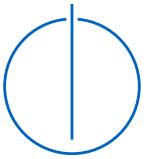
DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Incorporating Large Vocabulary Object Detection and Tracking into Visual SLAM

Maximilian Kempa



DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition, Intelligence

Incorporating Large Vocabulary Object Detection and Tracking into Visual SLAM

Integration von Objekterkennung und Objektverfolgung mit einer großen Anzahl an Objektklassen in visuellen SLAM

Author:	Maximilian Kempa
Supervisor:	Prof. Dr. Laura Leal-Taixé
Advisors:	Dr. Aljoša Ošep, Nikolaus Demmel (M.Sc.)
Submission Date:	15. Dec. 2020

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

Ludwigsburg, 14. Dec. 2020 Maximilian Kempa

Acknowledgments

I would like to thank my advisor Dr. Aljoša Ošep for many fruitful discussions and his helpful hints and detailed comments on many aspects of this work. His profound knowledge in object detection and tracking were really valuable for the development of this thesis project. I wish to acknowledge the help provided by my advisor Nikolaus Demmel through sharing his experience in the fundamentals and current developments of SLAM. Furthermore, I would like to thank Prof. Dr. Laura Leal-Taixé for giving me the opportunity to write the thesis in her research group, the Dynamic Vision and Learning Group.

Abstract

In this master's thesis in Robotics, Cognition, Intelligence, we present a dynamic visual simultaneous localization and mapping (SLAM) system based on a neural network for semantic tracking of 2D bounding boxes and bundle adjustment (BA) optimization of geometric keypoints. The system is able to estimate the 3D camera trajectory in parallel to the 3D object trajectories of many objects in the environment. In contrast to most current works in the field that focus on a small set of object classes, our approach is able to classify and track a big variety of different object classes (1230 classes). We evaluate our system on real world data in order to show quantitative results of the 3D object tracking for cars and pedestrians and compare the 3D object tracking performance to another recent approach to dynamic SLAM. Additionally, we show qualitative results of the object tracking of other object classes than cars and pedestrians which are rarely considered in the literature. Based on our evaluation, we find the key limitations of the approach and propose potential directions for improvements of the system in future work.

Kurzfassung

In dieser Masterarbeit im Studiengang Robotics, Cognition, Intelligence präsentieren wir ein dynamisches, visuelles SLAM System, welches auf einem neuronalen Netzwerk zur semantischen Objektverfolgung von 2D Objektboxen und Bündelblockausgleichung geometrischer Merkmalspunkte basiert. Das System ist in der Lage die 3D Kamera-Trajektorie und die 3D Trajektorien mehrerer Objekte in der Umgebung gleichzeitig abzuschätzen. Im Gegensatz zu vielen aktuellen Arbeiten in diesem Feld, die sich auf eine kleine Menge an Objektklassen fokussieren, ist unser Ansatz in der Lage eine große Vielfalt an verschiedenen Objektklassen (1230 Klassen) zu verfolgen. Wir evaluieren unser System auf realen Daten, um quantitative Ergebnisse der 3D Objektverfolgung für Autos und Fußgänger zu zeigen und vergleichen die Leistung der 3D Objektverfolgung mit einem anderen aktuellen Ansatz für dynamischen SLAM. Außerdem zeigen wir qualitative Ergebnisse der Objektverfolgung von weiteren Klassen (zusätzlich zu Autos und Fußgängern), welche in der Literatur selten betrachtet werden. Basierend auf unserer Evaluation identifizieren wir die wichtigsten Limitierungen unseres Ansatzes und schlagen potentielle Möglichkeiten für Verbesserungen des Systems in zukünftigen Arbeiten vor.

Contents

1 Introduction

One of the key prerequisites for autonomous mobile systems, e.g. autonomous vacuum cleaners, autonomous lawn mowers or self-driving cars is an accurate perception of the environment and the robot's pose in the environment. The problem of building a map and simultaneously estimating the own position inside this map based on video camera data is a popular field of research in the computer vision community. It is called Simultaneous Localization and Mapping (SLAM). Another important part of perception of the environment is to realize which objects are around the robot, more specifically: Where are the other objects and what kind of objects are they? This problem is referred to as object detection, respectively object tracking in case the same unique object can be identified through subsequent image frames. In the last decades most literature in the SLAM field put an emphasis on accurate tracking of the ego trajectory and robust geometric representations of the environment [1], [2], [3]. The association of map points to semantic objects was not in scope of most works. On the other hand, object detection and tracking algorithms put their focus on accurate estimation of the relative position of objects with respect to the camera but not on the trajectory of the ego camera. In order to solve more complicated real world tasks by autonomous systems, there is a strong need for a coherent environment model that consists of an accurate camera trajectory **and** an accurate estimation of the position and type of surrounding objects. This combination is often referred to as dynamic SLAM. In the last years, this field of research has gained more and more attention. Many of the new approaches focus on a limited set of classes, very often the prominent classes car and pedestrian as for example present in the KITTI [4] dataset. Nevertheless, a generic dynamic SLAM system which can be deployed to a lot of different real world applications should be able to track a wide variety of different object classes in the environment. Therefore, this thesis shows how a combination of powerful state of the art SLAM and object detection and tracking algorithms could help to create expressive models of the environment consisting of camera trajectories and object trajectories of many different object classes in one system.

1.1 Problem Statement

The goal of this thesis is to develop a system that is able to estimate the camera trajectory and the trajectory of surrounding objects of a big variety of classes simultaneously.

1.2 Outline

Chapter 2 describes relevant literature in both fields: SLAM and object detection and tracking. Chapter 3 explains the neural network architecture and hyperparameter tuning in order to track objects of a lot of different classes in the 2D image plane. Chapter 4 gives an overview of the complete system and describes how the 2D information extracted with the subsystem of chapter 3 is incorporated in the 3D world by modification of a state of the art SLAM system. Chapter 5 shows qualitative and quantitative evaluation of the object tracking and object trajectories in 3D. The thesis closes with chapter 6 which summarizes the most important findings and gives a short outlook in potential future directions of research.

2 Related Work

2.1 Object Detection and Tracking

Object detection defines the task to find objects of a predefined set of classes in an image. More specifically, one wants to find an axis-aligned bounding box that defines the position of the object in the 2D image plane and a class label that defines the type of object. Object tracking refers to the task to assign a consistent instance identifier to an object instance over time.

2.1.1 Object Detection

One of the major paradigms for object detection are region-based Convolutional Neural Network (CNN) networks. This concept was introduced in [5]. The main idea is to split up the detection process in three modules. The first module is responsible for creating region proposals, i.e. parts of the image where an object might be located (regardless which kind of object). These region proposals are forwarded to the second module, which is a CNN. The CNN calculates a low-level embedding of the image region which is then used as an input for the third module. The third module consists of a Support Vector Machine (SVM) for each class. As a last post-processing step Non-Maximum Suppression (NMS) is performed to remove multiple detections and to only keep the best scoring detections.

A rather new concept to target the object detection problem is to use keypoint estimation. CornerNet [6] defines two corner points of the bounding box as keypoints and tries to detect them. ExtremeNet [7] builds on the ideas of CornerNet but uses all four bounding box corners and the center of the bounding box as keypoints. Using keypoint estimation leads to much shorter inference time compared with region-based approaches and still reaches a similar detection performance [8].

Because of the short inference time while maintaining high accuracy this thesis builds up on CenterNet [8] which uses only the center point of the bounding box as keypoint. Using only one individual keypoint per object has the big advantage that no grouping of the keypoints to form objects is required. CenterNet predicts a keypoint heatmap for each individual class. Each pixel of the downsampled input image is assigned a value between 0 and 1 for each class individually. Training data is created by using a Gaussian kernel to assign the heatmap values. The maximum is placed at the center point of the ground truth bounding box. In case two Gaussians of the same class overlap, element-wise maximum is applied. The architecture of CenterNet is a fully-convolutional encoder-decoder network. The authors used three different backbone networks:

- stacked hourglass network [9]

- Deep Layer Aggregation (DLA) [10]
- up-convolutional residual networks [11]

The performance of the different backbones and decision on a backbone for the thesis is described in section 3.3.

In addition to the heatmap of all classes, a class-independent local offset is regressed. Its purpose is to reduce the discretization error. Furthermore, the class-independent object size is regressed as well. The training loss consists of three parts:

- penalty-reduced pixel-wise logistic regression with focal loss [12] for the heatmap
- L1 loss for the local offset prediction (class-agnostic)
- L1 loss for the size prediction (class-agnostic)

CenterNet [8] was trained and evaluated on the MS COCO [13] dataset which contains bounding box and segmentation mask labels of common objects in context (80 classes). CenterNet [8] uses the bounding box labels for training.

2.1.2 Object Tracking

The tracking-by-detection paradigm is the most popular paradigm for current trackers. It means that in each individual frame the present objects are detected. The actual tracking is done in a second step where the bounding boxes of two adjacent frames have to be associated. One example for this paradigm is SORT [14] which uses a Kalman filter for bounding box association.

Recently joint detection and tracking has become more and more popular. An example network of this category is CenterTrack [15]. It achieves a very good performance on various tracking benchmarks like MOT17 [16] and KITTI [4]. CenterTrack builds up on CenterNet [8] (see 2.1.1). CenterTrack uses the following input sources:

- the current image
- the image of the previous frame
- a class-agnostic, single heatmap of the detections of the previous frame

The network learns the displacement of the center point between the current and the previous frame in order to associate corresponding objects in two consecutive frames. Then, the actual association is just a greedy assignment. The current object is associated with the closest unmatched detection of the previous frame. If there is no unmatched detection in a certain distance to the displaced center point, a new track is started. The training procedure is very similar to CenterNet [8]. An additional L1 loss for the tracking offset regression is added to the CenterNet loss. The CenterTrack network can be trained straight-forward on video data. But, also training on static image data is possible. To do so, artificial training data is created by simulating movement between two frames. The movement is faked by randomly translating and scaling the input images.

2.2 Datasets

Large Vocabulary Instance Segmentation (LVIS) [17] is a dataset dedicated to object detection. It uses the same images as the COCO [13] dataset. In contrast to COCO, where 80 different classes are labeled, LVIS version 0.5 contains labels for 1230 classes and it consists of 57000 training images, 20000 test images and 5000 validation images. The 57000 training images correspond to 50% of the training images of COCO. By significantly increasing the number of classes compared to COCO, LVIS enables detectors to learn to detect rare classes.

Not all classes are labeled in all images to make the annotation workload feasible. If one instance of a class is labeled in a specific image, all other instances of the same class are labeled in this image as well. Additionally, for each image there is a list of classes available which are definitely not present in this specific image. For the training images it is known which classes are annotated exhaustively (set of positive classes) and which classes are definitely not present in the specific training image (set of negative classes). This means that the training loss is only calculated for the union of the set of positive classes and the set of negative classes. Classes that are neither inside the positive set nor the negative set are ignored in the loss calculation of a specific training image. For the test images, the information of positive and negative set is not available and thus the detector needs to predict all classes.

The KITTI Vision Benchmark Suite [4] contains several datasets for computer vision tasks in the automotive domain. All datasets are recorded by a vehicle equipped with two RGB cameras. The scenes include urban, interurban and highway scenarios. It contains datasets to evaluate systems for optical flow estimation, visual odometry / SLAM and 2D and 3D object detection and tracking. The SLAM dataset comprises 22 sequences and in total 39.2 km. We will use the datasets for 2D and 3D object detection and tracking and the dataset for SLAM in this thesis.

2.3 Simultaneous Localization and Mapping (SLAM)

SLAM describes the optimization problem of joint localization of a robot and mapping of its environment. Solving this problem is a key prerequisite for intelligent navigation and path planning of mobile robots (e.g. autonomous cars, autonomous vacuum cleaners or autonomous lawn mowers). SLAM is required when neither the position of the robot nor the map of its environment are known. The position of the robot is usually not known due to uncertainties in the robot's control commands and odometry data. The environment is usually also not known due to uncertainties in the data of the environment sensors. Therefore, a joint optimization of the robot's position in space and the map of its environment is required. [1]

SLAM can be performed by usage of different perception sensors. The following sensor types are commonly used for this problem [3]:

- laser range finder (2D and 3D)
- monocular camera
- stereo camera

- RGB-D camera

In the remainder of the thesis we will only consider vision based SLAM-approaches using monocular, stereo or RGB-D cameras. These approaches can be categorized into the following three subcategories [18]:

- direct methods (optimization based on the intensity values of all pixels)
- indirect methods (optimization based on features extracted from the raw images)
- semantic methods (accuracy is improved by using semantic information)

2.3.1 Direct Methods

The input for direct methods is the raw image data, i.e. the intensity values of each pixel. Direct methods aim to maximize the photometric consistency whereas indirect methods aim to maximize the geometric consistency of distinct reprojected feature points. [18]

2.3.2 Indirect Methods

Indirect methods are based on features and can be further distinguished into Bundle Adjustment (BA) and filter-based methods [18].

Filter-based Methods

Filter-based methods are based on an Extended Kalman filter (EKF) or on particle filters. The main idea is to represent the SLAM problem by a state vector that contains the position of the robot and the position of the landmarks (features). This state vector is then optimized by either using an EKF, an unscented Kalman filter or a particle filter. [3]

Bundle Adjustment (BA) Methods

BA describes the optimization process which aims to minimize the reprojection error. This means that a 3D map point is reprojected to the image plane and the distance of all reprojected points to their corresponding 2D image points is minimized. It is called local BA in case only a fixed number of frames is taken into account and global BA in case all frames are taken into account. [2]

2.3.3 Semantic Methods

In the recent years with the advent of reliable and fast object detectors the usage of semantic information for SLAM has become more and more popular. There are works that don't use the semantic information to improve the accuracy of the ego trajectory and the map but only add it after the optimization process to the final map. This way the map is enhanced with semantic information. One example for such a work is [19]. This thesis focuses on the usage of semantic information to enhance the solution of the actual SLAM problem by providing

object trajectories in addition to the static map usually estimated by SLAM systems. In the literature semantic information is mainly used in three different ways in the context of SLAM:

- usage of objects as landmarks/features
- removal of features from moving objects to improve SLAM accuracy
- simultaneous tracking of ego motion and moving objects

Usage of Objects as Landmarks/Features

Classical landmarks are low-level visual features (e.g. corners, blobs, lines). There are many works in the literature that use objects as landmarks because objects can be better uniquely distinguished than low-level features like corners. The corner from a table and a chair might look exactly the same but the two related objects are obviously completely different. Example works for this approach include [20, 21, 22, 23, 24].

In [20] the authors propose to recognize objects based on SURF features. They create an object database that contains a limited number of objects. This approach works for indoor scenes only as their object database only contains very specific instances of objects, e.g. a specific office chair. Therefore, this approach can not scale to arbitrary open-world outdoor scenarios. The basic pipeline consists of an EKF based monocular SLAM system and a visual recognition system that detects the predefined objects in an image stream. The detected objects are used as landmarks for the state vector of the EKF. In [21] the authors introduce SLAM++. As in [20] the approach uses an object database. They use pose-graph optimization to solve the SLAM problem. Each node in the graph represents either the estimated pose of an object and its object type or a historical pose of the camera. The object recognition is performed in real-time using a generalized Hough transform in the parameter space of point pair features. The camera tracking and accurate object pose estimation is done with the Iterative Closest Point (ICP) algorithm. They introduce a ground plane constraint in the pose graph optimization which is reasonable for indoor use cases.

In [22] a tight coupling between inertial, geometric and semantic information into a single optimization framework is introduced. This work provides a formal decomposition of the continuous metric pose optimization problem and the discrete data association and semantic information problem. In [23] the authors describe Quadric SLAM. The main idea in their paper is to compactly represent objects in 3D as dual quadrics. The resulting map consists of objects encoded as quadrics which are constrained to be ellipsoids. They don't tackle the data association problem in their work but assume the data associations to be given. Visual Semantic Odometry (VSO) [24] improves the accuracy of SLAM by using semantic information for tracking. As semantic information is invariant to viewpoint and illumination changes it increases the robustness and reduces the drift. The authors introduce a novel cost function to minimize the semantic reprojection error using an Expectation Maximization (EM) scheme.

Removal of Features from Moving Objects to Improve SLAM Accuracy

The main idea of this approach is to remove all features from moving objects based on semantic information. Moving objects disturb the accuracy of the map as these landmarks are not located at constant positions in the world. This means that the moving objects are landmarks that can be considered as outliers for the BA optimization process. Example works for this approach include [25, 26].

In [25] the authors introduce DS-SLAM which is based on ORB-SLAM2 [27]. It consists of five threads: tracking, semantic segmentation network to filter out dynamic objects, local mapping, loop closing and dense semantic map creation (voxel map). Its performance on the TUM RGB-D dataset [28] is better than the performance of ORB-SLAM2 [27]. [26] is also based on [27] and enhances it with semantic information. It consists of two modules: RGB-D SLAM with sparse features and object detection based on YOLO [29]. The main contribution of [26] is to include the output of the two modules into an integrated RGB-D semantic framework that builds relationships between keyframes and objects.

Simultaneous Tracking of Ego Motion and Moving Objects

The first two approaches build a map of the static environment and don't aim to track the position of moving objects. Depending on the application a static map is not sufficient but the position of the moving objects is also very important, e.g. in autonomous driving. Existing works that track the motion of the robot and the motion of the objects simultaneously are [30, 31, 32, 33, 34, 35].

In [30] a dynamic BA approach with tight coupling of semantic and feature measurements is introduced. The algorithm is able to track cars and pedestrians. 2D detections are combined with viewpoint classification in order to generate initial estimates for the 3D object bounding boxes and occlusion masks that help improving the feature matching accuracy. These initial estimates are refined by a dynamic BA optimization which makes use of a vehicle motion model and class-specific dimension priors. The algorithm is evaluated on the KITTI [4] tracking dataset using the Average Precision (AP) metric based on bird's-eye view 2D Intersection over Union (IoU) and 3D bounding box IoU as criteria. The estimation of the camera trajectory is evaluated on sequences of the KITTI [4] raw dataset using Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) metrics for comparison to ORB-SLAM2 [27].

ClusterSLAM [31] consists of two submodules: cluster assignments and motion property estimation. Dynamic landmarks are clustered to objects and static landmarks are assigned to one static cluster. Then, the motion of the dynamic objects and the motion of the camera (relative to the static cluster) are estimated in a decoupled fashion. At first, the position of the dynamic clusters with respect to the camera is estimated. Then, the camera motion with respect to the static landmarks is estimated. Finally, concatenation of the camera poses and the dynamic cluster poses gives the motion of the objects with respect to the static world. In general, this approach is able to track objects of any type. But as it does not contain any object classification module, there is no semantic information regarding the specific object type of a dynamic cluster available. The authors of [31] evaluate the object tracking performance on a

small dataset of synthetic indoor and outdoor scenes. The authors evaluate the performance of the camera trajectory estimation on three scenes of the KITTI [4] raw dataset.

MaskFusion [32] combines a MaskRCNN [36] network trained on the 80 COCO [13] classes with a geometric segmentation algorithm to obtain semantic object masks in the image plane. The object pose of each dynamic object is optimized by combining a geometric ICP error with a photometric cost based on brightness consistency. The authors of [32] evaluate the camera trajectory estimation performance on the TUM RGB-D dataset [28] but do not compare the 3D object tracking performance with any benchmark.

DynaSLAM II [33] computes a pixel-wise-semantic segmentation for each 2D frame. The authors do not state which network is used for this segmentation. DynaSLAM II [33] builds up on ORB-SLAM2 [27] and thus also uses ORB features [37] to match 2D keypoints. The initial camera pose is estimated based on the static keypoints (all keypoints not matched to a potentially dynamic object class, e.g. car or pedestrian) using BA. The initial pose of the different objects is estimated based on the 2D keypoints that are associated to them by the semantic segmentation. Using these initial estimates of camera and object poses, the camera and object trajectories, as well as the object 3D bounding boxes are optimized considering a smooth-motion prior. The authors of [33] evaluate the performance of the camera trajectory estimation as well as the performance of the object trajectories and 3D object bounding boxes on the KITTI Tracking [4] dataset.

OrcVIO [34] uses inertial information, semantic keypoints, 2D bounding box detections and geometric keypoints in order to estimate the camera trajectory and the poses of cars. Other classes than cars are not considered in this work. The authors of OrcVIO [34] evaluate the performance of the camera trajectory estimation on the KITTI Odometry [4] dataset and the accuracy of the object pose estimations on the KITTI [4] raw dataset.

ClusterVO [35] uses the 2D detector network YOLO [38] to detect 2D bounding boxes of objects. Additionally, it detects ORB features [37] for every 2D frame. Similarly to ClusterSLAM [31], objects are represented by clusters of 3D landmarks. In contrast to ClusterSLAM [31], the semantic class of the object is known from the detections of the 2D detector network. ClusterVO [35] uses probabilistic association to associate 2D features with 3D landmarks and to associate the detected 2D bounding boxes with the 3D object clusters. The static world is represented by the cluster that contains the static landmarks. The camera trajectory poses are estimated based on the static cluster. Each dynamic object is represented by the cluster that contains the corresponding landmarks. The poses of these object clusters are optimized by minimization of the reprojection error and consideration of a smooth motion prior while keeping the camera pose fixed. The authors of [35] evaluate the performance of the camera and object trajectory estimation on the stereo Oxford Multimotion dataset (OMD) [39] for indoor scenes and on the KITTI [4] raw (camera trajectory evaluation) and Tracking (3D object tracking evaluation) dataset for outdoor scenes.

2.3.4 ORB-SLAM and ORB-SLAM2

Bundle Adjustment (BA)

BA is an optimization method to jointly estimate the camera parameters, camera poses and 3D positions of points that are observed from two or more different views [40]. The general cost function for BA sums the reprojection errors of all observed 3D points. In case of ORB-SLAM [41] the formulation of the cost function is as follows:

> "Map point 3D locations $\mathbf{X}_{w,j} \in \mathbb{R}^3$ and keyframe poses $\mathbf{T}_{iw} \in SE(3)$, where w stands for the world reference, are optimized minimizing the reprojection error with respect to the matched keypoints $\mathbf{x}_{i,j} \in \mathbb{R}^2$. The error term for the observation of a map point j in a keyframe i is:
>
> $$\mathbf{e}_{i,j} = \mathbf{x}_{i,j} - \pi_i(\mathbf{T}_{iw}\mathbf{X}_{i,j}) \tag{2.1}$$
>
> where π_i is the projection function:
>
> $$\pi_i(\mathbf{T}_{iw}\mathbf{X}_{i,j}) = \begin{bmatrix} f_{i,u}\frac{x_{i,j}}{z_{i,j}} + c_{i,u} \\ f_{i,v}\frac{y_{i,j}}{z_{i,j}} + c_{i,v} \end{bmatrix} \tag{2.2}$$
>
> $$[x_{i,j}y_{i,j}z_{i,j}]^T = \mathbf{R}_{iw}\mathbf{X}_{wj} + \mathbf{t}_{iw} \tag{2.3}$$
>
> where $\mathbf{R}_{iw} \in SO(3)$ and $\mathbf{t}_{iw} \in \mathbb{R}^3$ are respectively the rotation and translation parts of $\mathbf{T}_{iw}$, and $(f_{i,u}, f_{i,v})$ and $(c_{i,u}, c_{i,v})$ are the focal length and principle point associated to camera i. The cost function to be minimized is:
>
> $$C = \sum_{i,j} \rho_h(\mathbf{e}_{i,j}^T \mathbf{\Omega}^{-1} \mathbf{e}_{i,j}) \tag{2.4}$$
>
> where ρ_h is the Huber robust cost function and $\mathbf{\Omega}_{i,j} = \sigma_{i,j}^2 \mathbf{I}_{2\times 2}$ is the covariance matrix associated to the scale at which the keypoint was detected." [41, p. 15]

ORB-SLAM

ORB-SLAM [41] is an open-source SLAM solution for visual monocular SLAM. It consists of three parallel threads:

- tracking
- local mapping
- loop closing

All threads use the same features (ORB-features [37]) and the same keyframes. The usage of keyframes ensure that the computational complexity of the BA optimization stays tractable even in large environments.

The tracking thread is responsible for the camera pose optimizations which are based on BA. The authors use a constant velocity motion model in order to guide the search of the map points that were observed in the previous frame. To bound the computational complexity only a local map is used for the camera pose optimization. The local map is based on the set of keyframes that have map points in common with the current frame. At the end of the tracking thread a new keyframe is inserted in case it tracks less than 90% of the reference key frame. According to the authors this ensures that new keyframes are only inserted if there is a minimum visual change.

The local mapping thread is executed on a per-keyframe basis. It removes map points that are potentially not trackable or wrongly triangulated. It creates new map points based on ORB descriptor matching of the features of the current keyframe with connected keyframes in the covisibility graph. Local BA is performed to optimize over the map points seen by the current keyframe and by the keyframes connected to it. As last step, redundant keyframes are removed and the covisibility graph is updated accordingly.

The loop closing thread tries to detect and close loops based on bag of words place recognition [42].

ORB-SLAM2

ORB-SLAM2 [27] is an open-source SLAM solution for monocular, stereo and RGB-D based visual SLAM. It builds on ORB-SLAM [41]. In case of stereo based SLAM the input images are rectified which simplifies the stereo matching significantly as all epipolar lines are horizontal. Stereo points are classified into close and far points based on the associated depth compared to the stereo baseline. The distinction into close and far points is used to add a new condition for keyframe insertion. If the number of close tracked points drops below a threshold and the current frame contains a significant number of new close stereo points, it is inserted as a new keyframe. This helps to estimate translation more accurately. The authors define separate projection functions for monocular (based on multiple views) and stereo keypoints in order to improve the accuracy of BA compared to the purely monocular approach in ORB-SLAM [41].

3 Object Detection and Tracking

3.1 Training Procedure Overview

Figure 3.1 shows an overview of the training procedure of the 2D tracker network. LVIS [17] and COCO [13] bounding box labels are applied to the COCO images. We remove duplicate labels by removing all COCO [13] labels that have an IoU bigger than 0.7 with a LVIS label as suggested by the authors of the TAO [43] paper and train CenterNet [8] with these images. Afterwards, we apply these weights as pretrained weights to the CenterTrack [15] network. We train this network based on artificial videos created by random scaling and shifting of the training images. This training run gives the final CenterTrack [15] model that serves as a 2D object tracker for all LVIS [17] classes.

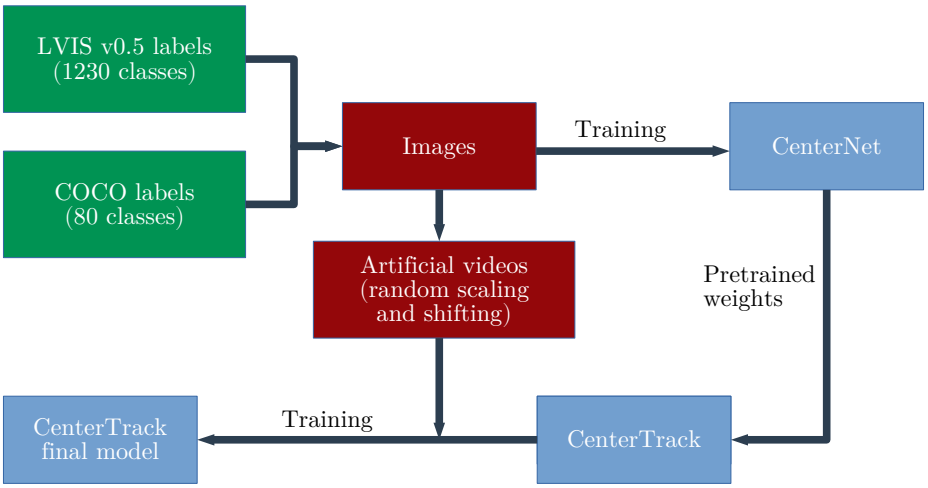


Figure 3.1: Training procedure of 2D tracker network.

3.2 Training of Detector Network

We use CenterNet [8] as a detector network. In contrast to the original paper in which it was trained on COCO data [13], we train it on the LVIS [17] v0.5 dataset. This dataset consists of around 57000 training images and 1230 different object classes. Different backbones lead to different performance of the detector. We train three different versions of the network with the following backbones:

- stacked hourglass network [9]

- DLA [10]
- up-convolutional residual networks [11]

We train the network with hourglass backbone for 140 epochs with a batch size of 9 and a learning rate of $1.25 * 10^{-4}$ applying a learning rate decrease by a factor of 10 after 90 epochs and after 120 epochs. This training schedule follows the schedule of the CenterNet [8] authors for the DLA backbone. We train the network with DLA backbone (batch size 20) and the network with up-convolutional residual network as backbone (batch size 24) with the same training schedule.

As not all classes are labeled exhaustively for all images in the LVIS dataset [17] only the ones which are labeled exhaustively in an image are considered for the calculation of the focal loss (heatmap loss).

3.3 Evaluation of Detector Network

The performance is measured with the AP metric. AP is based on the precision and recall performance of a model. Precision is defined by:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

where TP is the number of true positives and FP is the number of false positives. Recall is defined by:

$$Recall = \frac{TP}{TP + FN} \tag{3.2}$$

where TP is the number of true positives and FN is the number of false negatives. Decreasing the confidence threshold to consider a detection as valid, leads to an increased recall (as more true positives get detected) but to a decrease in precision. Plotting the precision value associated to a certain recall value gives the precision-recall curve. AP is defined as the area under this curve:

$$AP = \int_0^1 p(r)dr \tag{3.3}$$

where $p(r)$ is the precision value when recall is equal to r. We use the COCO [13] definition of AP which replaces the integral by a sum and evaluates the precision value at 101 discrete recall values:

$$AP_{IoU=x} = \frac{1}{101} \sum_{r \in \{0,0.01,0.02,\ldots,1\}} p(r) \tag{3.4}$$

This AP value is calculated for each class for 10 different IoU thresholds:

$$AP_c = \frac{1}{10} \sum_{iou \in \{0.5,0.55,\ldots,0.95\}} AP_{IoU=iou} \tag{3.5}$$

The final AP value for a model is calculated as the mean of the AP values of all classes:

$$AP = \frac{1}{C} \sum_{c \in \mathcal{C}} AP_c \tag{3.6}$$

where C is the number of classes and $\mathcal{C}$ is the set of all classes in the dataset. If not specified otherwise, we will use this definition of AP in the remainder of the thesis.

Table 3.1 shows the AP values of the three networks on the LVIS v0.5 validation dataset (5000 images). We evaluate the model on the validation set as we did not use it for training or hyperparameter tuning.

Table 3.1: CenterNet LVIS AP values for the different network architecture backbones.

Network	AP
Hourglass Network [9]	19.5%
DLA[10]	13.8%
Up-convolutional Residual Network [11]	9.5%

A qualitative analysis of the detection results suggests that some common classes like persons are not detected very robustly. The authors of [43] made a similar observation when training Detectron2 [44] on the LVIS [17] dataset. The TAO [43] authors solve this problem by adding the COCO labels to the LVIS labels and removing COCO labels that have IoU > 0.7 with a LVIS label. Retraining CenterNet on LVIS using these labels, leads to an improvement of the detection quality. Quantitatively, the AP value for the DLA backbone network increases from 13.8% to 17.0%.

3.4 Training of Tracker Network

We use CenterTrack [15] as tracker network. It is based on the CenterNet detector with DLA [10] backbone described in the previous section. We train two models, one only trained on LVIS labels (**model LVIS**), the other one (**model LVIS + COCO**) on LVIS and COCO labels as described in [43]. Tracking is performed by regressing the offset between the center points of an object in frame t and frame $t-1$. We create artificial videos consisting of two frames in order to train the regression head on the LVIS v0.5 dataset [17]. To do so the LVIS training images are randomly scaled and shifted to create this synthetic training data. This is analogue to the training procedure of the CenterTrack [15] authors for the training of their network on the COCO dataset [13].

3.5 Hyperparameter Tuning and Evaluation of Tracker Network

The performance of the tracker network cannot be evaluated directly on the LVIS dataset [17] as there is no ground truth data for tracking available in it (and no video data in general). Furthermore, we will use the KITTI tracking dataset [4] for evaluation of the combined object tracking and SLAM system (see chapter 4 and 5). Therefore, we use two different auxiliary tasks to evaluate the performance of the tracker.

The first auxiliary task is based on the Multiple Object Tracking Accuracy (MOTA) score [45] and the KITTI [4] dataset. The MOTA score [45] is defined as follows:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t} \tag{3.7}$$

where FN_t, FP_t and IDS_t are the number of false negatives, false positives and ID switches at frame t respectively and GT_t is the number of ground truth objects at frame t. If a correctly tracked object gets assigned a different ID in frame t compared to frame $t-1$, this is called ID switch. The highest reachable MOTA score is 1. This corresponds to a perfect tracker that tracks all ground truth objects without any ID switch and without detecting any wrong objects (i.e. no false positives).

We compare the output tracks of the trained network with the ground truth 2D bounding boxes of the KITTI tracking dataset for the classes car and pedestrian. As the network is trained on non-amodal bounding boxes, the Multi-Object Tracking and Segmentation (MOTS) [46] bounding boxes are used and not the original KITTI dataset amodal bounding boxes. If the overlap of a network track and a KITTI ground truth box is bigger than 0.5, this detection is kept and it gets mapped to the corresponding KITTI ground truth class. This process is called ground truth filtering. We perform ground truth filtering to avoid creating false positives for the tracks of classes, other than car and pedestrian, as LVIS [17] contains 1230 classes compared to the two ground truth classes of KITTI [4]. We evaluate the tracks that remain after ground truth filtering for their MOTA score. The second auxiliary task is a statistical analysis of the track consistency of all classes and the specifically relevant classes car and pedestrian. We analyze the mean, median and maximum track length.

3.5.1 Hyperparameter Tuning Based on MOTA Scores on KITTI Tracking Dataset

The impact of many different hyperparameters on the tracking accuracy, especially of the classes car and pedestrian can be evaluated by running MOTA score evaluations on the KITTI tracking dataset. We use the sequences 01, 02, 03, 04, 09 11, 12, 15, 17, 19 and 20 for network selection and hyperparameter tuning (training dataset). We use the remaining sequences 00, 05, 06, 07, 08, 10, 13, 14, 16 and 18 for model verification and generalization validation (validation dataset). This split is proposed in [47].

Tuning of Object Score Threshold

The object score threshold is a value between 0 and 1. The scores for all classes of one center point sum up to 1. If the highest scoring class has a score bigger than the object score threshold, the corresponding bounding box gets tracked. This hyperparameter is an inference time parameter and has a significant impact on the MOTA scores. Increasing this threshold leads to less false positives but more false negatives whereas decreasing this threshold leads to more false negatives and less false positives. The NMS threshold is set to 0.8 for the following experiments.

The following evaluations are based on **model LVIS**. As explained in section 3.3 this leads to weaknesses regarding the detection of very common classes like cars or persons. Table 3.2 shows the MOTA scores for the class car on the training dataset.

Table 3.2: CenterTrack MOTA car scores depending on object score threshold.

Object Score Threshold	MOTA Car Score	#FP	#FN	#ID-Switches
0.2	0.14	2	15714	29
0.15	0.32	19	12246	217
0.1	0.49	253	8243	875
0.05	0.51	892	5940	2154

Table 3.3 shows the MOTA scores for the class pedestrian on the training dataset.

Table 3.3: CenterTrack MOTA pedestrian scores depending on object score threshold.

Object Score Threshold	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
0.2	0.00	0	8215	0
0.15	0.01	0	8111	0
0.1	0.09	17	7418	63
0.05	0.17	98	6314	288

A small object score like 0.05 is required to reduce the number of false negatives to an acceptable value as the network was trained on a large vocabulary dataset with around 1230 classes. Due to this, the probability mass is distributed between a high number of classes. Assuming an uniform distribution over all classes, the expected object score would be $8.13 * 10^{-4}$. An object score of 0.05 can therefore be considered as a high value because this is more than 60 times higher than $8.13 * 10^{-4}$.

When using **model LVIS + COCO**, we get a different picture. The main reason is that this network has a much better detection accuracy for common classes like cars and persons.

Table 3.4 shows the MOTA scores for the class car on the training dataset based on LVIS and COCO label training. Table 3.5 shows the MOTA scores for the class pedestrian on the training dataset based on LVIS and COCO label training.

Having a stronger detector for common classes, a much higher object score threshold (e.g. 0.4) can be chosen and the number of false negatives for cars is still only around 50% of the number of false negatives when using object score threshold 0.05 and a weaker car detector (like **model LVIS**). The number of false positives is higher but the overall MOTA value for cars increases from 0.51 to 0.6 because of the big reduction of the number of false negatives. This reduction in the number of false negatives is more important than the increase in the number of false positives. The change for the class pedestrian is huge. Using the stronger detector for common classes (**model LVIS + COCO**), the MOTA value is more than three times higher (0.6 compared to 0.17). Also for pedestrians choosing a high object score threshold, e.g. 0.4,

Table 3.4: CenterTrack MOTA car scores depending on object score threshold based on LVIS and COCO label training.

Object Score Threshold	MOTA Car Score	#FP	#FN	#ID-Switches
0.4	0.60	2956	3053	1366
0.35	0.59	3301	2662	1481
0.3	0.58	3757	2336	1608
0.25	0.55	4442	2039	1771
0.2	0.52	5056	1831	1940
0.15	0.47	5776	1711	2194
0.1	0.44	6167	1620	2498
0.05	0.42	6366	1576	2700

Table 3.5: CenterTrack MOTA pedestrian scores depending on object score threshold based on LVIS and COCO label training.

Object Score Threshold	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
0.4	0.69	761	1430	398
0.35	0.69	850	1292	434
0.3	0.69	935	1146	487
0.25	0.68	1010	1046	536
0.2	0.68	1083	965	580
0.15	0.68	1136	905	633
0.1	0.67	1184	848	694
0.05	0.67	1199	803	718

performs well. To summarize, one can see that the very low object score thresholds were only needed because the base detector network (**model LVIS**) had weaknesses in detecting common classes like cars or persons. By fixing this issue (as described in section 3.3) higher object score thresholds can be chosen which leads to a more robust tracking, especially regarding false positives.

Tuning of NMS Threshold

NMS is helpful to suppress false positives caused by double detections of the same object and by detections of semantically similar classes, e.g. motor vehicle and car. NMS considers all detections that share an IoU value bigger than the NMS threshold and keeps only the one with the highest confidence score. Thus, setting the NMS threshold to 1 is equivalent with having no NMS at all. Setting the NMS threshold to 0 means that all bounding boxes that have at least some overlap are considered for the decision which bounding box to keep.

We conduct the following experiments based on **model LVIS** (refer to section 3.4) and set

the object score threshold to 0.05. Table 3.6 shows the MOTA scores for the class car on the training dataset (**model LVIS**). Table 3.7 shows the MOTA scores for the class pedestrian on the training dataset (**model LVIS**).

Table 3.6: CenterTrack MOTA car scores depending on NMS threshold (**model LVIS**).

NMS Threshold	MOTA Car Score	#FP	#FN	#ID-Switches
1	0.3	3137	5925	3674
0.9	0.45	1507	5927	2628
0.8	0.51	892	5940	2154
0.7	0.54	549	5956	1944
0.6	0.56	271	5982	1760
0.5	0.58	62	6047	1634
0.4	0.58	9	6162	1538
0.3	0.58	0	6310	1404
0.2	0.57	0	6536	1243
0.1	0.55	0	7297	954

Table 3.7: CenterTrack MOTA pedestrian scores depending on NMS threshold (**model LVIS**).

NMS Threshold	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
1	0.15	303	6287	436
0.9	0.16	200	6296	379
0.8	0.19	98	6314	288
0.7	0.19	63	6326	258
0.6	0.2	36	6351	218
0.5	0.19	7	6523	160
0.4	0.15	0	6960	74
0.3	0.12	0	7219	33
0.2	0.11	0	7321	20
0.1	0.09	0	7467	11

We conduct the following experiments based on **model LVIS + COCO** (refer to section 3.4) and set the object score threshold to 0.2. Table 3.8 shows the MOTA scores for the class car on the training dataset (**model LVIS + COCO**). Table 3.9 shows the MOTA scores for the class pedestrian on the training dataset (**model LVIS + COCO**).

Decreasing the NMS threshold, i.e. suppressing more objects leads to an increase of false negatives but a decrease of ID-Switches and false positives. In case of the class pedestrian the best MOTA values are reached for a NMS threshold in the range between 0.5 and 0.8 for **model LVIS** and between 0.2 and 0.5 for **model LVIS + COCO** . For the class car the best MOTA values are reached for a NMS threshold in the range between 0.2 and 0.6 for

Table 3.8: CenterTrack MOTA Car scores depending on NMS threshold (**model LVIS + COCO**).

NMS Threshold	MOTA Car Score	#FP	#FN	#ID-Switches
1	0.46	5940	1831	2081
0.9	0.47	5783	1831	2019
0.8	0.52	5056	1831	1940
0.7	0.63	3192	1842	1788
0.6	0.72	1702	1890	1572
0.5	0.81	263	1998	1252
0.4	0.82	32	2200	1000
0.3	0.82	2	2484	802
0.2	0.80	0	3017	588
0.1	0.74	0	4334	375

both models. As detections for the class car are rather stable (i.e. high object score values), suppressing more objects is helpful to reduce the number of false positives and ID-Switches. As detections for the class pedestrian (i.e. class person in LVIS) are less stable (i.e. lower object score values) for **model LVIS**, suppressing less objects (i.e. a higher NMS threshold) is more helpful to reduce the number of false negatives. **Model LVIS + COCO** has a better capability in detecting persons than **model LVIS** and thus shows a similar behavior regarding impact of NMS threshold for both classes.

Table 3.9: CenterTrack MOTA pedestrian scores depending on NMS threshold (**model LVIS + COCO**).

NMS Threshold	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
1	0.66	1176	965	629
0.9	0.67	1152	965	605
0.8	0.68	1083	965	580
0.7	0.72	828	967	526
0.6	0.76	544	984	471
0.5	0.81	149	1026	365
0.4	0.83	15	1099	270
0.3	0.83	1	1217	189
0.2	0.81	0	1444	142
0.1	0.76	0	1888	88

Tuning of Maximum number of Detections per Frame Parameter

The parameter K defines the maximum number of detections per frame. Increasing this parameter helps to reduce the number of false negatives but might increase the number of false positives. We execute the following experiments with an object score threshold of 0.05 and a NMS threshold of 0.4 with **model LVIS**. Table 3.10 shows the MOTA scores for the class car on the training dataset. Table 3.11 shows the MOTA scores for the class pedestrian on the training dataset.

Table 3.10: CenterTrack MOTA car scores depending on K with NMS threshold 0.4.

K	MOTA Car Score	#FP	#FN	#ID-Switches
100	0.58	8	6161	1537
120	0.58	9	5842	1732
140	0.59	11	5599	1879
160	0.60	12	5381	1995
180	0.60	14	5210	2086
200	0.60	14	5067	2180
220	0.60	14	4980	2233
240	0.61	15	4931	2267
260	0.61	15	4897	2282
280	0.61	15	4879	2295
300	0.61	16	4865	2301

We execute the following experiments with an object score threshold of 0.05 and a NMS threshold of 0.8. Table 3.12 shows the MOTA scores for the class car on the training dataset.

Table 3.13 shows the MOTA scores for the class pedestrian on the training dataset.

Table 3.11: CenterTrack MOTA pedestrian scores depending on K with NMS threshold 0.4.

K	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
100	0.15	0	6960	73
120	0.16	0	6828	88
140	0.17	0	6732	112
160	0.18	0	6653	118
180	0.18	1	6585	138
200	0.19	1	6522	155
220	0.19	1	6483	159
240	0.20	1	6447	163
260	0.20	1	6422	167
280	0.20	1	6402	173
300	0.20	1	6392	171

For a NMS threshold of 0.4, increasing K improves MOTA scores for both classes: car and pedestrian. For a NMS threshold of 0.8, increasing K only improves MOTA scores for the class pedestrian but leads to lower MOTA scores for the class car. The score for cars decreases due to a significant increase in the number of false positives.

When using the stronger detector network (**model LVIS + COCO**) an increase of the maximum number of detections per frame does not improve the MOTA scores. The reason for this is that already most cars and pedestrians are detected by the better network (**model LVIS + COCO**) and an increase in the number of detections mainly influences the number of ID-switches negatively. This can be seen in table 3.14 and table 3.15 which show the results of **model LVIS + COCO** when setting the NMS threshold to 0.4 and the object score threshold to 0.1.

Table 3.12: CenterTrack MOTA car scores depending on K with NMS threshold 0.8.

K	MOTA Car Score	#FP	#FN	#ID-Switches
100	0.51	892	5940	2153
120	0.50	1084	5587	2449
140	0.49	1265	5304	2718
160	0.49	1401	5064	2899
180	0.48	1548	4868	3064
200	0.47	1677	4704	3245
220	0.47	1761	4603	3321
240	0.47	1813	4550	3357
260	0.47	1856	4503	3384
280	0.47	1883	4481	3410
300	0.46	1907	4464	3423

Table 3.13: CenterTrack MOTA pedestrian scores depending on K with NMS threshold 0.8.

K	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
100	0.19	98	6313	288
120	0.20	130	6051	378
140	0.21	159	5865	440
160	0.23	181	5666	489
180	0.24	207	5502	552
200	0.25	228	5356	613
220	0.25	252	5255	659
240	0.26	268	5172	686
260	0.26	280	5107	716
280	0.26	294	5060	736
300	0.26	303	5032	746

Table 3.14: CenterTrack MOTA car scores depending on K with NMS threshold 0.4 (**model LVIS + COCO**).

K	MOTA Car Score	#FP	#FN	#ID-Switches
100	0.81	36	2015	1433
120	0.80	36	1974	1555
140	0.80	36	1947	1625
160	0.80	37	1940	1667
180	0.80	37	1935	1677
200	0.80	37	1932	1684
220	0.80	37	1928	1690
240	0.80	37	1928	1690
260	0.80	37	1928	1690
280	0.80	37	1928	1690
300	0.80	37	1928	1690

Table 3.15: CenterTrack MOTA pedestrian scores depending on K with NMS threshold 0.4 (**model LVIS + COCO**).

K	MOTA Pedestrian Score	#FP	#FN	#ID-Switches
100	0.84	17	994	335
120	0.83	17	984	369
140	0.83	17	972	402
160	0.83	17	967	422
180	0.83	17	967	430
200	0.83	17	965	440
220	0.83	17	963	442
240	0.83	17	961	444
260	0.83	17	961	444
280	0.83	17	961	446
300	0.83	17	961	446

3.5.2 Statistical Evaluation of Track Length Distribution

The evaluations in this section all consider one of the following sets of classes:

- all 1230 LVIS classes
- only the LVIS class car
- only the LVIS class person

We have trained **model LVIS** only based on LVIS labels and we have trained **model LVIS + COCO** on LVIS labels in conjunction with COCO labels as described in [43].

Impact of Object Score Threshold

The object score threshold is a value between 0 and 1. The scores for all classes of one center point sum up to 1. If the highest scoring class has a score bigger than the object score threshold, the corresponding bounding box gets tracked. This hyperparameter is an inference time parameter. We set the NMS threshold to 0.8 for the following experiments. Table 3.16 shows the impact of the object score threshold on the mean track length (higher is better) for **model LVIS**. Decreasing the object score threshold leads to a small decrease of the mean track length because much more objects get tracked. It improves the mean track length of the class person as their object score is often quite small in case of **model LVIS**.

Table 3.16: Mean track length of CenterTrack depending on object score threshold with NMS threshold 0.8 (**model LVIS**).

Object Score Threshold	Mean All Classes	Mean Car	Mean Person
0.2	3.18	3.00	0.00
0.15	3.18	4.16	1.13
0.1	3.04	4.48	1.95
0.05	2.84	3.34	2.16

Table 3.17 shows the impact of the object score threshold on the mean track length for **model LVIS + COCO**. As **model LVIS + COCO** is better in detecting persons and cars, the impact of the object score threshold on the mean track length is similar for all classes (including cars and persons). Decreasing the object score threshold leads to more detected objects and thus a little bit shorter mean track length. For all classes **model LVIS + COCO** shows a mean track length of around one frame more than **model LVIS**. For persons the improvement of mean track length between **model LVIS** and **model LVIS + COCO** is even around 1.5 to 2 frames.

Table 3.18 shows the impact of the object score threshold on the median track length in case of **model LVIS**. Most tracks are very short (only one or two frames). This problem can be overcome by only accepting tracks with a certain minimal track length.

For **model LVIS + COCO** the median track length is two frames for all object score thresholds between 0.5 and 0.4 and for all three categories (all classes, cars, persons).

Table 3.17: Mean track length of CenterTrack depending on object score threshold with NMS threshold 0.8 (**model LVIS + COCO**).

Object Score Threshold	Mean All Classes	Mean Car	Mean Person
0.4	4.17	3.92	3.99
0.35	4.15	3.87	3.96
0.3	4.14	3.82	3.95
0.25	4.13	3.78	3.94
0.2	4.11	3.78	3.86
0.15	4.09	3.77	3.77
0.1	3.96	3.66	3.66
0.05	3.74	3.53	3.58

Table 3.18: Median track length of CenterTrack depending on object score threshold with NMS threshold 0.8 (**model LVIS**).

Object Score Threshold	Median All Classes	Median Car	Median Person
0.2	1	1	0
0.15	1	2	1
0.1	1	2	1
0.05	1	1	1

Impact of NMS Threshold

The impact of the NMS threshold on the track length statistics is neglectable for **model LVIS** and **model LVIS + COCO**.

Impact of Maximum Number of Detections per Frame Parameter

We execute the following experiments using **model LVIS** with a NMS threshold of 0.8 and an object score threshold of 0.05. The parameter K (refer to section Tuning of Maximum number of Detections per Frame Parameter) has no impact on the general mean, median and maximum track length (of all classes). It does have an impact on the mean track length for the classes car and person as the number of detected cars and persons increases with increasing K. On the other hand more unreliable detections of cars and persons are included which leads to more tracks with a slightly shorter mean track length. Table 3.19 summarizes this impact.

Conducting the same experiment using **model LVIS + COCO** with a NMS threshold of 0.8 and an object score threshold of 0.05 leads to the results shown in table 3.20. The impact of parameter K on the mean track lengths is comparable between **model LVIS** and **model LVIS + COCO**.

Table 3.19: Mean track length of CenterTrack depending on K with NMS threshold 0.8 and object score threshold 0.05 using **model LVIS**.

K	Mean All Classes	Mean Car	Mean Person
100	2.84	3.34	2.16
120	2.84	3.16	2.25
140	2.84	3.03	2.22
160	2.85	2.94	2.23
180	2.85	2.86	2.18
200	2.85	2.8	2.17
220	2.86	2.77	2.18
240	2.86	2.76	2.18
260	2.87	2.75	2.19
280	2.87	2.74	2.16
300	2.87	2.73	2.17

Table 3.20: Mean track length of CenterTrack depending on K with NMS threshold 0.8 and object score threshold 0.05 using **model LVIS + COCO**.

K	Mean All Classes	Mean Car	Mean Person
100	3.74	3.53	3.58
120	3.75	3.50	3.52
140	3.74	3.47	3.47
160	3.74	3.45	3.43
180	3.73	3.41	3.39
200	3.74	3.38	3.36
220	3.75	3.36	3.32
240	3.76	3.33	3.31
260	3.77	3.31	3.28
280	3.78	3.29	3.26
300	3.79	3.28	3.24

3.5.3 Validation of Hyperparameter Tuning on Validation Sequences

According to the results of the previous sections (i.e. hyperparameter tuning), the best parameter set is using **model LVIS + COCO** with an object score threshold of 0.2, $K = 100$ and a NMS threshold of 0.5. In order to validate that this parameter set is not overfitting on the training data, MOTA scores are evaluated on the training and on the validation set. Table 3.21 shows the results for the classes car and pedestrian. The performance for the class

Table 3.21: MOTA scores of best parameter set on training and validation dataset.

Class	Training Set	Validation Set
MOTA score car	0.81	0.87
MOTA score pedestrian	0.81	0.76

car is slightly better and the performance for the class pedestrian is slightly worse on the validation set compared to the training set. As the differences are not really big, one can state that the hyperparameter tunining did not lead to overfitting on the training data. Table 3.22 shows the mean track lengths when using **model LVIS + COCO** with the best parameter set (object score threshold 0.2, $K = 100$ and NMS threshold 0.5) on the training and the validation set. The comparison of the mean track lengths also shows that there is no overfitting on the training dataset due to hyperparameter tuning.

Table 3.22: Mean track lengths of best parameter set on training and validation dataset.

Class	Training Set	Validation Set
Mean Track Length All Classes	3.97	3.95
Mean Track Length Car	3.64	3.75
Mean Track Length Person	3.78	3.17

4 Incorporating 2D Object Tracking into SLAM

This chapter describes how we incorporate the 2D object tracks of the LVIS [17] classes that are predicted by the CenterTrack [15] network described in the previous chapter into a modified ORB-SLAM2 [27] system. We use the **model LVIS + COCO** as described in chapter 3 with an object score threshold of 0.2, $K = 100$ and a NMS threshold of 0.5 (best hyperparameter setting).

4.1 System Overview

We feed sequences of the KITTI [4] dataset into the CenterTrack [15] network. The system also works with any other video sequences but we use KITTI [4] sequences for evaluation in this thesis. CenterTrack predicts the 2D bounding boxes with associated tracking ID and class ID for every frame. These predictions are stored in a JSON file to decouple the 2D tracking part from the 3D tracking part which is executed in a modified ORB-SLAM2 [27] system. This gives the possibility for future works to seamlessly exchange the 2D tracker network. Then, we feed the 2D tracks (stored in the JSON file) as input to our 3D object tracking system (modified ORB-SLAM2 [27]). This system tries to estimate the relative pose of all tracked objects in a frame with respect to the camera pose. As the camera trajectory with respect to the static world is estimated by the standard ORB-SLAM2 [27] algorithm in parallel to these object tracks, it is also possible to localize the tracked objects with respect to the static world. Figure 4.1 shows an overview of this system.

We will describe the details of the 3D tracking part in the following section.

4.2 Mathematical Foundations of 3D Tracking Algorithm

This section describes the mathematical foundations of the 3D tracking algorithm.

4.2.1 Notation

The transformation that transforms points in the 3-dimensional Euclidean space from coordinate frame B to coordinate frame A is denoted as $\mathbf{T}_{AB}$. We represent it as a 4×4 matrix:

$$\mathbf{T}_{AB} = \begin{bmatrix} \mathbf{R}_{AB} & \mathbf{t}_{AB} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.1}$$

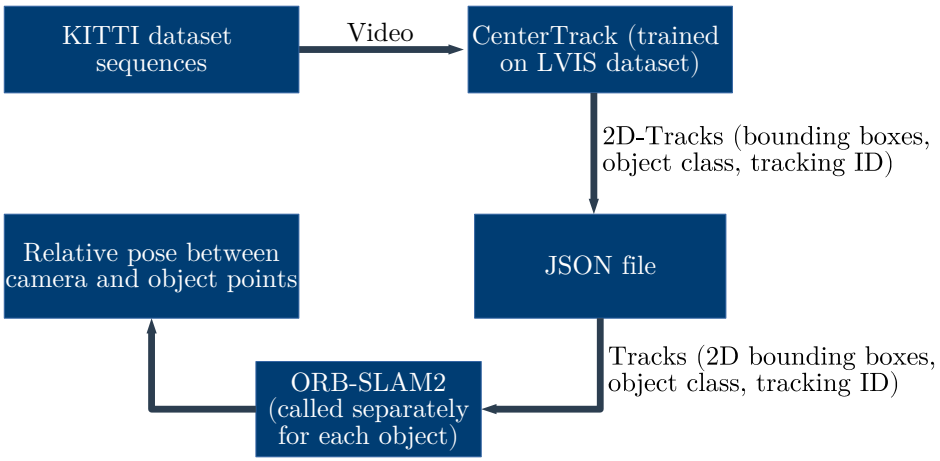


Figure 4.1: System overview.

where $\mathbf{R}_{AB} \in SO(3)$ is the 3×3 rotation matrix from coordinate frame B to coordinate frame A and $\mathbf{t}_{AB} \in \mathbb{R}^3$ is the 3×1 translation vector, translating the origin of frame B to frame A and $\mathbf{0} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$. The transform $\mathbf{T}_{AB}$ also transforms the coordinate frame B into the coordinate frame A.

The following equation transforms the coordinates of a homogenous 3D point defined in the coordinate frame B $\mathbf{p}_B$ into the coordinate frame A:

$$\mathbf{p}_A = \mathbf{T}_{AB}\mathbf{p}_B = (x, y, z, 1)^T \tag{4.2}$$

4.2.2 Mathematical Formulation of Object Tracking

For each object we define an object map coordinate frame, denoted with M and a camera frame, attached to the optical center of the camera, denoted with C. We denote the time steps of a track as $t = \{0, 1, \ldots, T\}$ and the transform from camera frame to map frame for the i-th time step as $\mathbf{T}_{M_i C_i}$. The object map frame M gets initialized in the first frame of the object track and stays fixed for the whole object track whereas the camera frame C moves relative to the object map frame M:

$$\mathbf{T}_{M_i C_j} = const. \quad \forall i = \{0, \ldots, T\} \tag{4.3}$$

We set the camera frame for the initial time step $t = 0$ to the origin of the map frame, i.e.

$$\mathbf{T}_{M_i C_0} = \mathbf{I} \quad \forall i = \{0, \ldots, T\} \tag{4.4}$$

4.2.3 Bundle Adjustment for Object Tracking

We adapt the general BA optimization scheme of ORB-SLAM2 [27] (refer to section 2.3.4) to the 3D object tracking problem at hand as described in this section. In ORB-SLAM2 "Stereo keypoints are defined by three coordinates $\mathbf{x}_s = (u_L, v_L, u_R)$, being (u_L, v_L) the coordinates on the left image and u_R the horizontal coordinate in the right image." [27, p. 3].

As we use rectified stereo images, these three coordinates are sufficient to uniquely determine a stereo keypoint. For object tracking, we only consider ORB features inside the 2D bounding box of the object detected in the left frame. Then, matches for these keypoints in the right frame are searched and corresponding stereo keypoints are created. In case no match in the right frame is found, a monocular keypoint defined by two coordinates $\mathbf{x}_m = (u_L, v_L)$ is created.

In order to optimize for the transform of the camera frame to the map frame $\mathbf{T}_{MC}$ the so called "motion-only BA" from the ORB-SLAM2 [27] paper is used:

> "Motion-only BA optimizes the camera orientation $\mathbf{R} \in SO(3)$ and position $\mathbf{t} \in \mathbb{R}^3$, minimizing the reprojection error between matched 3D points $\mathbf{X}_i \in \mathbb{R}^3$ in world coordinates and keypoints $\mathbf{x}^i_{(\cdot)}$, either monocular $\mathbf{x}^i_m \in \mathbb{R}^2$ or stereo $\mathbf{x}^i_s \in \mathbb{R}^3$, with $i \in \mathcal{X}$ the set of all matches:
>
> $$\{\mathbf{R}, \mathbf{t}\} = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum_{i \in \mathcal{X}} \rho \left(\left\| \mathbf{x}^i_{(\cdot)} - \pi_{(\cdot)} \left(\mathbf{R}\mathbf{X}^i + \mathbf{t} \right) \right\|^2_{\Sigma} \right) \tag{4.5}$$
>
> where ρ is the robust Huber cost function and Σ the covariance matrix associated to the scale of the keypoint. The projection functions $\pi(\cdot)$, monocular π_m and rectified stereo π_s, are defined as follows:
>
> $$\pi_m \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}, \pi_s \left(\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix} \tag{4.6}$$
>
> where (f_x, f_y) is the focal length, (c_x, c_y) is the principal point and b the baseline, all known from calibration." [27, p. 4]

We run a separate BA optimization for each object. We get the pose of the camera frame relative to the map frame as result of this optimization. As we build the map only based on keypoints that are located inside the 2D bounding box of the object, we can assume that most map points belong to the same object and thus we define the point cloud of all map points as object point cloud. We can calculate the relative pose of the camera with respect to the median of this object point cloud for every frame. The median of the object point cloud is defined as the point-wise median of the three dimensions:

$$\mathbf{p}_{median} = \begin{bmatrix} x_{median} \\ y_{median} \\ z_{median} \end{bmatrix} = \begin{bmatrix} \text{median}(x_{list}) \\ \text{median}(y_{list}) \\ \text{median}(z_{list}) \end{bmatrix} \tag{4.7}$$

where x_{list}, y_{list} and z_{list} are ordered lists of the x, y and z coordinates of the set of map points $\mathcal{X}_{map} \in \mathbb{R}^3$ in the map frame M. Using the median of the point cloud as a representative object position instead of using the mean of the point cloud makes the algorithm more robust towards outliers. We analyze the impact of using median instead of mean on the 3D tracking performance in section 5.3.1.

As we do not use any shape priors or similar pieces of information, the orientation of the object is unknown and we set the related rotation matrix $\mathbf{R}_O = \mathbf{I}$ to identity. We define the resulting transformation matrix from object frame O, attached to the median of the object point cloud, into the map frame M as:

$$\mathbf{T}_{MO} = \begin{bmatrix} \mathbf{R}_O & \mathbf{p}_{median} \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.8}$$

We can calculate the pose of the object point cloud median relative to the camera with the following equation:

$$\mathbf{T}_{CO} = \mathbf{T}_{MC}^{-1}\mathbf{T}_{MO} = \mathbf{T}_{CM}\mathbf{T}_{MO} \tag{4.9}$$

where $\mathbf{T}_{MO}$ is given by equation 4.8 and $\mathbf{T}_{MC}$ is given as result of "motion-only BA" (refer to equation 4.5).

This is the main difference to plain ORB-SLAM2 [27] where keypoints of the whole frame are used and thus the optimized camera pose is expressed with respect to the whole (most probable static) environment and not with respect to an object. In our approach, the tracked object can be either static or dynamic with respect to the static world as this has no influence on the relative pose between object and camera. This means that the object tracking is independent of the absolute object movement.

4.3 Description of 3D Tracking Algorithm Implementation

We add an additional member variable that stores a C++ vector of physical objects to the ORB-SLAM2 [27] system class and define a new class 'Object' as described in subsection 4.3.1.

4.3.1 C++ Object Class

Each object stores its class ID, tracking ID and a vector containing the estimated camera poses with respect to the object map for all frames in which the respective object is tracked successfully. Each object has its own map (containing all map points) and the corresponding related mapping thread.

4.3.2 Multithreading Architecture

When a new object is detected in the image plane by the CenterTrack [15] network, a new map for this object is created and a thread to perform the mapping for this object is spawned. The tracking of the camera with respect to this object is performed in the main thread of ORB-SLAM2 [27]. This is the same thread in which also the camera tracking with respect to the static world is performed.

4.3.3 Object Tracking and Mapping

Tracking of the camera with respect to the object is based on the tracking of ORB keypoints, as in standard ORB-SLAM2 [27]. In contrast to standard ORB-SLAM2 we use only the keypoints

inside the respective 2D bounding box for tracking. On implementation level, all keypoints that are located outside the 2D bounding box are removed from the tracking optimization.

Adaptions to Plain ORB-SLAM2 Logic for Object Tracking

This section summarizes the main adaptions to the plain ORB-SLAM2 [27] logic that are necessary to enable object tracking based on BA using map points that get projected into the 2D bounding box of the tracked object.

In standard ORB-SLAM2 [27] a new keyframe is only inserted if certain conditions are met. The most relevant conditions are as follows [27] [41]:

- more than 20 frames must have passed from the last global relocalization [41]
- current frame tracks at least 50 points [41]
- current frame tracks less than 90% points than the reference keyframe [41]
- the number of tracked close stereo keypoints drops below a certain threshold [27]

As object tracks are rather short, computational constraints that prohibit using a large number of keyframes do not come into play. Therefore, we use every frame as a keyframe for object tracking in order to use as much information as possible.

ORB-SLAM2 [27] requires at least 500 keypoints to do stereo initialization. As we build our map not based on the whole frame, but only on the 2D object bounding box, the amount of keypoints is significantly lower for our approach. Therefore, we change this threshold from 500 keypoints to 50 keypoints.

In case tracking based on the motion model does not succeed, ORB-SLAM2 [27] tries to track the reference key frame. It searches for ORB matches between the current frame and the reference keyframe. If less than 15 matches are found, ORB-SLAM2 aborts the tracking. We change this threshold from 15 to 10 matches as the object maps in general will be much sparser than the static maps that are used by original ORB-SLAM2.

ORB-SLAM [41] and ORB-SLAM2 [27] track the camera pose based on the local map after an initial pose estimation from the last frame. If a relocalization was performed recently, ORB-SLAM2 [27] expects at least 50 inlier matches, respectively 30 inlier matches if the last relocalization was not performed recently. Similar to the previously described thresholds, an adaption to the sparser object maps compared to the static map is required. We adapt the number of inlier thresholds to 15 and 10 respectively.

Because we do not build up a global map in object tracking, but only a map that can be interpreted as a sparse reconstruction of the tracked object, relocalization on the map does not make sense in contrast to standard ORB-SLAM2 [27] (where one can re-localize the camera on the map of the static world). Therefore, we re-initialize lost object tracks that are shorter than 5 frames directly and do not attempt to relocalize the camera on the map. As we want to use every frame as a keyframe, we do not perform keyframe removal compared to the standard ORB-SLAM2 algorithm [27].

Table 4.1 summarizes the adaptions mentioned in section 4.3.3.

Table 4.1: Adaptations for Object Tracking and Object Mapping.

Parameter	Plain ORB-SLAM2 [27]	Modification for Object Tracking (**Ours**)
Keyframe Insertion Conditions	Conditions apply	Use every keyframe
Minimum number of keypoints for stereo initialization	500	50
Minimum number of matches for reference keyframe tracking	15	10
Inlier threshold for camera pose tracking	30	10
Inlier threshold for camera pose tracking after relocalization	50	15
Relocalization after track loss	Yes	No
Keyframe Removal	Yes	No

5 Evaluation

We use the **model LVIS + COCO** as described in chapter 3 with an object score threshold of 0.2, $K = 100$ and a NMS threshold of 0.5 (best hyperparameter setting) for all evaluations in this chapter.

We evaluate the impact of excluding the keypoints inside the bounding boxes of specific LVIS [17] classes in section 5.1.

We evaluate the performance of our 3D Tracking approach on the KITTI Tracking [4] dataset. As only the classes car and pedestrian are labeled in this dataset but our approach is able to detect all 1230 LVIS v0.5 [17] classes, we can only evaluate car and pedestrian quantitatively (see section 5.3). Nevertheless, we give some qualitative evaluation of the performance on other LVIS [17] classes in section 5.2.

5.1 Removal of Object Regions from Feature Extraction

There are three main scenarios in which semantic information is used in current research of visual SLAM (refer to section 2.3.3). One of these is the removal of object regions from the feature extraction. Features on dynamic objects are outliers in the camera trajectory optimization process due to its static world assumption. Therefore identifying dynamic objects through semantic information and subsequently excluding the features on them from the optimization process could help to improve the accuracy of camera trajectory estimates.

Our 2D tracker network does not distinguish between static and dynamic objects. Nevertheless, some of the LVIS classes like cars or persons can be considered as potentially moving. Therefore, we evaluate the impact of excluding the features on cars and persons on the accuracy of the camera trajectory estimates of ORB-SLAM2 [27] on the KITTI [4] odometry training dataset. Table 5.1 shows the relative translational error (%) of the training sequences for plain ORB-SLAM2 [27] and for a modified version of ORB-SLAM2 where all features inside the 2D bounding boxes of cars respectively persons are removed. For some sequences plain ORB-SLAM2 [27] works better, for some sequences the removal of car features improves the error a little bit and for others the removal of person features leads to a small improvement. Considering the fact that ORB-SLAM2 [27] is a multi-threading system and thus the results of it will vary a little bit between two runs on the same sequence, we do not assess these small differences as significant. On top of that, the average over all sequences is also differing only by 0.01%.

Table 5.2 shows that the impact of the feature removal on the rotational error ($^\circ$/100 m) behaves similarly to the impact on the translational error. For some sequences the removal of car features is beneficial, for others the removal of person features and for some keeping all

Table 5.1: Comparison of translational error (%) of different variations of ORB-SLAM2.

Sequence	Plain ORB-SLAM2	Removing Car Features	Removing Person Features
00	**0.70**	0.72	0.71
01	**1.35**	1.36	1.42
02	0.78	**0.74**	0.77
03	0.71	**0.68**	0.74
04	0.45	**0.43**	0.49
05	0.40	0.41	**0.39**
06	0.55	0.64	**0.52**
07	**0.48**	0.52	0.52
08	1.03	1.02	**1.00**
09	0.88	0.87	**0.84**
10	**0.60**	0.64	0.64
Average All Sequences	**0.72**	0.73	0.73

features leads to a better performance than the removal.

Thus, we conclude that removing the features on cars and persons extracted by our 2D tracker network has no significant impact, neither positive nor negative, on the accuracy of camera trajectory estimates in the KITTI [4] dataset. Therefore, we focus in the remainder of the thesis on our approach to use 2D tracking information for 3D object tracking of various classes in parallel to the camera trajectory estimation of ORB-SLAM2 [27] (refer to chapter 4). We evaluate this approach extensively in the following sections 5.2 and 5.3.

5.2 Qualitative Evaluation 3D Object Tracking

This section contains a qualitative evaluation of different LVIS [17] classes. Subsection 5.2.1 gives a qualitative evaluation of the KITTI [4] ground truth classes car and pedestrian (i.e. LVIS [17] class person). Subsection 5.2.2 gives a qualitative evaluation of other LVIS [17] classes that are present in the KITTI Tracking [4] dataset.

We set the number of extracted features per frame N_{feat} of our modified ORB-SLAM2 [27] system to $N_{feat} = 10000$ for all following experiments and evaluations in this section.

In the following two subsections (5.2.1 and 5.2.2), we will show a few example trajectories of object instances of different classes. We show 2D frames with the corresponding 2D bounding box detection. Please note, that only the bounding box of the evaluated object is shown. There might exist a lot more 2D bounding boxes in the frame. But we don't show the 2D detections of other objects than the evaluated one in order to avoid a cluttered visualization. The timestamp of each frame (beginning with 0 s for each track) is displayed in the upper left corner of the 2D frame images. To the right of the 2D frames we show a bird's-eye view of the camera trajectory and object trajectory in world frame. We take the complete camera

Table 5.2: Comparison of rotational error (°/100 m) of different variations of ORB-SLAM2.

Sequence	Plain ORB-SLAM2	Removing Car Features	Removing Person Features
00	**0.25**	0.26	**0.25**
01	0.25	0.27	**0.23**
02	0.25	**0.22**	0.24
03	0.21	**0.18**	**0.18**
04	0.15	0.16	**0.12**
05	**0.16**	**0.16**	**0.16**
06	0.17	0.28	**0.14**
07	**0.27**	0.28	0.28
08	0.32	**0.29**	0.30
09	**0.24**	0.26	0.29
10	**0.26**	0.32	0.31
Average All Sequences	**0.23**	0.24	**0.23**

trajectory (blue curve) from the output of the plain ORB-SLAM2 [27] system (i.e. BA based on features of the whole frame assuming a static world). We show the object trajectory as green curve. We highlight the part of the complete camera trajectory that corresponds to the time steps for which the object track exists with orange.

We calculate the object poses in world frame (required for the object trajectory) as follows: Our 3D object tracking approach (refer to chapter 4) gives the object poses in camera frame $\mathbf{T}_{CO}$ (refer to equation 4.9). Concatenating these poses with the camera poses in world frame (given by ORB-SLAM2 [27]) gives the object poses in world frame:

$$\mathbf{T}_{WO} = \mathbf{T}_{WC}\mathbf{T}_{CO} \tag{5.1}$$

5.2.1 KITTI Ground Truth Classes

Figure 5.1 shows an example trajectory of a car. At the beginning of the scene the tracked car drives on the right lane and the camera vehicle drives on the left lane. This is also clearly visible in the plot of the trajectories. Later in the scene the camera vehicle changes the lane and drives behind the tracked car. Qualitatively, we can also observe this when looking at the trajectories in the bird's-eye view plot: Both trajectories are nearly identical, except that the one of the tracked car is longer because it drives in front of the camera vehicle and accelerates.

Figure 5.2 shows an example trajectory of a pedestrian (i.e. LVIS[17] class person). The pedestrian is crossing the road at the traffic light. The camera vehicle approaches this traffic light and stops a few meters in front of the crossing pedestrian. Qualitatively, the trajectories in the bird's-eye view plot match visually well to the scene that is shown by the 2D frame images.

We conclude that in principle our approach is capable to track cars and pedestrians in 3D. A detailed quantitative evaluation of the 3D tracking performance and accuracy in terms of various metrics is given in section 5.3.

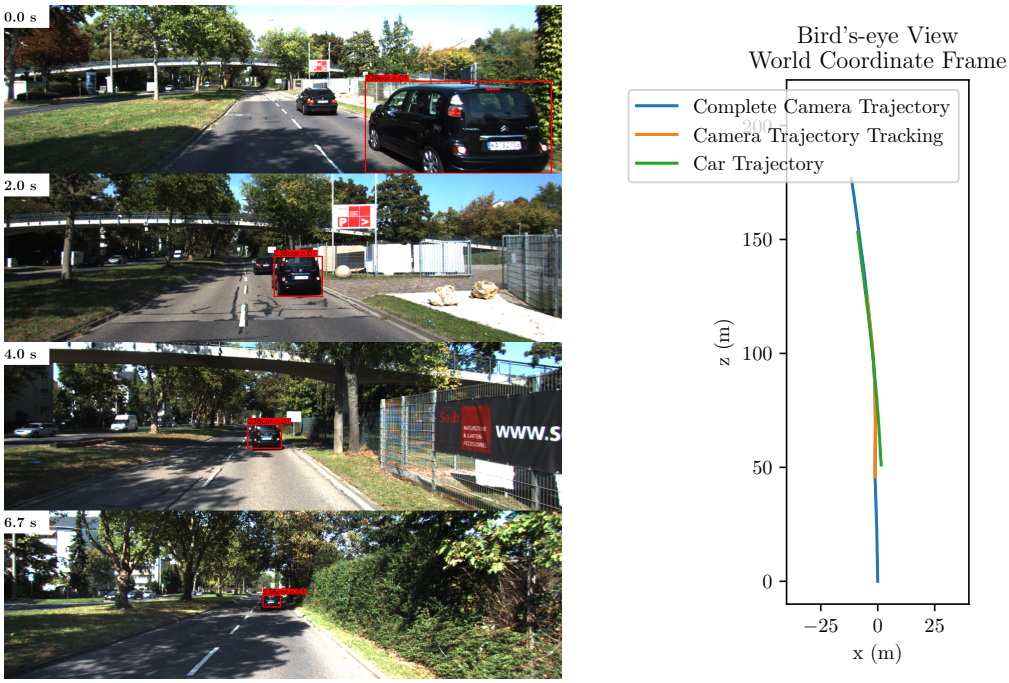


Figure 5.1: Example trajectory of class car.

5.2.2 Other Classes

Table 5.3 shows the number of tracked LVIS classes by our 2D tracker network in comparison to the number of tracked classes by our 3D Tracking algorithm (refer to section 4.2 and 4.3). As all KITTI [4] sequences are from the same domain (traffic scenes), they only contain a subset of the very diverse LVIS [17] classes (63/1230 classes). In theory, our 3D tracking algorithm can track objects with an arbitrary track length. In the following evaluations we consider only 3D tracks with a minimum track length of 5 frames because of the higher importance of long, stable tracks for real world applications. Consequently, we also only consider 2D tracks of the 2D tracker network with a minimum track length of 5 frames.

22 of 25 tracked 2D classes get also tracked at least once in 3D. The 3 classes that are not tracked in 3D are the following ones:

- streetlight.n.01
- ball.n.06
- awning.n.01

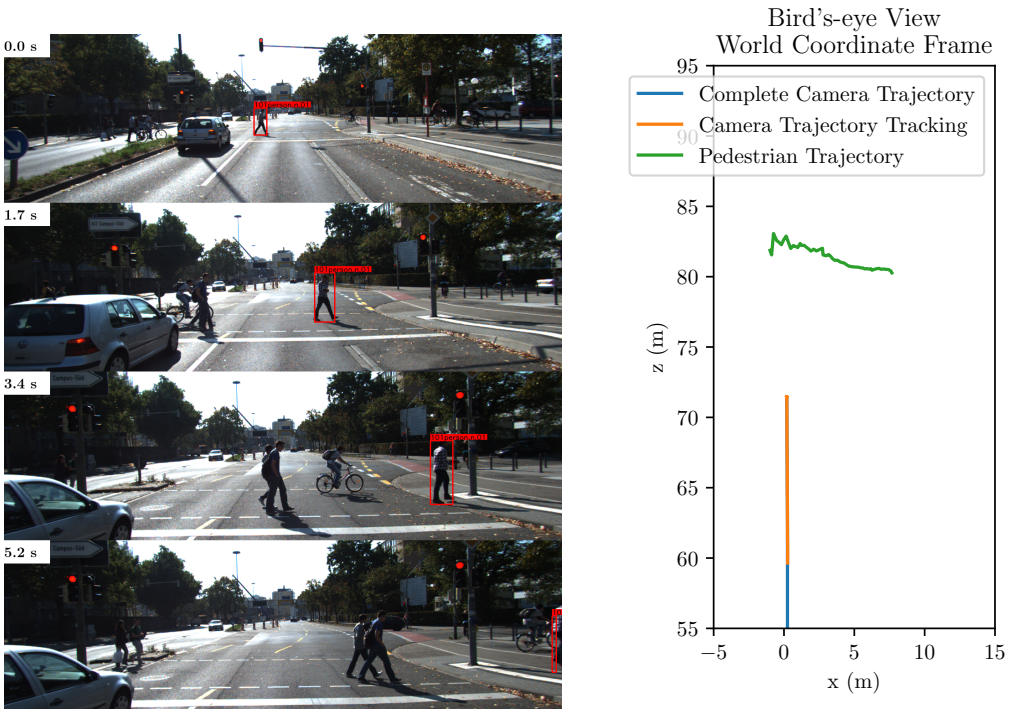


Figure 5.2: Example trajectory of class pedestrian.

The number after n. in the LVIS [17] class names refers to the corresponding WordNet [48] synset which describes the semantic of a word. This is useful in case of words that can have different meanings, e.g. ball which can refer to the ball one can play with (WordNet [48] synset n.06) or to the ball where people dance (WordNet [48] synset n.04). In the following, we will omit the synset specification of classes in all cases, where the referred synset is equivalent to the most frequent meaning of a word, for better readability. In case a word is potentially ambiguous we will give a more detailed definition in brackets, e.g. "pot (flower pot)".

Streetlight and ball are tracked only once in 2D and awning is tracked only twice in 2D, all of them with a maximum track length of 5 or 6 frames. Thus, we cannot conclude if there is a systematic problem in our approach that prevents the 3D tracking of streetlights, balls and awnings or not. Nevertheless, we assume that the reason of the missing 3D tracks for these classes is the lack of sufficiently many and sufficiently long 2D tracks of these classes. We do not assume that the three mentioned classes themselves are particularly difficult to track for our approach.

Table 5.4 shows the LVIS classes for which there is at least one successful estimated 3D track sorted by the number of tracks per class. In this table recall refers to the ratio between predicted 3D tracks and predicted 2D tracks of the respective class:

$$Recall = \frac{N_{3D}}{N_{2D}} \tag{5.2}$$

where N_{3D} is the number of estimated 3D tracks and N_{2D} is the number of predicted 2D

Table 5.3: Comparison of the number of tracked LVIS classes.

	Number of tracked classes
Total number of classes present in LVIS dataset v0.5	1230
Tracked classes in 2D	63
Tracked classes in 2D with at least one track >= 5 frames	25
Tracked classes in 3D (all 3D tracks are >= 5 frames)	22

tracks of the respective class.

As the KITTI [4] Tracking dataset consists of traffic scenes recorded by a car, the three most frequent estimated objects (besides the KITTI ground truth classes car and pedestrian) belong to classes that appear frequently in traffic scenarios:

- bicycle
- traffic light
- bus

The following five classes exhibit a recall value that is significantly lower than the recall value considering all classes (45.38%):

- traffic light
- taillight
- bag
- license plate
- ashcan

We hypothesize that these objects are tracked less frequently successful in 3D because these are rather small objects. This leads to smaller 2D bounding boxes and thus less features for the 3D tracking algorithm. To verify this hypothesis we show the tracked classes along with the average edge length of their 2D bounding boxes sorted by ascending edge length in table 5.5. We mark the five classes with low recall by writing their name in **bold** font. Four out of the five classes have the four lowest average 2D bounding box edge lengths. The class ashcan has a little bit higher average 2D bounding box edge length than some classes with a higher recall. Nevertheless, the difference is not very big (only around 12 pixels between fireplug and ashcan). Furthermore, the 3D track of the ashcan is a false positive (because the 2D detector classified a chair as ashcan). Thus, we consider our hypothesis as verified by table 5.5.

Figures 5.3, 5.4, 5.5, 5.6 and 5.7 show example trajectories of the five most frequently tracked classes in 3D (except the KITTI [4] ground truth classes car and pedestrian). Refer to section 5.2 for a detailed description of the derivation and meaning of the differently colored

Table 5.4: 3D Tracking: Successfully tracked LVIS classes.

Class	Number of tracked 3D objects	Recall
All Classes	393	45.38%
Bicycle	107	58.47%
Traffic Light	60	26.91%
Bus	31	75.61%
Street Sign	26	61.90%
Pot (flower pot)	26	66.67%
Truck	20	80.00%
Train	19	86.36%
Chair	19	65.52%
Taillight	13	11.30%
Telephone Pole	13	72.22%
Bag	13	31.71%
Stop Sign	8	53.33%
Motorcycle	8	80.00%
Parking Meter	7	63.64%
Cone	6	100.00%
License Plate	6	23.08%
Umbrella	3	60.00%
Fireplug	3	50.00%
Signboard	2	66.67%
Skateboard	1	100.00%
Dog	1	100.00%
Ashcan	1	25.00%

trajectories in these plots. Qualitatively, we can see that the trajectories of moving objects, like the bicycle and the bus fit well to what we can see in the 2D images. For standing objects, like traffic light, street sign and pot, we expect a stable static object position in world frame. This is also fulfilled except some small pseudo movement due to inaccuracies in the tracking of around 1 to 2 meters. We conclude that our approach is able to localize classes that are not part of the KITTI [4] Tracking challenge.

Figure 5.8, figure 5.9 and figure 5.10 show example trajectories of three of the four classes with the smallest average bounding box edge length. One example trajectory of the class traffic light (the remaining one of the four classes with smallest average bounding box edge length) is shown by figure 5.4. Even though the recall of these classes with smaller object sizes is below the recall considering all classes (see table 5.5), the accuracy of the successful 3D tracks is similar to the one of classes with bigger object sizes. We can see this by qualitative visual comparison of the bird's-eye view trajectory and of what is visible in the corresponding 2D frame images. The trajectory of the taillight in figure 5.8 resembles the trajectory of the

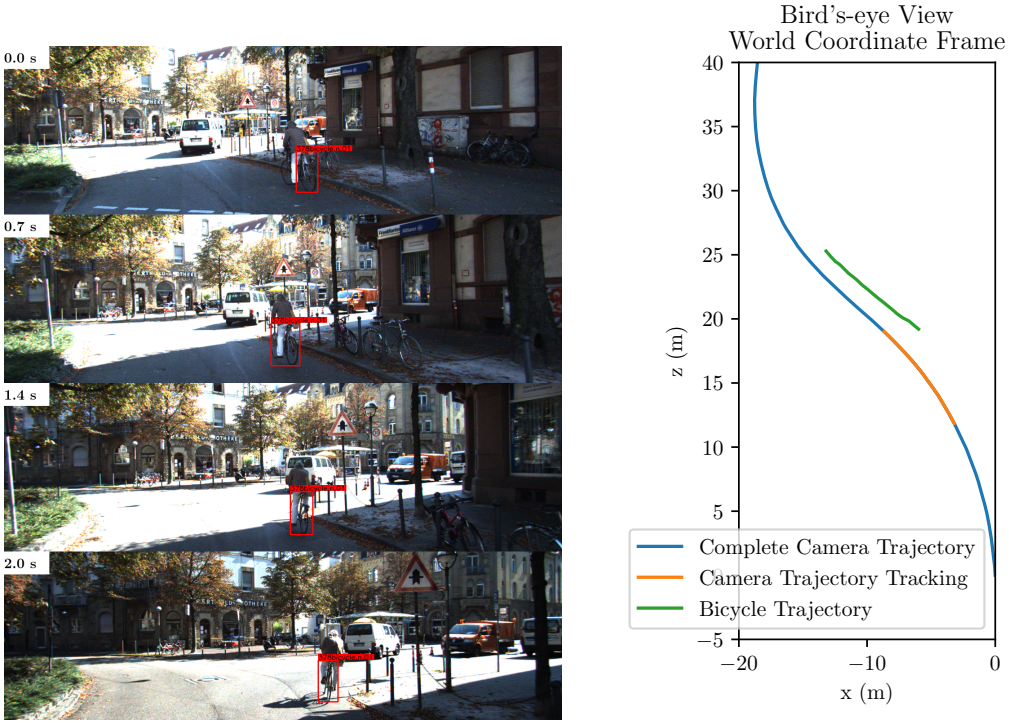


Figure 5.3: Example trajectory of class bicycle.

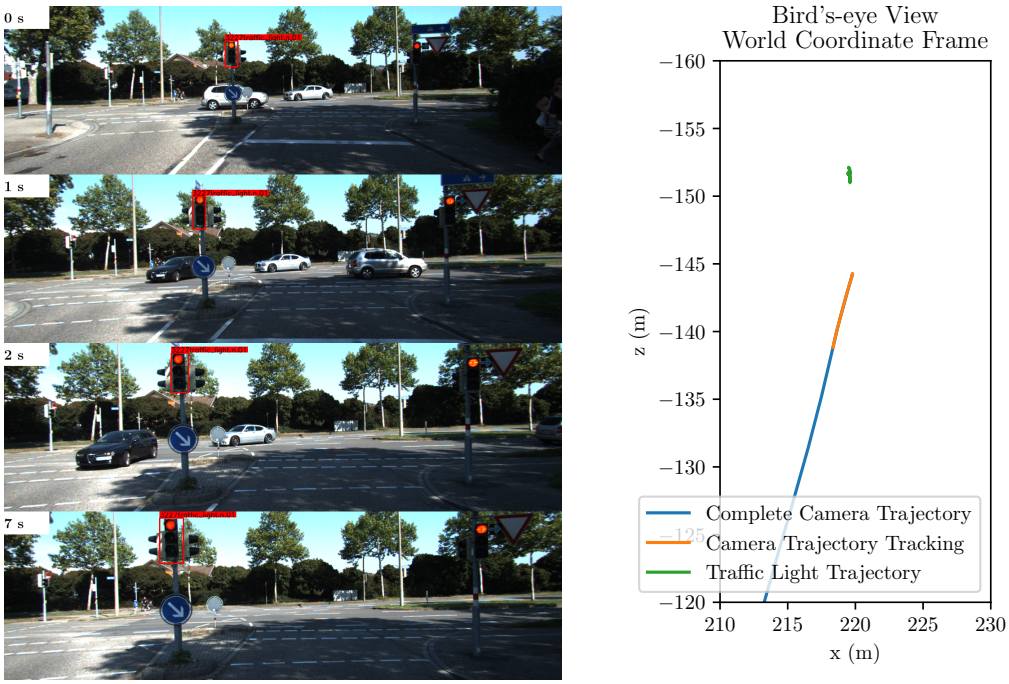


Figure 5.4: Example trajectory of class traffic light.

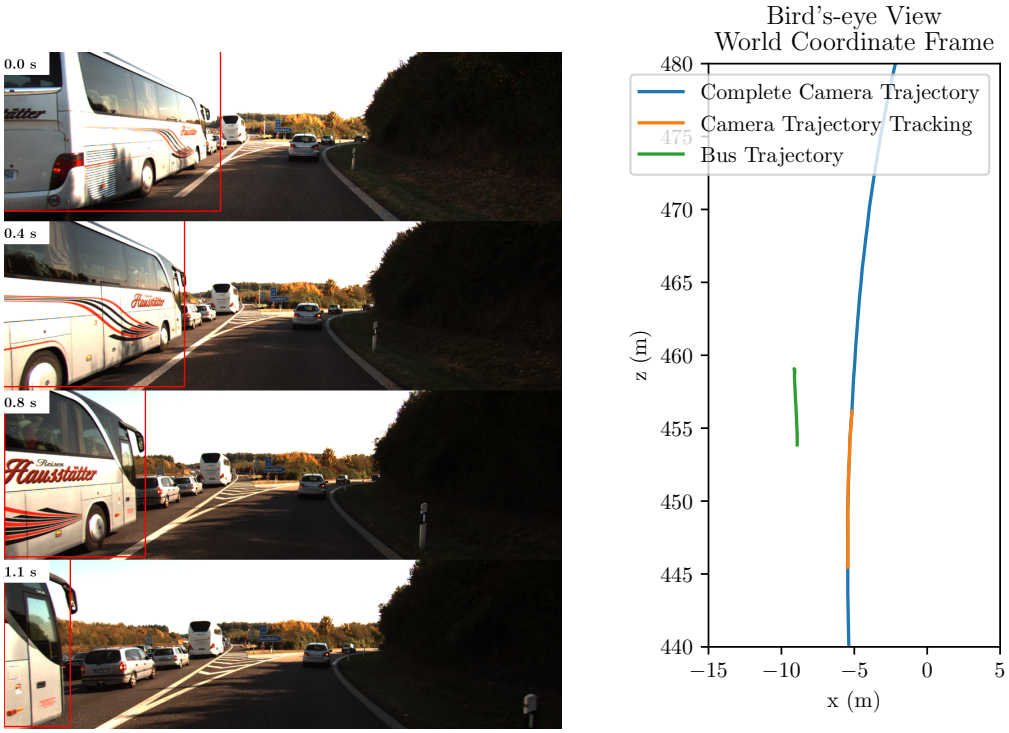


Figure 5.5: Example trajectory of class bus.

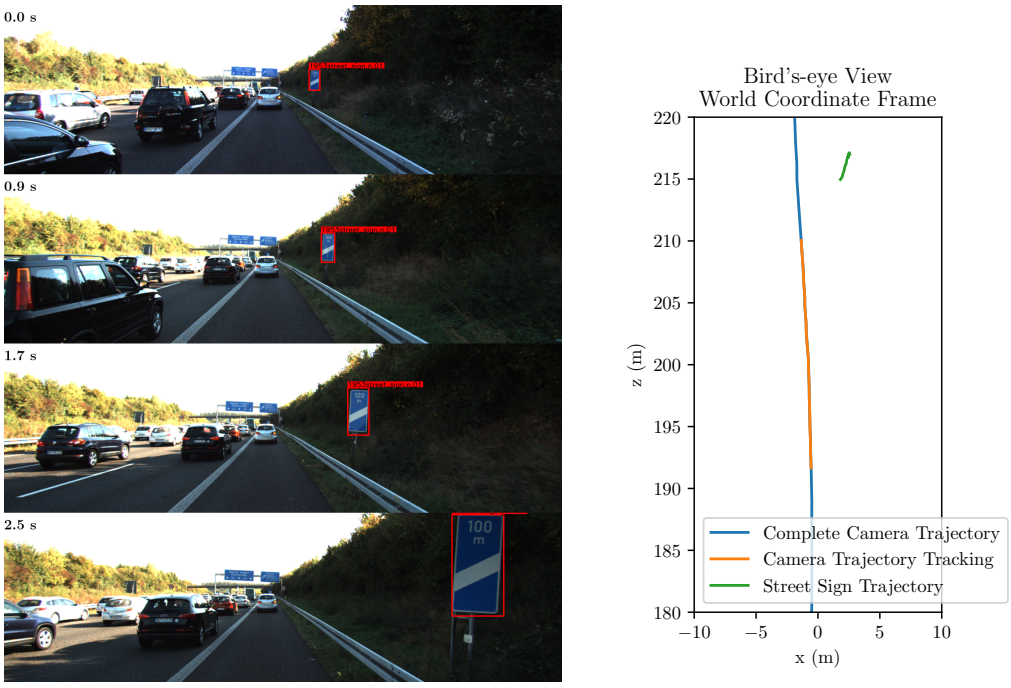


Figure 5.6: Example trajectory of class street sign.

Table 5.5: 3D Tracking: Object classes and corresponding average 2D bounding box edge length.

Class	Mean of $\sqrt{w_{bbox} * h_{bbox}}$	Recall
Taillight	**19.44**	**11.30%**
License Plate	**32.00**	**23.08%**
Traffic Light	**33.66**	**26.91%**
Bag	**39.52**	**31.71%**
Fireplug	44.32	50.00%
Skateboard	46.79	100.00%
Parking Meter	48.83	63.64%
Cone	54.30	100.00%
Telephone Pole	55.51	72.22%
Stop Sign	56.24	53.33%
Ashcan	**56.34**	**25.00%**
Street Sign	63.25	61.90%
Chair	77.06	65.52%
Dog	77.95	100.00%
Bicycle	85.96	58.47%
Signboard	86.43	66.67%
Motorcycle	94.35	80.00%
Pot	96.77	66.67%
Umbrella	112.26	60.00%
Truck	143.95	80.00%
Bus	165.08	75.61%
Train	177.23	86.36%

red car that drives on the middle lane, next to the lane of the camera vehicle. The track of the license plate in figure 5.9 shows a similar traffic situation. Also this trajectory looks plausible as the vehicle, the license plate belongs to, moves next to the camera vehicle. Figure 5.10 shows an example trajectory of the class bag. It shows that our 3D tracking approach is also able to track objects that move orthogonal to the camera. We conclude that the 3D tracking of small objects is in principle possible by our approach. Nevertheless, reaching a higher recall (proportion of 3D tracks to 2D tracks) is desirable and might be tackled in future works e.g. by forcing the feature extractor to extract a minimum amount of features per bounding box.

5.3 Quantitative Evaluation 3D Object Tracking

The KITTI Tracking [4] dataset provides locations and dimensions of objects of the classes car and pedestrian in 3D space, specified in the camera coordinate frame. The location refers to the center point of the related 3D bounding box whereas the dimensions specify the size of

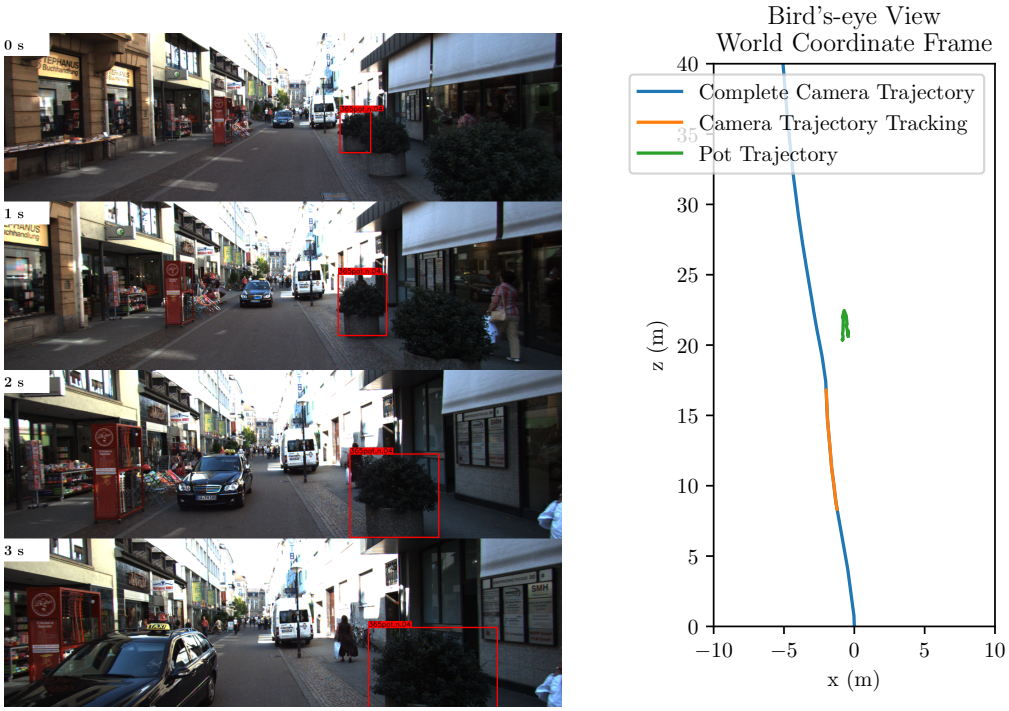


Figure 5.7: Example trajectory of class pot.

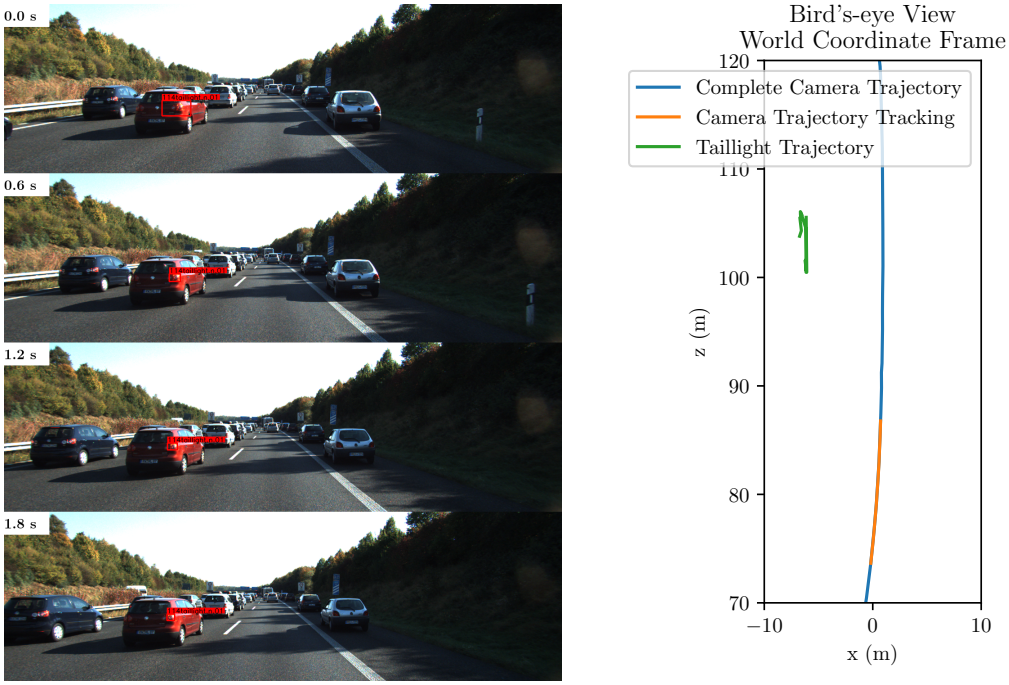


Figure 5.8: Example trajectory of class taillight.

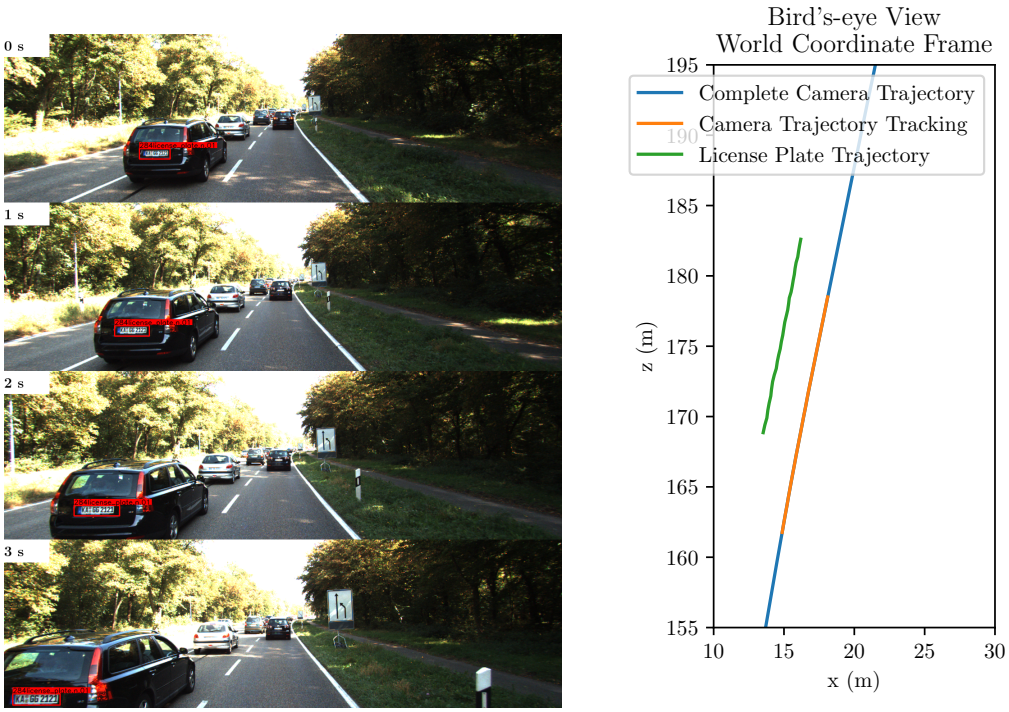


Figure 5.9: Example trajectory of class license plate.

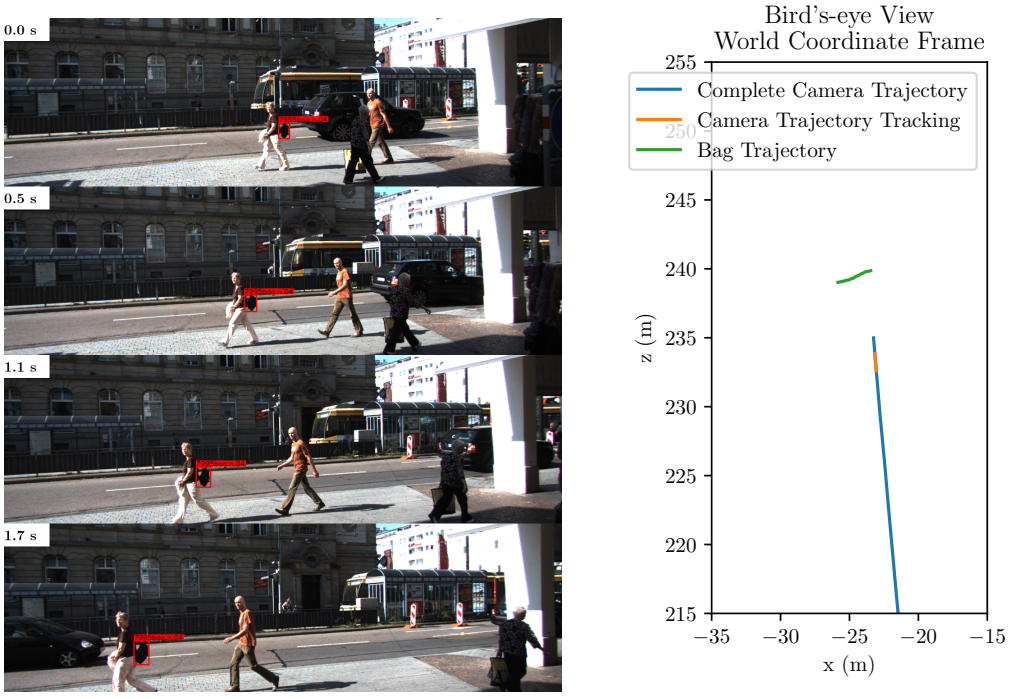


Figure 5.10: Example trajectory of class bag.

this bounding box. All quantitative experiments only consider predicted 3D tracks with a minimum length of 5 frames because of the higher importance of long, stable tracks for real world applications.

One important parameter of ORB-SLAM2 [27] is N_{feat} which sets the maximum number of ORB features that are extracted by the feature detector per image. Increasing this parameter leads to more extracted features on the whole image domain and thus indirectly also to more features inside the 2D object bounding boxes. We analyze the impact of this parameter on the different metrics in the following subsections.

The following sections show the analysis of the object tracking in 3D based on different metrics. The evaluation of different metrics helps to get a deeper understanding of the strengths and weaknesses of the proposed system.

5.3.1 Average Euclidean Distance Between Estimated and Ground Truth Locations per Frame

In this section, we assess the performance of the 3D tracking by calculating the Euclidean distance between the ground truth object location and the estimated object location for every frame of a 3D track. Then, we average these distances over the number of tracked frames. We assess each track independently.

We match estimated and ground truth object tracks based on the 2D bounding box overlap in the image plane. We compute the IoU of a predicted bounding box with all ground truth bounding boxes. Then, we match the predicted object with the ground truth bounding box with which it shares the biggest IoU value. In case there is no IoU value bigger than 0.5, we ignore the predicted object for this evaluation as it can not be matched adequately with any ground truth object.

The following experiment shows the performance differences induced by either picking the arithmetic mean of the object point cloud as object position, or by picking the point-wise median as object position. We execute the experiment with the default value for the number of ORB features $N_{feat} = 2000$ per image. Table 5.6 compares the average Euclidean distance e_{dist} between predicted and ground truth location averaged over all tracks:

$$e_{dist} = \frac{1}{N} \sum_{i=0}^{N} \frac{1}{F_i} \sum_{j=0}^{F_i} \left\| \mathbf{x}_{ij} - \hat{\mathbf{x}}_{ij} \right\|_2 \tag{5.3}$$

where N is the number of tracks, F_i is the number of frames of the i-th track, $\mathbf{x}_{ij}$ is the predicted position of the object center of track i in frame j in camera coordinates and $\hat{\mathbf{x}}_{ij}$ is the corresponding ground truth location, specified in camera coordinates as well.

The average distance error when using point-wise median is significantly lower than when using arithmetic mean. This is due to the fact that the arithmetic mean is sensitive to outliers whereas the point-wise median is robust to outliers. Because we consider all keypoints inside a 2D bounding box as object points, there is a certain probability that some background keypoints near the border of the bounding box get wrongly assigned to the object. Point-wise median suppresses these outlier points. Another approach to a priori avoid these outlier

Table 5.6: Average Euclidean distance between ground truth and predicted object position.

Object point cloud representation	Average Euclidean distance e_{dist}
Arithmetic mean	2.53 m
Point-wise median	1.73 m

points would be to use segmentation masks instead of bounding boxes. To do so, one would have to replace our 2D tracker network by a network that predicts segmentation masks instead of 2D bounding boxes. The general framework would work in the same way as with 2D bounding boxes.

In the following, the impact of the distance of tracked objects from the camera on the average Euclidean distance error e_{dist} is evaluated. We evaluate it with two different settings for N_{feat}. This parameter is set to 2000, respectively 5000 to analyze the impact of the number of features on the 3D tracking performance. Figure 5.11 shows the Euclidean distance error e_{dist} with respect to the distance of the object from the camera when setting $N_{feat} = 2000$. Figure 5.12 shows the Euclidean distance error e_{dist} with respect to the distance of the object from the camera when setting $N_{feat} = 5000$. In general, it is notable that the distance error e_{dist} is not growing significantly with increasing distance of the objects from the camera. Increasing the number of ORB features per image from 2000 to 5000 leads to more tracked objects that are far away from the camera (30 - 60 m). The reason is that objects that are further away are represented by smaller bounding boxes and thus less features. Increasing the total amount of features helps to reach a sufficient number of features also in smaller 2D bounding boxes. There is a small increase in wrong tracks with very high errors e_{dist} when using more features, especially for objects that are far away (> 30 m). The amount of these wrong tracks is very small but the improvement regarding the tracking range is significant. Thus, we propose that the usage of a high number of features (high value of N_{feat}) is overall beneficial for the 3D tracking task. The downside of extracting a higher number of ORB features is the increase in computation time. This could be improved in future work by running separate feature extractors only inside the bounding boxes instead of increasing the total number of features per image.

In the following experiments we consider three different parameter settings for the number of features per frame:

- $N_{feat} = 2000$
- $N_{feat} = 5000$
- $N_{feat} = 10000$

Table 5.7 shows the number of tracked cars and pedestrians in the KITTI [4] Tracking dataset in order to evaluate the impact of N_{feat} on the number of tracked objects of the KITTI [4] ground truth classes. Increasing the number of features gives an improvement regarding the number of tracked pedestrians of a factor of around 7 and regarding the number of tracked cars of a factor of around 3.

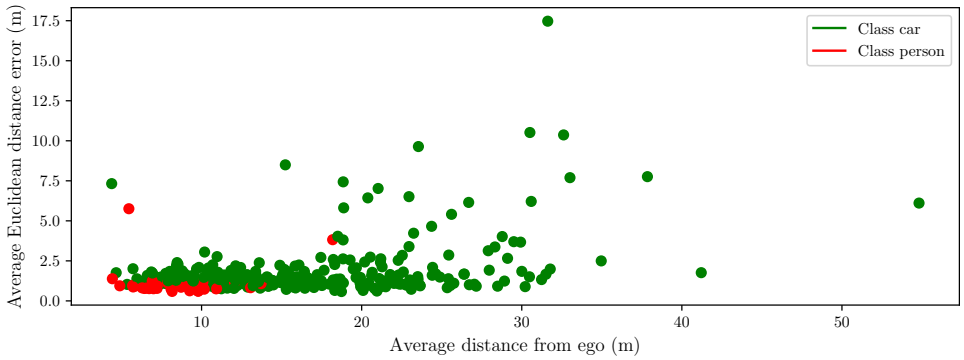


Figure 5.11: Euclidean distance error e_{dist} with respect to the distance of the object from the camera ($N_{feat} = 2000$).

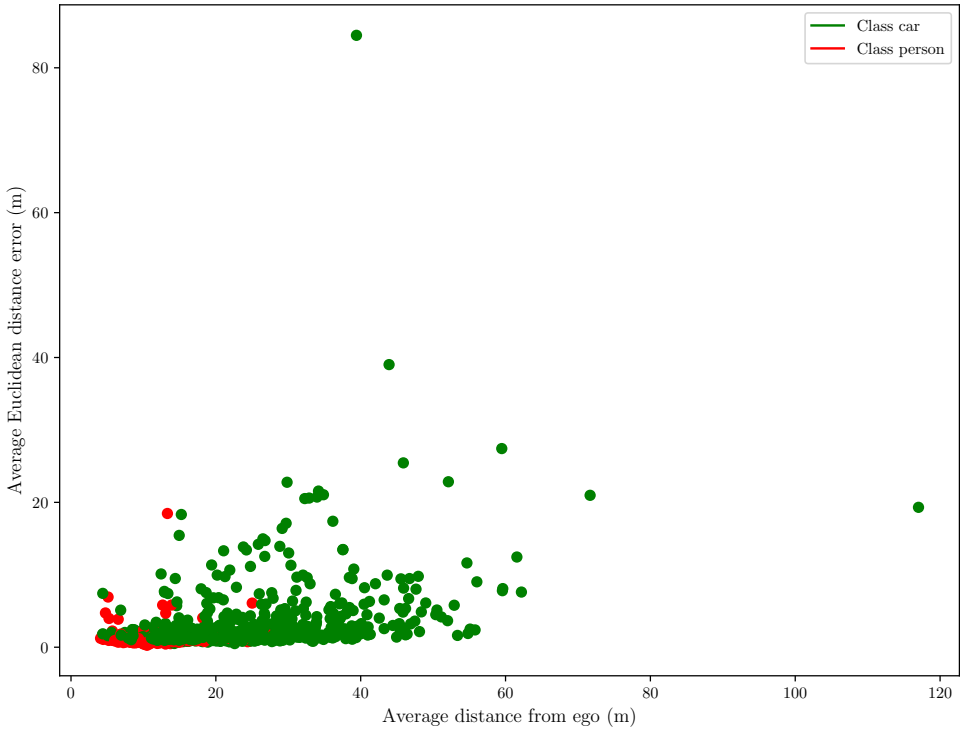


Figure 5.12: Euclidean distance error e_{dist} with respect to the distance of the object from the camera ($N_{feat} = 5000$).

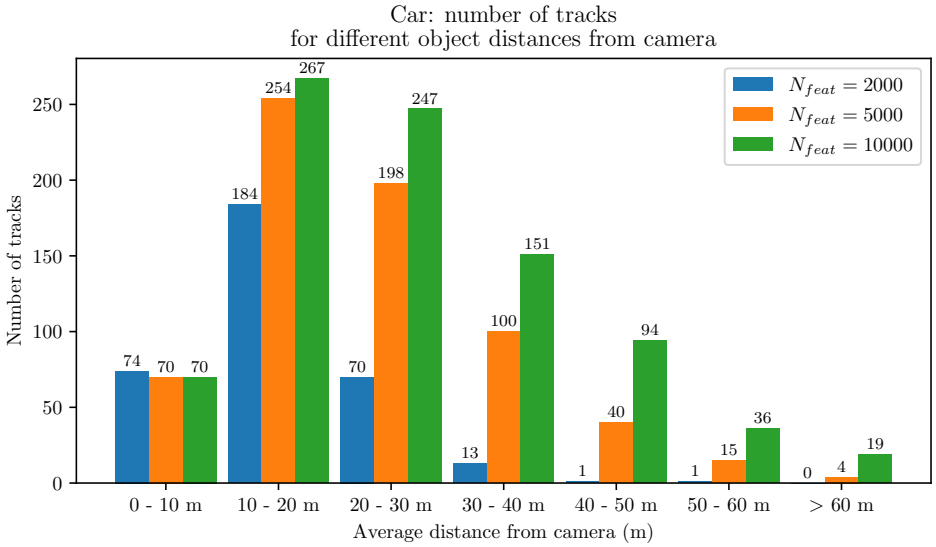
Table 5.7: Number of tracked tracked cars and pedestrians depending on parameter N_{feat}.

	$N_{feat} = 2000$	$N_{feat} = 5000$	$N_{feat} = 10000$
Number of tracked pedestrians	34	160	259
Number of tracked cars	343	681	884

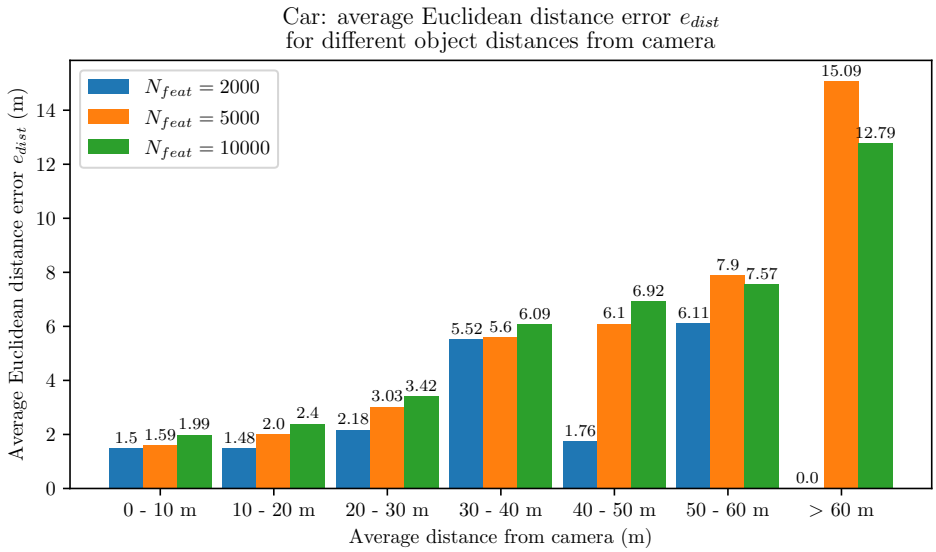
Figure 5.13a shows the number of tracked ground truth cars for different distances from the camera. Increasing the amount of features per image N_{feat} leads to a significant increase in the number of tracks, especially for objects that are further away from the camera. The main reason is that cars that are further away are projected into smaller 2D bounding boxes and thus contain less features. This can be compensated by increasing the number of features. Figure 5.13b shows the average Euclidean distance error e_{dist} for different distances from the camera. Increasing the number of features does not have a big negative impact on the average error for close objects. Objects that are further than 40 m away from the camera exhibit significantly larger errors on a first glance. But as there are only two tracked objects for $N_{feat} = 2000$ for these distances, a fair comparison is impossible. This means that the increase in tracked ground truth objects (even with a significant error) when setting $N_{feat} = 5000$ or even setting $N_{feat} = 10000$ is a good trade-off because having tracks with an average error of roughly 6 to 7 m is still better than tracking nearly no objects at this distance. A downstream application that requires a certain accuracy, could easily adopt to this error by simply ignoring tracks of objects that are further away from the camera than a user defined threshold.

Figure 5.14 shows the same comparison for pedestrians. Increasing the amount of features leads to around 4 times more tracked pedestrians in the range from 0 to 10 m from the camera and around 16 times more tracked pedestrians in the range between 10 and 20 m from the camera. With the higher number of features even 10 pedestrians, respectively 39 pedestrians, in a range between 20 and 30 m from the camera get tracked (the system did not track any pedestrians in this range with $N_{feat} = 2000$!). Analyzing the average Euclidean error e_{dist} shows only a very small decrease in the 3D tracking accuracy when increasing from $N_{feat} = 2000$ to $N_{feat} = 5000$. The increase in average Euclidean distance error e_{dist} for objects with an average distance from the camera between 20 and 30 m when increasing N_{feat} from 5000 to 10000 is significant. Nevertheless, we get around 4 times more pedestrian tracks in this range by increasing the number of features from 5000 to 10000. Thus, a higher amount of features is very beneficial for tracking pedestrians in 3D with our approach.

In order to exclude the impact of additional tracks that are created by increasing the amount of features, we consider only tracks that are present with $N_{feat} = 2000$, $N_{feat} = 5000$ and $N_{feat} = 10000$ in the following analysis. To eliminate also the impact of longer track lengths that are estimated based on a higher number of features, we consider only object positions in the frames that are tracked with all three parameter settings. Figure 5.15 shows the results for the class car. Figure 5.16 shows the results for the class pedestrian. There is only a big difference visible for cars with a distance from the camera > 40 m. As there are only two common tracks in this range, no fair comparison is possible and we can ignore this deviation.

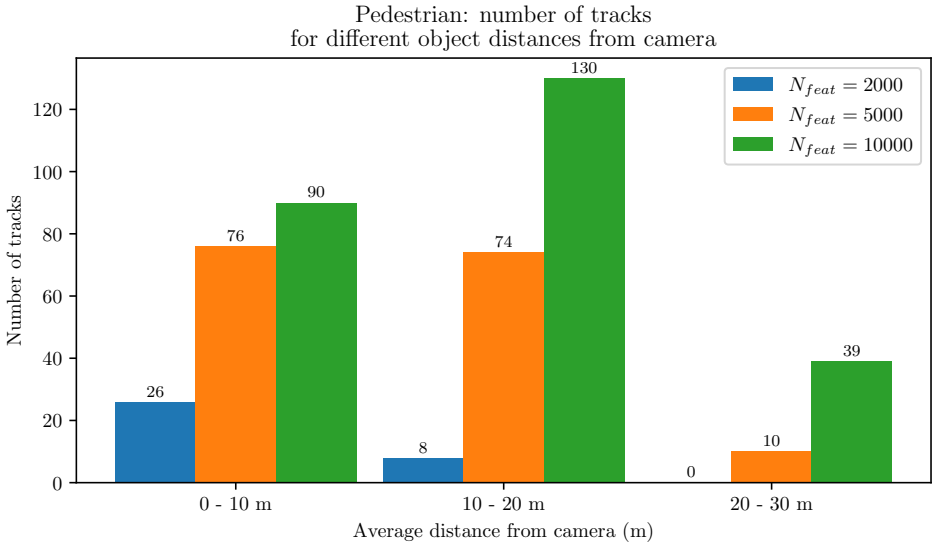


(a) Number of car tracks for different object distances from the camera.

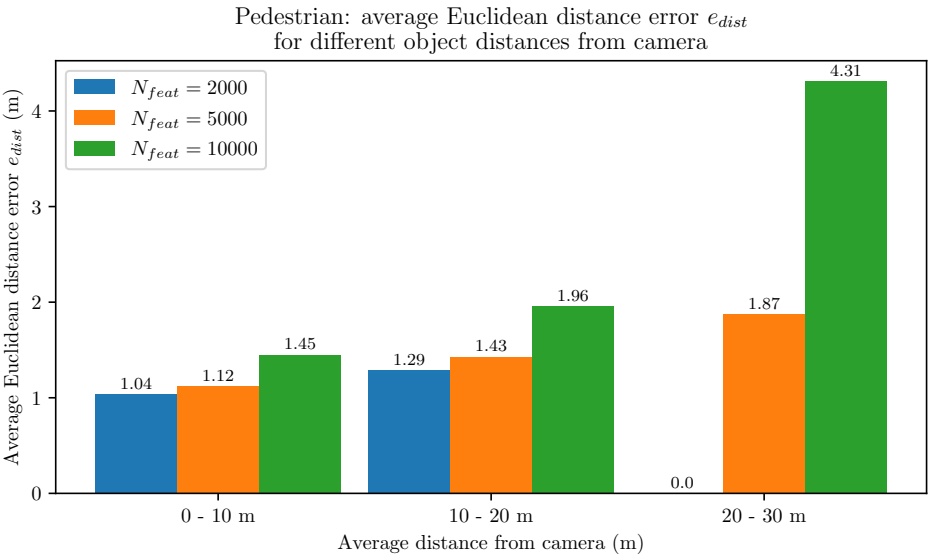


(b) Car: Average Euclidean distance error for different object distances from the camera.

Figure 5.13: Car: comparison of number of tracks and average Euclidean error e_{dist}.



(a) Number of pedestrian tracks for different object distances from the camera.



(b) Pedestrian: Average Euclidean distance error for different object distances from the camera.

Figure 5.14: Pedestrian: comparison of number of tracks and average Euclidean error e_{dist}.

We conclude that the increase of the number of features has no significant negative impact on the 3D position estimation accuracy of objects that are tracked both with a high amount of features ($N_{feat} = 5000$ or $N_{feat} = 10000$) and a low amount of features ($N_{feat} = 2000$).

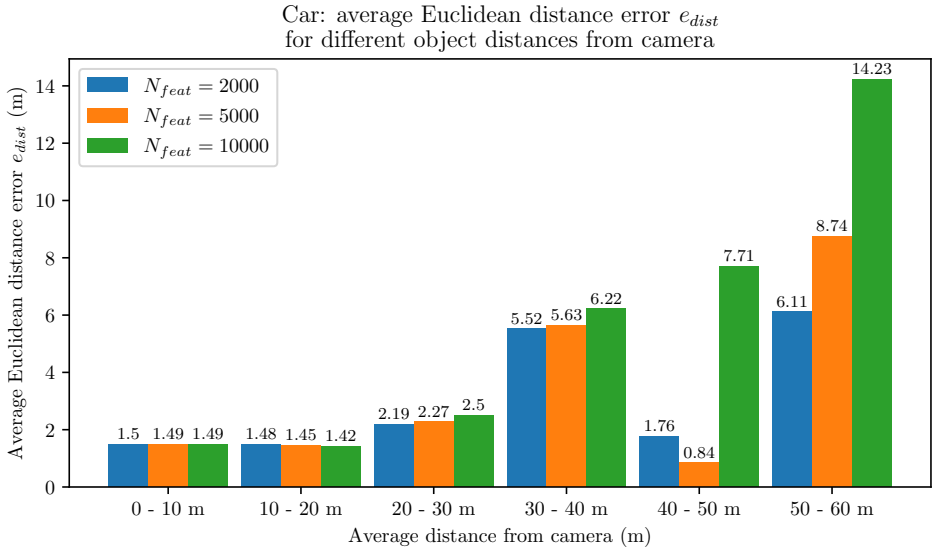


Figure 5.15: Euclidean distance error e_{dist} with respect to the distance of the object from the camera only considering common tracks of all three parameter settings.

5.3.2 Comparison of 2D Tracks and 3D Tracks

The 3D tracking performance of our system strongly depends on a stable 2D tracker. If there is no 2D bounding box tracked by the 2D tracker, no 3D track for the corresponding object will be initialized. Therefore the performance of the 2D tracker must be considered for a fair evaluation of the 3D tracker. Table 5.8 provides such a comparison depending on the different parameter values of N_{feat}. Most of the ground truth objects are tracked in 2D (618/635 for cars and 158/167 for persons). The relative amount of tracked 3D objects strongly depends on the number of extracted features N_{feat}. Using a high amount of features per frame, i.e. $N_{feat} = 10000$ leads to a significant proportion of 2D tracks being also tracked in 3D by our modified ORB-SLAM2 system (530/618 for cars and 129/158 for persons).

5.3.3 Localization Precision Metric

Average Localization Precision (ALP) is a metric to evaluate the performance of 3D object detectors. It evaluates the localization accuracy of the object center points. An object detection is considered as a correct detection (true positive) when the estimated center point is closer to the ground truth center point than a defined threshold (e.g. 1 m or 2 m). The calculation of ALP is equivalent to the calculation of AP as defined in section 3.3 without averaging over different IoU thresholds and replacing the IoU criterion by the localization threshold criterion.

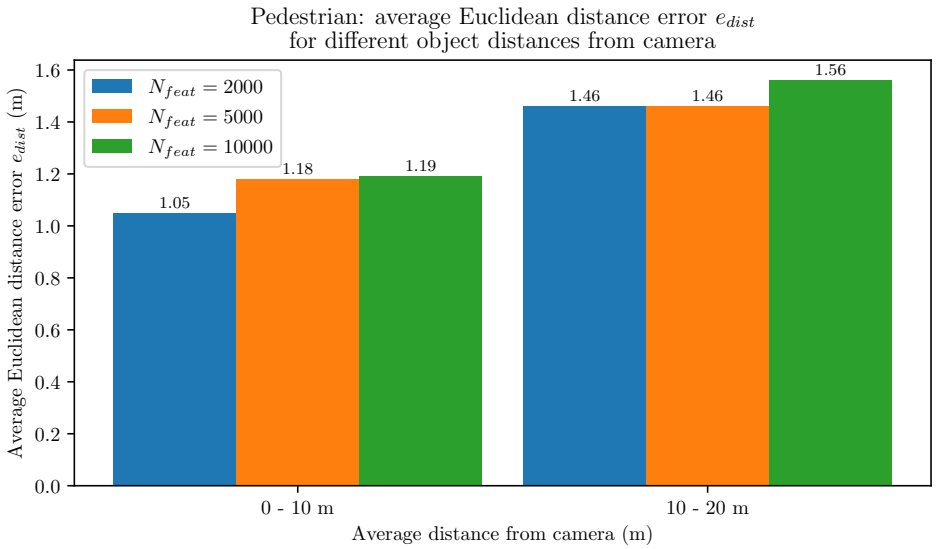


Figure 5.16: Euclidean distance error e_{dist} with respect to the distance of the object from the camera only considering common tracks of all three parameter settings.

Table 5.8: Comparison of number of ground truth objects and tracks in 2D and 3D.

	$N_{feat} = 2000$	$N_{feat} = 5000$	$N_{feat} = 10000$
Ground Truth Car	635	635	635
Estimated 2D Tracks Car	618	618	618
Estimated 3D Tracks Car	324	489	530
Ground Truth Person	167	167	167
Estimated 2D Tracks Person	158	158	158
Estimated 3D Tracks Person	40	102	129

As our system does not calculate confidence scores for the estimated 3D positions, we do not report ALP values but precision, recall and F_1 score (in substitution to ALP) values instead. In theory, one could use the confidence scores predicted by the 2D tracking network and forward it to the 3D detections in order to calculate ALP values. We decide not to do this because the 2D confidence scores are not directly related to the accuracy of the 3D detections. A very precise 2D detection can still lead to a weak 3D position estimate if there are features inside the 2D bounding box that do not belong to the object in question but to background. On the other hand a rather imprecise 2D detection might still lead to a precise 3D detection if the extracted features mostly belong to the object in question. The F_1 score is a suitable substitution for the ALP metric because similar to the ALP metric it penalizes low precision and low recall and favors high precision and high recall values at the same time.

The F_1 score is defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2 * precision * recall}{precision + recall} \tag{5.4}$$

The F_1 score may assume values between 0 and 1 where 1 represents a perfect system with precision and recall equal to 1.

Figure 5.17 shows precision, recall and F_1 score for different values of N_{feat} for the class car. Increasing N_{feat} leads to lower precision but higher recall. The increase in recall is more significant than the decrease in precision and thus leads to an increase in F_1 score as well. The behavior for the class pedestrian (refer to figure 5.18) shows the same tendency. However, for pedestrians the increase in the number of features shows an even bigger positive impact on the recall than for cars. The reason is that bounding boxes of pedestrians are smaller in average and thus, there are much more cases where the number of features inside a pedestrian bounding box gets too small for 3D tracking when setting N_{feat} to a small value.

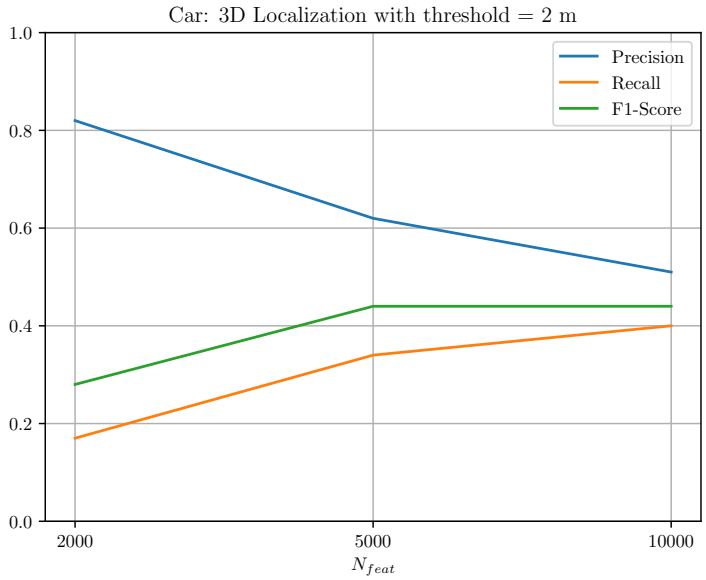


Figure 5.17: Car: 3D Localization performance with threshold = 2 m.

In order to analyze how big the typical localization errors for cars and pedestrians are, we show the prediction, recall and F_1 score values for localization error thresholds between 0 m and 10 m in figure 5.19 and figure 5.20. We set $N_{feat} = 10000$ for these experiments. We can see that most 3D detections of cars show a localization error between 0.8 m and 3.0 m. Increasing the tolerated localization error from 0.8 m to 3.0 m leads to a huge improvement in the F_1 score from around 0.04 to around 0.5. Further increasing the tolerated localization error from 3.0 m to 10.0 m leads only to an improvement from 0.5 to around 0.61 in F_1 score. We conclude that the typical localization error for cars is below 3 m. For pedestrians (see figure 5.20) most 3D detections show a localization error between 0.4 m and 2.0 m. Increasing the tolerated localization error from 0.4 m to 2.0 m leads to a huge improvement in the F_1

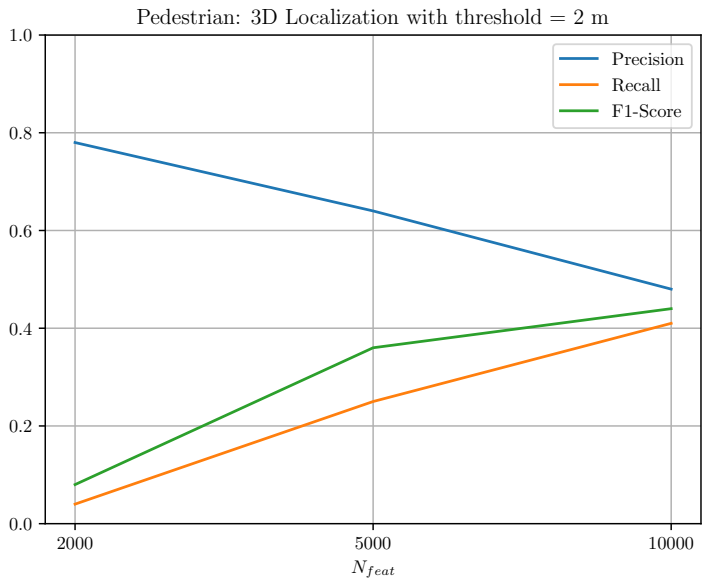


Figure 5.18: Pedestrian: 3D Localization performance with threshold = 2 m.

score from around 0.01 to around 0.44. Further increasing the tolerated localization error from 2.0 m to 10.0 m leads only to an improvement from 0.44 to around 0.49 in F_1 score. We conclude that the typical localization error for pedestrians is below 2 m.

5.3.4 Comparison of 3D Track Lengths

Figure 5.21 shows the number of tracks for different track lengths (m) and compares these numbers for estimated tracks with $N_{feat} = 2000$, $N_{feat} = 5000$, $N_{feat} = 10000$ and ground truth tracks for the class car. Estimated tracks are only considered in case their 2D bounding boxes have an IoU value with a ground truth 2D bounding box bigger than 0.5. Figure 5.22 shows the same statistics for the class person.

It is obvious that increasing the number of features is helpful for 3D tracking as it enables the system to create more tracks. Apparently there are much more short tracks estimated than there are present in the ground truth data. One reason is that longer ground truth tracks get split into more than one estimated track, e.g. due to ID-switches in the 2D bounding box tracking or due to track losses during the 3D tracking. 3D track losses occur due to a low amount of features inside the 2D object bounding box in one frame which leads to a track loss in that frame.

An additional root cause is that the system does not track all ground truth objects for the whole time, e.g. due to later initialization of further objects. One reason for such a late initialization is that further objects are projected to small bounding boxes with a small number of features that might not be sufficiently high for ORB-SLAM2 [27] stereo initialization. Table 5.9 displays the mean difference between the first frame of the ground truth tracks and the first frame of the estimated 3D tracks in frames and in meters to verify this hypothesis. Estimated

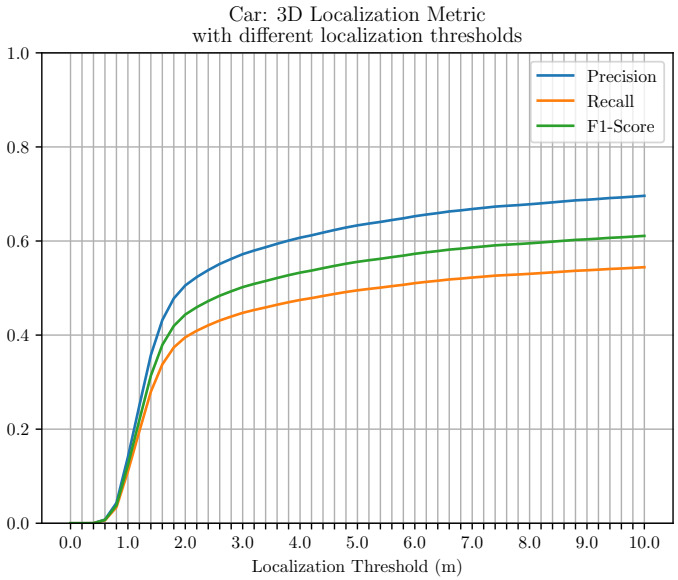


Figure 5.19: Car: 3D Localization performance with different localization thresholds.

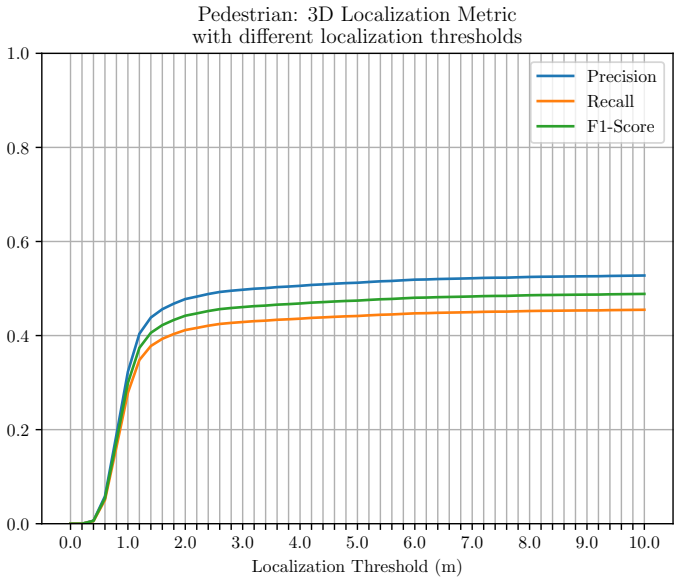


Figure 5.20: Pedestrian: 3D Localization performance with different localization thresholds.

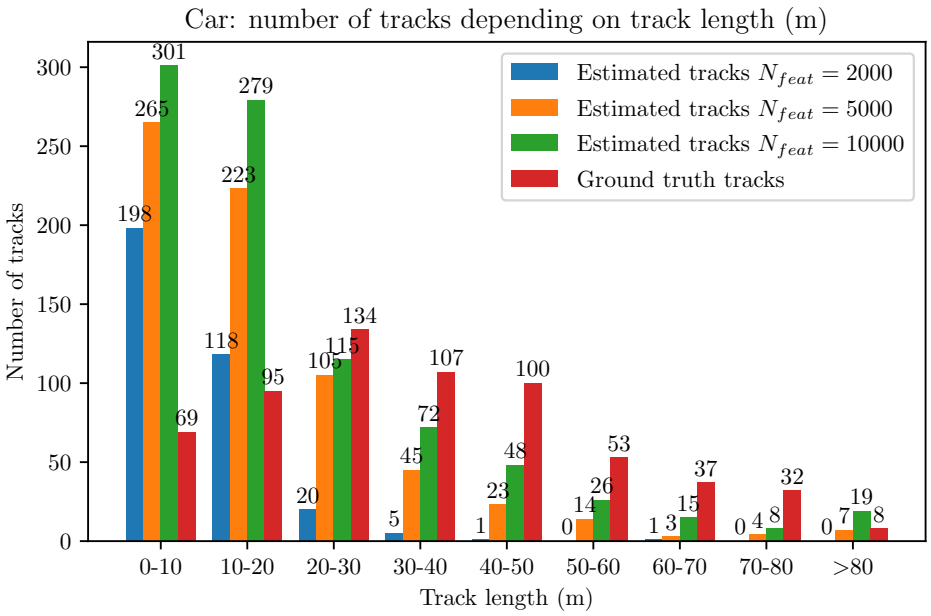


Figure 5.21: Car: comparison of number of tracks for different track lengths (m).

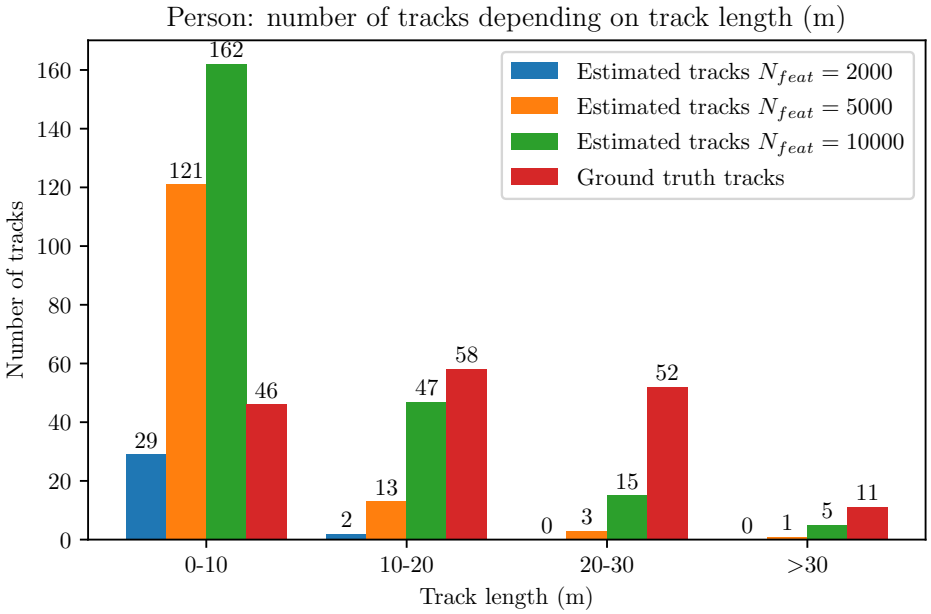


Figure 5.22: Person: comparison of number of tracks for different track lengths (m).

tracks that start before the corresponding ground truth track get assigned a difference of 0 frames and 0 m. Table 5.9 shows the mean difference of all ground truth tracks that are present in the estimated tracks for the different N_{feat} parameter settings. Increasing the number of features leads to much earlier starting points of the 3D tracks. Comparing $N_{feat} = 10000$ with $N_{feat} = 2000$ shows that an improvement of more than 12 meters regarding the starting point of the estimated tracks is achievable by using more features. This explains also the large increase in the number of tracks > 20 m for $N_{feat} = 10000$ compared to $N_{feat} = 2000$. Table 5.10 shows the mean of the square roots of all bounding box sizes at the beginning of the 3D tracks. The table supports the hypothesis that using more features enables the system to track objects in 3D based on smaller 2D bounding boxes. Thus, the system is able to track the objects earlier and the track length increases.

Table 5.9: Mean difference between first estimated frame and first ground truth frame depending on parameter N_{feat}.

	Person (frames)	Car (frames)	Person (m)	Car (m)
$N_{feat} = 2000$	62.12	28.36	9.71	19.76
$N_{feat} = 5000$	14.63	10.40	5.32	11.02
$N_{feat} = 10000$	0.80	4.34	3.41	7.27

Table 5.10: Mean of square root of bounding box sizes in first estimated frame depending on parameter N_{feat}.

	Person $\sqrt{wh}$ (pixels)	Car $\sqrt{wh}$ (pixels)
$N_{feat} = 2000$	109.31	101.13
$N_{feat} = 5000$	63.58	65.01
$N_{feat} = 10000$	48.17	52.62
Ground Truth	47.27	47.16

Note that there are more estimated tracks with a track length > 80 m for $N_{feat} = 10000$ than in the ground truth data. This is possible because the ground truth tracks might start later in case the initial bounding box height is smaller than 25 pixels [4]. We don't impose such a restriction on our estimated tracks and thus they might start earlier than the ground truth tracks in some cases which leads to potential longer track lengths.

The impact of 2D tracking ID switches on the 3D track length is analyzed in the following by suppressing these ID-switches artificially. This is done as follows: All tracks that belong to the same ground truth track but different estimated objects get connected if the first object track ends exactly one frame before the second object track starts. The impact for the class person is neglectable, figure 5.23 shows the impact for the class car. The 2D ID-Switches have a small but noticeable impact on the 3D track length histogram. Some of the very short tracks get removed and help to increase the amount of longer tracks (e.g. > 80 m).

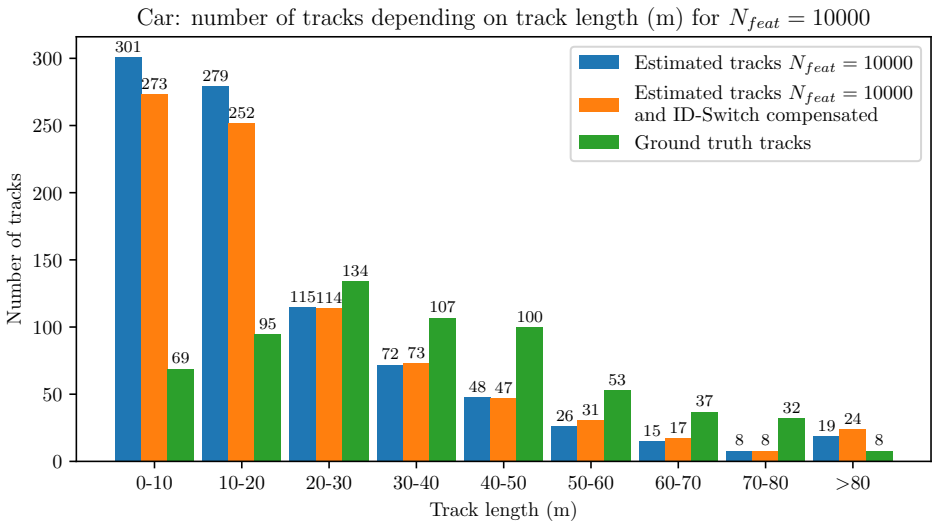


Figure 5.23: Car: comparison of number of tracks for different track lengths (m) depending on ID-Switch compensation.

In order to eliminate the impact of faulty 2D tracks caused by the 2D tracker network, we execute an additional experiment where we replace the predicted 2D tracks by 2D ground truth bounding boxes based on the segmentation masks of the MOTS [46] challenge. We use these bounding boxes because they cover only the visible portion of the relevant objects which helps to improve the performance of our 3D tracking algorithm on occluded objects. This experiment shows the upper performance boundary for our 3D tracking algorithm when setting $N_{feat} = 10000$. We can see in figure 5.24 that the usage of 2D ground truth tracks eliminates the very high number of short tracks which are mainly caused by ID-switches in the 2D tracker domain. Additionally, the amount of long tracks is significantly improved when using the 2D ground truth tracks. For example, the number of tracks in the range between 50 and 60 meters is increased from 26 tracks to 42 tracks. Figure 5.25 shows the result of the same experiment for the class person. We can see, similar to the class car, that a lot of short trajectories caused by ID-switches in the 2D tracker network don't exist when using the ground truth 2D bounding boxes. On top of that, the amount of longer tracks between 20 and 30 meters is increased significantly when using the 2D ground truth bounding boxes (from 15 to 24).

We assume that further improvements are possible by replacing the 2D bounding boxes by 2D segmentation masks in order to reduce the number of outliers. In addition, improving the feature selection procedure could help to improve the overall performance. For example, one could extract a minimum amount of features per 2D segmentation mask instead of a fixed number of features distributed over the whole image plane.

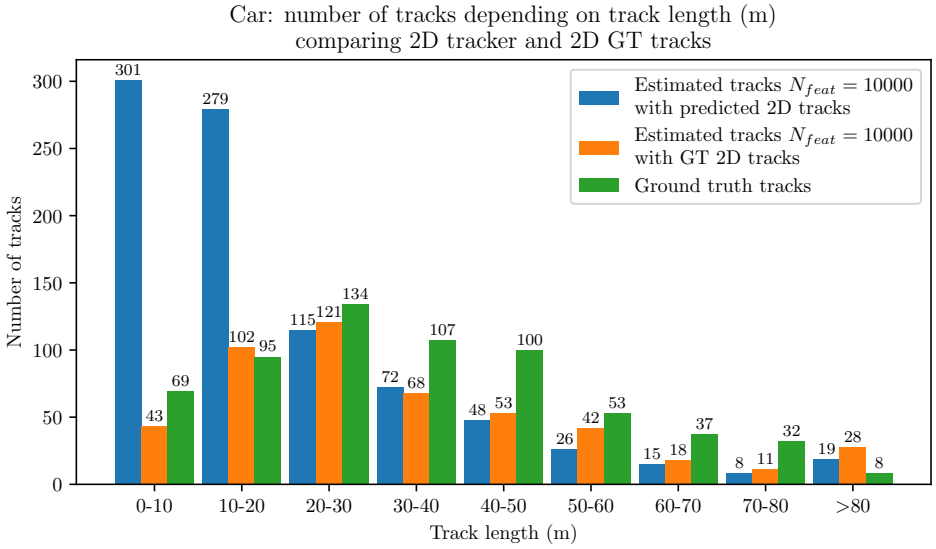


Figure 5.24: Car: comparison of number of tracks for different track lengths (m) when using the 2D tracker network or 2D ground truth tracks.

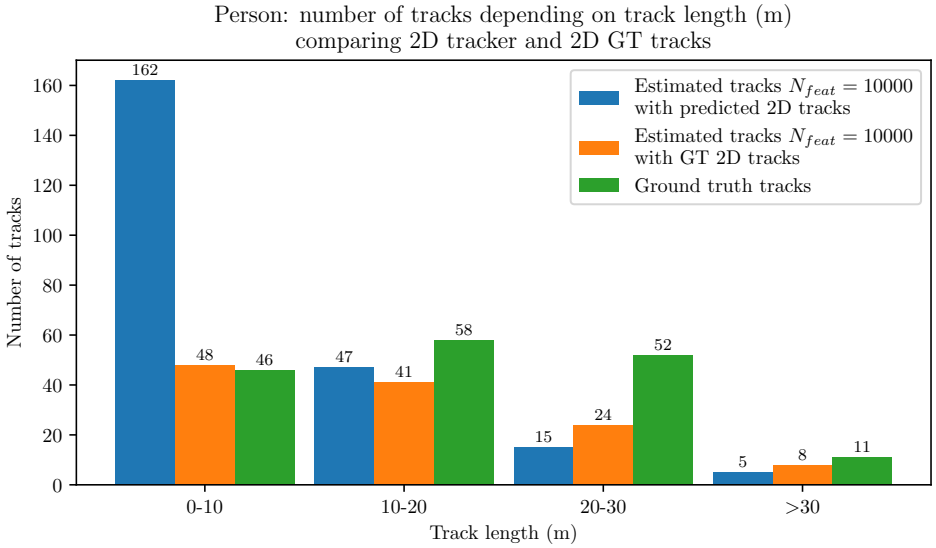


Figure 5.25: Person: comparison of number of tracks for different track lengths (m) when using the 2D tracker network or 2D ground truth tracks.

5.3.5 Evaluation on Object Trajectory Level

In this subsection 3D object tracking is evaluated based on the ATE metric which is a common metric in the SLAM community for assessing the quality of the camera trajectory estimation. This metric is not applied on the camera trajectory estimation as usual in SLAM but on the object trajectory estimation where the object trajectory is expressed in the camera coordinate frame. We evaluate the object trajectories and not the camera trajectory because the estimation of object trajectories in addition to the camera trajectory is the novelty of our algorithm compared to plain ORB-SLAM2 [27]. We express the object trajectory in camera coordinate frame because the object position is estimated relative to the camera (see equation 4.9). Furthermore, the ground truth object positions in the KITTI Tracking dataset [4] are also given in camera coordinates.

In [28] ATE is defined as follows: The poses of the estimated trajectory are given as $\mathbf{P}_1, \ldots, \mathbf{P}_n \in SE(3)$ and the poses of the ground truth trajectory are given as $\mathbf{Q}_1, \ldots, \mathbf{Q}_n \in SE(3)$. Then, the ATE of a time step i can be computed as:

$$\mathbf{F}_i := \mathbf{Q}_i^{-1} \mathbf{S} \mathbf{P}_i \tag{5.5}$$

where $\mathbf{S}$ is the similarity transform that aligns the predicted and ground truth trajectory by minimizing the least-squares error. It can be computed with the method of Horn [49]. This alignment is necessary because ground truth and predicted trajectory are not necessarily expressed in the same coordinate frame. In this thesis, the root mean squared error over all time indices, as suggested in [28], is used when referring to ATE values:

$$RMSE(\mathbf{F}_{1:n}) := \left(\frac{1}{n} \sum_{i=0}^{n} \| trans(\mathbf{F}_i) \|^2 \right)^{1/2} \tag{5.6}$$

In our case no trajectory alignment is required and we can set $\mathbf{S} = \mathbf{I}$, as ground truth object poses are given relative to the camera coordinate frame and we express the object positions also with respect to the camera frame.

Figure 5.26 shows the average ATE value for different track length bins for the class car. In the upper plot the figure shows the ATE values of all tracks of the respective setting of N_{feat}. It shows that increasing the number of features in general leads to higher ATE values for small track lengths. This is due to a lot of new short tracks that get estimated with $N_{feat} = 10000$ which do not get estimated with smaller values for N_{feat}. On the other hand, increasing the number of features can reduce the ATE values for longer tracks (> 30 m). The lower plot of figure 5.26 shows the same statistics but only considers those frames and tracks that are present for all three settings of N_{feat}. This means that the lower plot eliminates the impact of longer and additional tracks that can be only estimated when a high number of features is extracted (i.e. high value of N_{feat}). The lower plot shows that increasing the number of features is not systematically decreasing the accuracy of the object trajectories. Whereas $N_{feat} = 2000$ performs better for a track length between 30 and 40 m, $N_{feat} = 10000$ performs better for a track length between 40 and 50 m.

Taking into account that our 3D tracking approach is completely class agnostic and does not use any class specific 3D shape models and does not estimate 3D bounding boxes, an

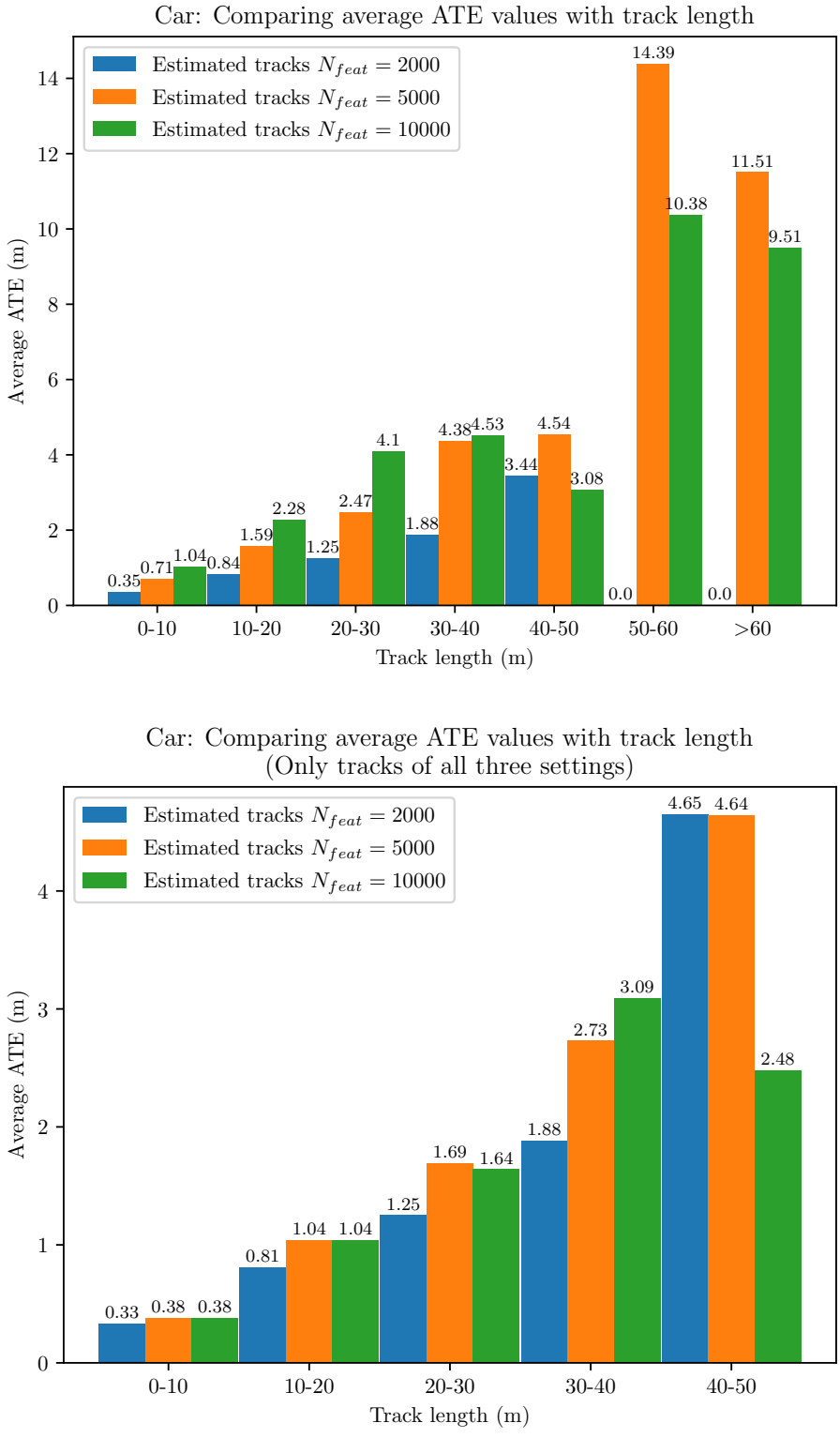


Figure 5.26: Car: Average ATE (m) for different track lengths.

average ATE value of 1 - 4.5 m for track lengths < 50 m is reasonable for cars as the typical length of a car is between 4 - 5 m.

Figure 5.27 shows the average ATE value for different track length bins for the class pedestrian. In the upper plot, the figure shows the ATE values of all tracks of the respective setting of N_{feat}. It shows that increasing the number of features leads to slightly higher ATE values. This is due to a lot of new short tracks that get estimated with $N_{feat} = 10000$ which do not get estimated with smaller values for N_{feat}. The lower plot of figure 5.27 shows the same statistics but only considers those frames and tracks that are present for all three settings of N_{feat}. This means that the lower plot eliminates the impact of longer and additional tracks that can be only estimated when a high number of features is extracted (i.e. high value of N_{feat}). The lower plot shows that increasing the number of features is not leading systematically to higher ATE values, as the ATE values increase between $N_{feat} = 2000$ and $N_{feat} = 5000$ but decrease between $N_{feat} = 5000$ and $N_{feat} = 10000$.

Taking into account that our 3D tracking approach is completely class agnostic and does not use any class specific 3D shape models and does not estimate 3D bounding boxes, an average ATE value of 0.5 - 1.0 m for track lengths < 30 m is reasonable for pedestrians as the typical height of a pedestrian is around 1.8 m and the typical width is around 0.5 m.

5.3.6 Comparison to DynaSLAM II

In the dynamic SLAM literature, different authors use different metrics to evaluate their approaches. There exists no common benchmark for the object tracking evaluation that is adapted by the dynamic SLAM community yet. Many of the used metrics rely on comparison of 3D bounding boxes for the 3D object tracking evaluation and are taken from the object tracking community (e.g. AP based on 3D bounding box IoU). As our approach does not estimate 3D bounding boxes but an object center point and respectively an object trajectory, these metrics are not applicable to our algorithm. DynaSLAM II [33] uses a metric that is based on 2D detections and 3D object trajectories and thus, we can apply it to our algorithm. Therefore, we compare our 3D object tracking performance to the DynaSLAM II [33] algorithm using the metric proposed by the authors of [33] in this subsection.

The authors identified the "12 longest sequences of the KITTI tracking dataset whose 2D detections are neither occluded nor truncated, and whose height is at least 40 pixels" [33, p. 7]. All 12 objects are cars. As our method does not predict 3D bounding boxes, we compare to [33] only based on the ATE metric and the 2D bounding box tracking performance. We compare the percentage of 2D true positives. We consider a bounding box as true positive if its IoU with the corresponding ground truth bounding box is bigger than or equal to 25%. We also compare based on the Multiple Object Tracking Precision (MOTP) which we define as the average 2D bounding box IoU of all 2D true positives. Table 5.11 shows the results. Our approach produces a higher 2D MOTP value for all objects as we use a dedicated 2D tracker network and don't project a 3D bounding box to the image plane like the authors of DynaSLAM II [33]. We reach a lower ATE error (object trajectory error) for 6 of the 12 objects. For 3 of these objects we also reach a higher 2D true positive rate which indicates that our object trajectory is more accurate and at the same time longer than the corresponding object

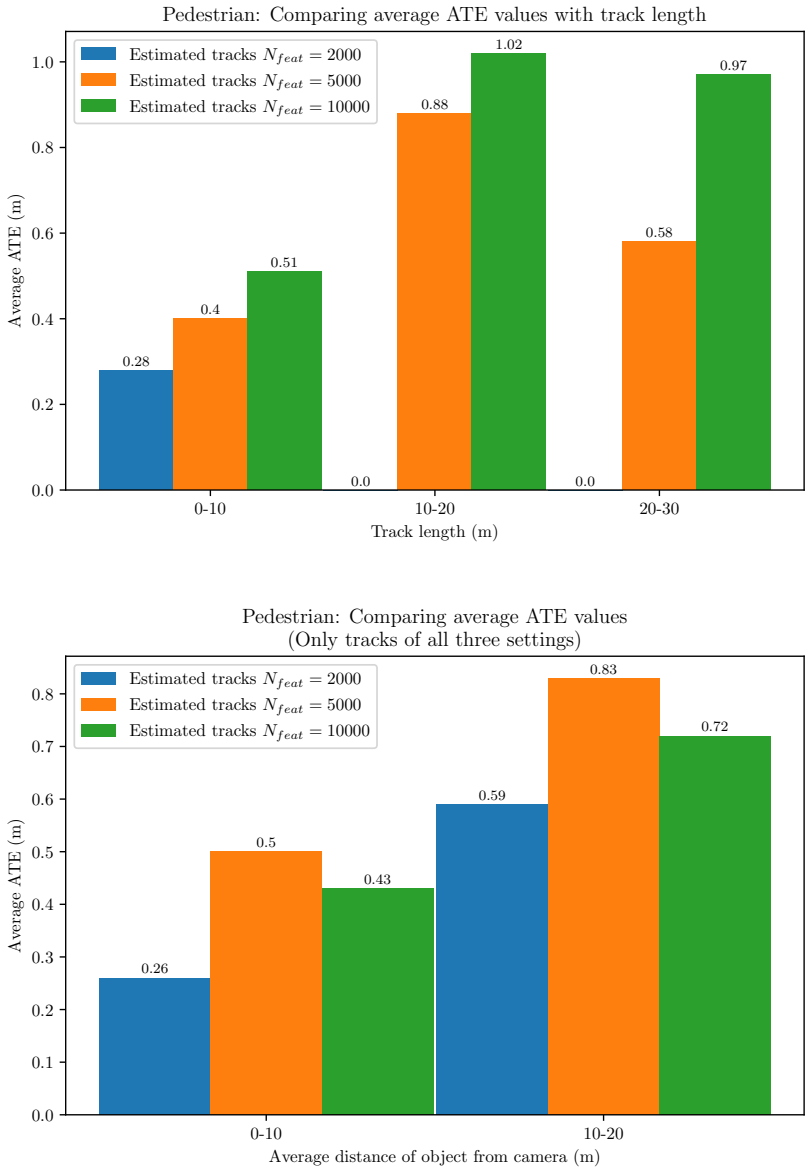


Figure 5.27: Pedestrian: Average ATE (m) for different track lengths.

trajectory of DynaSLAM II [33]. We get a higher 2D true positive rate for 6 of the 12 objects. It is notable that all three of our estimated object trajectories for sequence 0020 exhibit a significantly higher ATE error than DynaSLAM II [33]. This sequence contains lots of cars that are visible at the same time. Potentially, the motion prior of DynaSLAM II [33] is beneficial for the keypoint matching in this sequence and superior over our class-agnostic brute force matching of keypoints. Besides sequence 0020 our approach shows a similar object tracking performance as DynaSLAM II [33].

Table 5.11: Comparison of 3D object tracking performance with DynaSLAM II [33].

Sequence	0003	0005	0010	0011	0011	0018	0018	0019	0019	0020	0020	0020
Object ID GT	1	31	0	0	35	2	3	63	72	0	12	122
DynaSLAM II ATE	**0.69**	**0.51**	0.95	1.05	1.25	0.86	**0.99**	0.86	0.99	**0.56**	**1.18**	**0.87**
Ours ATE	0.72	0.78	**0.53**	**0.38**	**0.66**	**0.16**	1.05	**0.28**	**0.17**	3.98	2.95	6.10
DynaSLAM II 2D TP (%)	50.00	28.96	**81.63**	72.65	53.17	**86.36**	**53.33**	35.26	**29.11**	63.68	**42.77**	**34.90**
DynaSLAM II 2D MOTP (%)	71.79	60.30	73.51	74.78	65.25	74.81	70.94	63.50	62.59	78.54	76.77	78.76
Ours 2D TP (%)	**55.74**	**78.79**	31.29	**98.39**	**64.75**	40.15	2.11	**52.02**	9.18	**88.06**	13.53	5.22
Ours 2D MOTP (%)	**96.15**	**96.23**	**97.22**	**95.91**	**81.81**	**94.22**	**94.28**	**95.91**	**92.56**	**98.07**	**90.42**	**95.02**

6 Conclusion

In this thesis, we have developed a system that is able to estimate the camera trajectory and the trajectories of surrounding objects of a big variety of classes simultaneously. We have used a 2D tracker network trained on the LVIS dataset [17] and a modified ORB-SLAM2 [27] algorithm to estimate the 3D object positions of the objects over time.

In the qualitative evaluation, we have shown that our approach is able to track 22 different classes in the KITTI tracking dataset [4]. By visual inspection in bird's-eye view plots, we have seen that most of these tracks coincide well with the real world trajectories of the objects. Additionally, we have found that our approach performs better in tracking big objects than in tracking small objects.

In the quantitative evaluation, we have shown that a sufficiently high number of features is crucial to enable successful 3D tracking. The comparison to DynaSLAM II [33] has proven that our approach is able to achieve competitive trajectory localization accuracy of the 3D tracks in many different scenes of the KITTI dataset [4]. However, this comparison also has shown limitations in the localization accuracy of our approach for crowded scenes (sequence 0020 of the tracking dataset).

We propose several directions for future work based on these findings.

6.1 Future Work

In order to overcome some of the limitations of our approach, we propose the following directions for future work:

- As we have trained the 2D tracker network only on artificially shifted and scaled LVIS [17] frames, it has sometimes difficulties to correctly track the movements of the objects in KITTI [4] which leads to ID-switches. Therefore, we propose to fine tune the 2D tracker network on a small subset of the final application dataset (dataset for dynamic SLAM). We assume that fine tuning of the tracking head on KITTI [4] data could lead to a more stable 2D object tracking and thus to less track losses in 3D.
- Feature points that belong to the background, but are associated to objects because they lie inside the 2D bounding box of the object, can have a negative impact on the solution of BA. Thus, we propose to replace the 2D bounding box tracker network by a semantic segmentation network in order to reduce the number of background features that are wrongly assigned to objects. We assume that this could improve the localization accuracy of the tracking.

- Our approach has a lower recall for small objects than for big objects. To overcome this, we suggest to extract features not on a per frame basis but on a per object basis and enforce a minimum and maximum number of features per bounding box. We hypothesize that this might improve the tracking accuracy of small objects and also improve the run time by avoiding to extract many not required features on big objects and on the background.
- The crowded KITTI [4] sequence 0020 in which we exhibit a weaker performance than DynaSLAM II [33] has shown that the brute force keypoint matching of our approach is potentially susceptible to failure in such scenarios. Therefore, we hypothesize that an improvement of the association of stereo keypoints, e.g. by either training a neural network on the task or by simply using optical flow for data association might be beneficial.

In order to enhance the value of our approach for the community, we propose the following directions for future work:

- Integration of a 3D bounding box estimation module, e.g. by enclosing the top 90% of the point cloud points ranked based on their distance from the point cloud median. Not using the whole point cloud for the bounding box enclosure should help to suppress outlier points. The estimated 3D bounding boxes would be beneficial in order to apply standard 3D tracking metrics and enable a much easier comparison of our algorithm with other works originating from the tracking community.
- Enhancement of the system with meaningful confidence scores for the 3D detections. A useful starting point might be to assign confidence scores based on the reprojection error of the 3D keypoints of an object. Confidence scores are required, in addition to the already mentioned 3D bounding boxes, to adopt the common AP metrics of the object tracking community. On top of that, they are useful for downstream applications to help them to decide on the reliability of a certain object track.
- Development and establishment of a standard benchmark for comparison of dynamic SLAM algorithms. The lack of such a benchmark complicates the comparison of different approaches at the moment.
- Application of our system to SLAM datasets from other domains than autonomous driving, e.g. to an indoor dataset. As our system does not include any autonomous driving specific assumptions, it works on any type of SLAM dataset to which feature based approaches like ORB-SLAM2 [27] have been applied successfully.

In conclusion, we have developed a prototypical system that is able to estimate the trajectory of the camera and of surrounding objects in 3D. We have identified important limitations and potential directions for further research based on our approach. Our system can be seen as a proof-of-concept for a dynamic SLAM system that deals with a huge amount of different object classes. We hope that this thesis is a valuable starting point for future research in the area.

List of Figures

List of Tables

Acronyms

ALP Average Localization Precision. 52, 53

AP Average Precision. 8, 13, 14, 52, 63, 67

ATE Absolute Trajectory Error. 8, 61, 63, 65

BA Bundle Adjustment. 6, 8–11, 29–32, 36, 66

CNN Convolutional Neural Network. 3

DLA Deep Layer Aggregation. 4, 13, 14

EKF Extended Kalman filter. 6, 7

EM Expectation Maximization. 7

ICP Iterative Closest Point. 7, 9

IoU Intersection over Union. 8, 12–14, 17, 46, 52, 55, 63

LVIS Large Vocabulary Instance Segmentation. 5, 12–14, 24, 34–40, 66, 71

MOTA Multiple Object Tracking Accuracy. 15–23, 27, 70

MOTP Multiple Object Tracking Precision. 63

MOTS Multi-Object Tracking and Segmentation. 15, 59

NMS Non-Maximum Suppression. 3, 15, 17–25, 27, 28, 34, 70

RPE Relative Pose Error. 8

SLAM Simultaneous Localization and Mapping. 1, 2, 5–8, 10, 11, 14, 34, 61, 63, 66, 67

SVM Support Vector Machine. 3

VSO Visual Semantic Odometry. 7

Bibliography

[1] U. Frese, R. Wagner, and T. Röfer. "A SLAM Overview from a User's Perspective." In: *KI* 24.3 (2010), pp. 191–198.

[2] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad. "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics". In: *Intelligent Industrial Systems* 1 (2015), pp. 289–311.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[4] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *CVPR*. 2012.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *CVPR*. 2014.

[6] H. Law and J. Deng. "CornerNet: Detecting Objects as Paired Keypoints". In: *ECCV*. 2018.

[7] X. Zhou, J. Zhuo, and P. Krähenbühl. "Bottom-up Object Detection by Grouping Extreme and Center Points". In: *CVPR*. 2019.

[8] X. Zhou, D. Wang, and P. Krähenbühl. "Objects as Points". In: *arXiv:1904.07850* (2019).

[9] A. Newell, K. Yang, and J. Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: *ECCV*. 2016.

[10] F. Yu, D. Wang, and T. Darrell. "Deep Layer Aggregation". In: *CVPR*. 2018.

[11] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *CVPR*. 2016.

[12] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. "Focal Loss for Dense Object Detection". In: *ICCV*. 2017.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. "Microsoft COCO: Common Objects in Context". In: *ECCV*. 2014.

[14] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. "Simple Online and Realtime Tracking". In: *ICIP*. 2016.

[15] X. Zhou, V. Koltun, and P. Krähenbühl. "Tracking Objects as Points". In: *ECCV*. 2020.

[16] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. "MOT16: A Benchmark for Multi-Object Tracking". In: *arXiv:1603.00831 [cs]* (2016).

[17] A. Gupta, P. Dollár, and R. Girshick. "LVIS: A Dataset for Large Vocabulary Instance Segmentation". In: *CVPR*. 2019.

[18] T. Taketomi, H. Uchiyama, and S. Ikeda. "Visual SLAM algorithms: a survey from 2010 to 2016". In: *IPSJ Transactions on Computer Vision and Applications* 9 (2017).

[19] K. Tateno, F. Tombari, I. Laina, and N. Navab. "CNN-SLAM: Real-Time Dense Monocular SLAM with Learned Depth Prediction". In: *CVPR*. 2017.

[20] J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardós, and J. M. M. Montiel. "Towards Semantic SLAM using a Monocular Camera". In: *IROS*. 2011.

[21] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison. "SLAM++: Simultaneous Localisation and Mapping at the Level of Objects". In: *CVPR*. 2013.

[22] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. "Probabilistic Data Association for Semantic SLAM". In: *ICRA*. 2017.

[23] L. Nicholson, M. Milford, and N. Sünderhauf. "QuadricSLAM: Dual Quadrics From Object Detections as Landmarks in Object-Oriented SLAM". In: *IEEE Robotics and Automation Letters* 4.1 (2019), pp. 1–8.

[24] N. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler. "VSO: Visual Semantic Odometry". In: *ECCV*. 2018.

[25] C. Yu, Z. Liu, X. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei. "DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments". In: *IROS*. 2018.

[26] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song. "Semantic SLAM Based on Object Detection and Improved Octomap". In: *IEEE Access* 6 (2018), pp. 75545–75559.

[27] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *IROS*. 2012.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *CVPR*. 2016.

[30] P. Li, T. Qin, and S. Shen. "Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving". In: *ECCV*. 2018.

[31] J. Huang, S. Yang, Z. Zhao, Y. Lai, and S. Hu. "ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation". In: *ICCV*. 2019.

[32] M. Runz, M. Buffier, and L. Agapito. "MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects". In: *ISMAR*. 2018.

[33] B. Bescos, C. Campos, J. D. Tardós, and J. Neira. *DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM*. 2020. arXiv: `2010.07820 [cs.RO]`.

[34] M. Shan, Q. Feng, and N. Atanasov. *OrcVIO: Object residual constrained Visual-Inertial Odometry*. 2020. arXiv: `2007.15107 [cs.RO]`.

[35] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu. "ClusterVO: Clustering Moving Instances and Estimating Visual Odometry for Self and Surroundings". In: *CVPR*. 2020.

[36] K. He, G. Gkioxari, P. Dollár, and R. Girshick. "Mask R-CNN". In: *ICCV*. 2017.

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An efficient alternative to SIFT or SURF". In: *ICCV*. 2011.

[38] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger". In: *CVPR*. 2017.

[39] K. M. Judd and J. D. Gammell. "The Oxford Multimotion Dataset: Multiple SE(3) motions with ground truth". In: *IEEE Robotics and Automation Letters (RA-L)* 4.2 (2019), pp. 800–807.

[40] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. "Bundle Adjustment - A Modern Synthesis". In: *ICCV'99*. 1999.

[41] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[42] D. Gálvez-López and J. D. Tardós. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[43] A. Dave, P. Tokmakov, C. Schmid, and D. Ramanan. "Learning to Track Any Object". In: *ICCV Workshop on Holistic Video Understanding*. 2019.

[44] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. *Detectron2*. `https://github.com/facebookresearch/detectron2`. 2019.

[45] K. Bernardin, A. Elbs, and R. Stiefelhagen. "Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment". In: *Proceedings of IEEE International Workshop on Visual Surveillance* (2006).

[46] P. Voigtlaender, M. Krause, A. Ošep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe. "MOTS: Multi-Object Tracking and Segmentation". In: *CVPR*. 2019.

[47] A. Ošep, P. Voigtlaender, M. Weber, J. Luiten, and B. Leibe. "4D Generic Video Object Proposals". In: *ICRA*. 2020.

[48] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[49] B. K. P. Horn. "Closed-form solution of absolute orientation using unit quaternions". In: *J. Opt. Soc. Am. A* 4.4 (1987), pp. 629–642.