June 2019, pp. 7934–7943. ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.00813.

[118] Xiaoyu Wang et al. "Regionlets for Generic Object Detection." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.10 (Oct. 2015), pp. 2071–2084. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2015.2389830.

[119] Xinshuo Weng and Kris Kitani. "A Baseline for 3D Multi-Object Tracking." en. In: *arXiv:1907.03961 [cs]* (Dec. 2019). arXiv: 1907.03961 [cs].

[120] Xinshuo Weng and Kris Kitani. "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud." In: *arXiv:1903.09847 [cs]* (Aug. 2019). arXiv: 1903.09847 [cs].

[121] Yu Xiang et al. "Subcategory-Aware Convolutional Neural Networks for Object Proposals and Detection." In: *arXiv:1604.04693 [cs]* (Mar. 2017). arXiv: 1604.04693 [cs].

[122] Xiao-Shan Gao et al. "Complete Solution Classification for the Perspective-Three-Point Problem." en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.8 (Aug. 2003), pp. 930–943. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1217599.

[123] Shichao Yang and Sebastian Scherer. "CubeSLAM: Monocular 3D Object SLAM." en. In: *IEEE Transactions on Robotics* 35.4 (Aug. 2019), pp. 925–938. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2019.2909168. arXiv: 1806.00557.

[124] Kwang Moo Yi et al. "LIFT: Learned Invariant Feature Transform." en. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Vol. 9910. Cham: Springer International Publishing, 2016, pp. 467–483. ISBN: 978-3-319-46465-7 978-3-319-46466-4. DOI: 10.1007/978-3-319-46466-4_28.

[125] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. "Center-Based 3D Object Detection and Tracking." en. In: *arXiv:2006.11275 [cs]* (Jan. 2021). arXiv: 2006.11275 [cs].

[126] Chao Yu et al. "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments." en. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid: IEEE, Oct. 2018, pp. 1168–1174. ISBN: 978-1-5386-8094-0. DOI: 10.1109/IROS.2018.8593691.

[127] Hongyi Zhang, Andreas Geiger, and Raquel Urtasun. "Understanding High-Level Semantics by Modeling Traffic Patterns." In: *2013 IEEE International Conference on Computer Vision*. Sydney, Australia: IEEE, Dec. 2013, pp. 3056–3063. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.379.

[128] Z. Zhang. "A Flexible New Technique for Camera Calibration." en. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (Nov./2000), pp. 1330–1334. ISSN: 01628828. DOI: 10.1109/34.888718.

[129] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing." In: *arXiv:1801.09847 [cs]* (Jan. 2018). arXiv: 1801.09847 [cs].

[130] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. "Objects as Points." In: *arXiv:1904.07850 [cs]* (Apr. 2019). arXiv: 1904.07850 [cs].