Technische Universität Berlin

Mutual Improvement of 2D Object Detection and 3D Tracking in Optical Navigation

Master Thesis
Field of Study Computer Science
Department of Computer Vision and Remote Sensing
Faculty IV Electrical Engineering and Computer Science
Technical University of Berlin

Submitted by
Maik Wischow

Submitted on
03.09.2019

1. Evaluator:	Prof. Dr.-Ing. Olaf Hellwich
2. Evaluator:	Prof. Dr.-Ing. Andreas Reigber
Supervisors:	MSc. Patrick Irmisch,
	Dr.-Ing. Anko Börner
	German Aerospace Center (DLR)
	Institute of Optical Sensor Systems
	Department of Real-time Data Processing

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Die selbstständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, den 03.09.2019.

.....
Unterschrift

Abstract

This thesis investigates a combination of visual 3D object trackers and 2D object detectors for mutual improvement in accuracy and runtime. In the light of an application in the field of optical navigation, requirements such as high reliability, real-time capability and general applicability are pursued for the implementation.
The combined application builds on the framework of the Integrated Positioning System – a multi-sensor system primarily used for self-localization and environment reconstruction. The developed object tracker approach works on point clouds, represents objects through Intrinsic Shape Signatures and Color Signature of Histograms of Orientations and locates objects using an advanced particle filter. As object detector, the Deep Learning based method YOLOv3 is employed. The semantics and the localization of the object detections are used to restrict the search space during object tracking. In return, the object tracking result is projected back onto the image, to additionally supplement object detections and enhance their precision.
The combined method is compared to each separate method using established metrics. The experiments are based on complementary datasets from the real world and 3D simulations. Within both datasets, parameter setups, object classes and environmental scenarios are varied according to different attributes.
The results show a faster and more accurate object tracking through the object detections. However, due to lower reliability, the proposed object tracker is not suitable for improving YOLOv3 detections.

Zusammenfassung

Diese Masterarbeit untersucht eine Kombination von visuellen 3D Objekttrackern und 2D Objektdetektoren in Hinsicht auf eine gegenseitige Verbesserung von Genauigkeit und Laufzeit. Vor dem Hintergrund einer Anwendung im Bereich der optischen Navigation, werden dabei Anforderungen wie hohe Zuverlässigkeit, Echtzeitfähigkeit und allgemeine Anwendbarkeit verfolgt.
Die kombinierte Anwendung baut auf das Framework des Integrated Positioning Systems auf – einem Multisensor-System, das unter anderem zur Verortung der Eigenbewegung und zur Rekonstruktion der Umwelt verwendet wird. Der entwickelte Objekttracker-Ansatz arbeitet auf Punktwolken, repräsentiert Objekte dabei durch Intrinsic Shape Signatures sowie Color Signature of Histograms of Orientations und lokalisiert Objekte mit Hilfe eines erweiterten Partikel Filters. Als Objektdetektor wird das auf Deep Learning basierende YOLOv3 eingesetzt. Die Semantik und die Position der Objektdetektionen dienen dem Objekttracker als Einschränkung des Suchraumes. Das Ergebnis des Objekttrackings in die Bildebene zurückprojiziert, ermöglicht darüber hinaus Objektdetektionen zu präzisieren und zu ergänzen.
Die kombinierte Methode wird anhand von etablierten Metriken mit der jeweils separaten Methode verglichen. Die Experimente basieren auf sich gegenseitig ergänzende Datensätze aus der realen Welt und 3D Simulationen. In beiden werden Parameter, Objekte und Umweltszenarien anhand von verschiedenen Attributen variiert.
Die Ergebnisse zeigen ein schnelleres und genaueres Objekttracking durch die Objektdetektionen. Auf Grund geringerer Zuverlässigkeit eignet sich der Objekttracker jedoch nicht zur Verbesserung von YOLOv3 Detektionen.

Contents

List of Figures

List of Tables

List of Abbreviations

(C)SHOT (Color) Signature of Histograms of Orientations

2D-OD 2D Object Detection

3D-OPT 3D Object Pose Tracking

6DoF Six Degrees of Freedom

AUC Area Under Curve

CNN Convolutional Neural Network

CODAPT Combined (2D) Object Detection And (3D) Object Tracking

DCNN Deep Convolutional Neural Network

DLR Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)

eSGM Memory-Efficient Semi-Global Matching

IMU Inertial Measurement Unit

IoU Intersection over Union

IPS Integrated Positioning System

ISS Intrinsic Shape Signatures

KLD Kullback-Leibler Distance

LIDAR Light Detection and Ranging

PCA Principal Component Analysis

PCL Point Cloud Library

PF Particle Filter

SGM Semi-Global Matching

STD Standard Deviation

SVM Support Vector Machine

VIO Visual Inertial Odometry

VO Visual Odometry

YOLO You Only Look Once

Introduction

This chapter motivates the work and states the objective, formulated as research questions. Following, a general overview to the structure of this work is provided.

1.1 Motivation

Many safety-critical indoor and outdoor applications – like autonomous driving, industrial robotics and the exploration of planets in the outer space – require a highly reliable navigation solution to interact with the 3D environment. To avoid dangerous situations, system requirements related to precision, flexibility, high generality and fast response times have to be fulfilled. Common global referencing approaches like the Global Positioning System fail in indoor areas and beyond earth, due to limited coverage and other physical limitations. Hence, they are too restricted for general environments and local referencing methods have to be employed. However, these methods have disadvantages on their own with accumulating drift leading the way. One solution to tackle this problem is to use object tracking in the 3D space, which can memorize the pose of tracked objects and correct the error at each visit (loop closure).
In recent years, camera sensors enjoyed increasing attention due to lower hardware cost, raising computation power and advances in camera technology. The variety of available approaches to capture the 3D environment, represent and track objects, predestines the camera as the sensor of choice in this thesis. Visual object trackers do not fulfill the stated requirements yet and lie far behind human level performance, when it comes to general applications. Problems are for instance tracking losses caused by unexpected motions, viewpoint variations and environment variations or tracking inaccuracies due to noisy 3D data of the environment and insufficient object representations. This thesis aims to improve visual 3D object tracking for general navigation applications.
Safety-critical systems are usually designed to utilize multiple sensors and techniques to analyze data. A multi-sensor system for general, local navigation applications is the Integrated Positioning System (IPS). Among others, the IPS framework employs a real-time capable 2D object detector, to obtain position and semantic information of objects in the environment. In recent years, 2D object detectors have become a relevant part of 2D tracking applications, the so called tracking-by-detection methods. The idea to reuse the 2D object detections to improve 3D object tracking too, seems natural. Hence, this thesis proposes a combination of both methods and evaluates, whether a 2D object detector can improve a 3D object tracking and vice versa, the 2D object detector might exploit the 3D object tracker, as well.

1.2 Objective

This thesis aims to create and evaluate a combined object pose tracking and object detection method, where the former works in the 3D space and the latter on 2D images. The combined method is denoted as CODAPT (Combined Object Detection And Pose Tracking) in this work. It is analyzed, how the combined approach performs in comparison to the respective isolated method. For this reason, the following research questions will be investigated:

(1) How can a 3D object pose tracker be combined with a 2D object detector?
A new method that combines both approaches is designed, implemented and discussed.

(2) Can a 2D object detector improve the accuracy and runtime performances of a 3D object tracker?
The tracking accuracy of a 3D object tracker with and without the support of a 2D object detector will be compared for different object classes to track in different tracking scenarios. Further, it is investigated with different parameter sets, whether 2D object detections can be employed to reduce the 3D tracking search space and therefore the runtime.

(3) Can a 3D object tracker improve the accuracy of a 2D object detector?
A 3D object tracker might detect objects more accurately than a 2D object detector, since it exploits depth information. The performances are compared analogously for different object classes to track in different tracking scenarios. As a consequence, it could provide more robust 2D object detections at runtime or refine the 2D object detector offline.

1.3 Scope

The scope of this work is limited to the following restrictions, to limit the field of study:

Navigation Scenario. Apart from the following scope, a navigation scenario application requires high generality, reliability and fast response times, due to potentially unknown and challenging environments. These requirements are considered as soft-constraints in this work, since it is uncertain yet that all can be achieved.

Hardware Equipment. The Integrated Positioning System with two equal cameras is used in this work to acquire arbitrary 3D environments including depth information.

Considered Objects. This work is motivated to support the estimation of a precise camera trajectory out of object trackings. To increase the precision and to simplify the evaluation procedure, only rigid and non-moving objects are considered in this work.

2D Object Detection. In the context of a navigation scenario, the chosen detection method should provide a good trade-off between runtime and accuracy as well as real-time capability. The chosen object detector should recognize many object classes in parallel, to enable multiple trackings of several objects.

3D Object Tracking. The requirements of the 2D object detection method also hold for the 3D object tracking method. As a further limitation, only one object tracking at a time is considered. Despite that, the chosen framework should be easily extensible to multiple object trackings for future work. A 3D object model and its starting position are assumed to be given to start the respective object tracking.

1.4 Organization of the Thesis

This thesis is structured as follows: Chapter 2 reviews state-of-the-art methods for object detection on images and object pose tracking in the 3D space, focusing preferably on the scope of this work. Following, works that combine both approaches are surveyed briefly and the approaches utilized in this work are outlined. Chapter 3 provides a theoretical background for the applied methods. First, the process of obtaining images and depth information are described. Moreover, the functionality of the chosen 2D object detector is summarized. Next, the outlined methods to represent objects in the 3D space and track them are elucidated – including 3D features, feature points, point descriptors and mechanisms to locate them. Finally, the IPS is introduced as a device to apply all the introduced functionalities.
Chapter 4 proposes and justifies the CODAPT algorithm in detail. The evaluation of CODAPT is then prepared in Chapter 5. It includes a general overview to evaluation execution and presents used real-world and simulated datasets, followed by applied metrics to quantify the performance. The Chapter 6 presents the evaluation results. Therefore, the results are first pre-analyzed and following evaluated with respect to the stated research questions. Next in Chapter 7, the results are discussed in the same structure and limitations of this work are stated.
Finally, this work is summarized in Chapter 8, where the contribution of this work is outlined, together with considerations for future work.

Related Work

2D object detection (2D-OD) and 3D object pose tracking (3D-OPT) are highly focused fields in Computer Vision and have been well researched in the recent decades. A comprehensive first entry point to these general topics provide the surveys (Zhang et al., 2013) and (Yilmaz et al., 2006). The works of (L. Liu et al., 2018) and (P. Li et al., 2018) review modern approaches.
In addition to the scope of this work, methods which do not counteract the stated limitations are considered as well. Since today's knowledge of 2D-OD and 3D-OPT is even under this scope widely scattered, brief overviews to the respective topics are given followed by a review of relevant state-of-the-art approaches.
At first, both topics are outlined separately. Subsequently, works combining both topics are reviewed. Last but not least, methods are chosen and reasoned with regard to the stated research questions and the scope for this work. Respective definitions of terms and concepts can be found in Chapter 3.

2.1 2D Object Detection

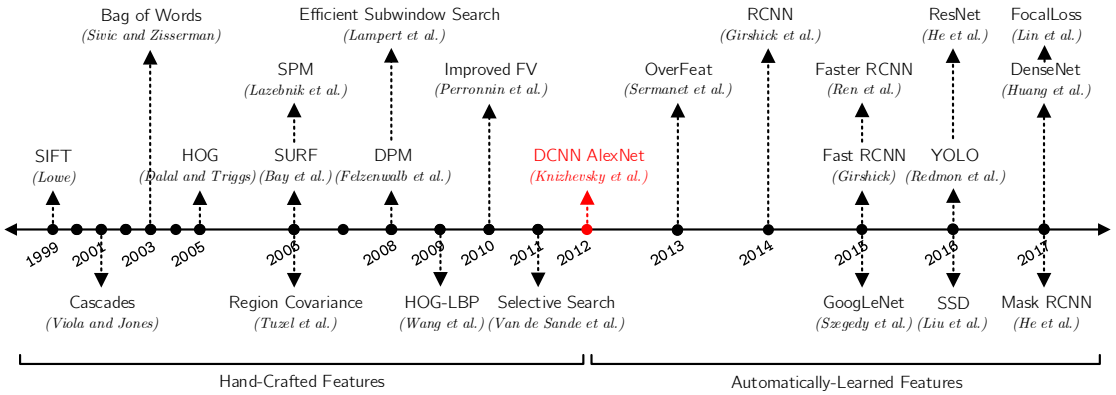


Figure 2.1: Milestones of 2D object detection and object feature representation from the past two decades. Most listed methods are highly cited and won one of the major ICCV or CVPR prizes, based on (L. Liu et al., 2018).

This section follows the timeline of Figure 2.1 from the past to the presence. First, the underlying tasks in 2D-OD are stated. Corresponding to these tasks, successful traditional hand-crafted methods are presented, followed by the introduction to automatically-learned feature methods. Then, state-of-the-art approaches are reviewed in comparison to each other.
The task of 2D-OD is dominated by two subtasks. The first one is finding a most discriminative feature representation for general objects which is also robust against variations such as object appearance or environmental changes. The second task is

to efficiently search these representations in a queried image. Typically, an image is transformed to the corresponding feature space, i.e. it is processed into a new image represented by the defined features, and occurrences with little deviation to the objects representation, regarding a certain metric are picked as matches. Concerning the first subtask, works from the past two decades like Scale-Invariant Feature Transform (SIFT) (Lowe, 1999), Speeded-Up Robust Features (Bay et al., 2006) or Features from Accelerated Segment Test (Rosten and Drummond, 2006) considered the detection and description of highly distinctive and transformation robust feature points, to represent objects. Representative works like Histogram of Oriented Gradients (Dalal and Triggs, 2005), Local Binary Pattern (LBP) (Ahonen et al., 2006) or Region Covariance (Tuzel et al., 2006) focused on describing feature information from the local area of and around the objects. In recent years, also advanced feature descriptor variants like RootSIFT Principal Component Analysis (RootSIFT-PCA) (Bursuc et al., 2015), Domain-Size Pooling SIFT (Dong and Soatto, 2015) or Rotation Invariant Co-occurrence among adjacent LBPs (X. Qi et al., 2014) were researched, to further enhance matching-performance and robustness. For the second subtask, Cascades (Viola and Jones, 2001), Efficient Subwindow Search (Lampert et al., 2008) and Selective Search (Sande et al., 2011) are key contributions to reduce the object search space by e.g. introducing a compressed image description or by finding a sub-search-space, which maximizes a given score function.

The year 2012 marks a big change in 2D-OD, because hand-crafted features are relegated from automatically-learned features. The term hand-crafted refers to features that require domain expertise and which are human-engineered beforehand. In 2012, the work of (Krizhevsky et al., 2012) proposed a CNN called AlexNet (see Figure 2.1 in red) which reached outstanding results for image classification in the Large Scale Visual Recognition Challenge (Russakovsky et al., 2015) and exceeding prior best results by a large margin. Since then, many research areas focused on deep learning methods. The subtasks of finding discriminative and robust object representations as well as efficient search methods have not changed – the novelty of deep learning is the automatic feature learning.

Most of the deep learning object detectors from the past years are based on one of two basic frameworks. The first framework consists of a two-stage pipeline: a pre-processing step for region-proposal followed by the object detection step. The seconds framework relies on the detection step only and on dense sampling of possible object locations. It is hence named one-stage detection, one-stage detection or region-proposal-free. While these one-shot frameworks have the potential to be faster, the two-stage ones perform more accurate in general (Lin, Goyal, et al., 2017).

The fundamental work for region-based framework is provided by Region-CNN (RCNN) (Girshick et al., 2014) – it combines the work of AlexNet and Selective Search, but suffers from slow and unnecessary complex training and testing procedures. Although its successor (Girshick, 2015) already sped up testing by one

order of magnitude, Faster RCNN (Ren et al., 2015) replaces the Selective Search region proposal by a more accurate and faster CNN-based method. In addition, Mask RCNN adds pixel-wise classification ability with only a small overhead (He et al., 2017). Also sharing CNN layers for similar tasks in the pipeline increases the training and testing time to make it a real-time capable method (L. Liu et al., 2018, p. 7). The Region-based Fully CNN from (Dai et al., 2016) aims to reduce computations, that can not be shared inside the pipeline and changed the region proposal procedure by adding position sensitive score maps. Although this approach reaches comparable accuracy scores, it often runs faster. Current top results on established benchmark datasets are mainly based on Faster RCNN (Deng et al., 2009; Geiger, Lenz, Stiller, et al., 2013; Lin, Maire, et al., 2014). Nevertheless, region-based methods are not appropriate for real-time applications on devices, which have limited memory and computational capability.

The research focus moved simultaneously to simplified region-proposal-free frameworks. The OverFeat CNN (Sermanet et al., 2013) is one of the first milestone works following this approach and saves computation time by sharing intermediate results of overlapping windows using a fully-connected CNN. The work of You Only Look Once (YOLO) (Redmon, Divvala, et al., 2016) transfers the classification problem to a regression problem of assigning image pixels to object bounding boxes and resulting class probabilities. Its two branches Fast YOLO (Shaifee et al., 2017) and YOLO9000 (Redmon and Farhadi, 2017) are extensions to a faster execution time and to a set of 9000 known object classes, respectively. The higher amount of classes is reached by a combination of multiple datasets using a wordtree for label abstraction and a joint training method. The subsequent works YOLOv2 (Redmon and Farhadi, 2017) and YOLOv3 (Redmon and Farhadi, 2018) led to state-of-the-art results in object detection benchmarks (Everingham et al., 2010; Lin, Maire, et al., 2014) by integrating outcomes from related works. The former adds for instance better anchor boxes or training on multiple scales, while the latter includes multi-label classification for non-mutually exclusive object classes. A different noteworthy approach is the Single Shot Detector (SSD) (W. Liu et al., 2016). SSD especially combines ideas from Faster RCNN and YOLO. In contrast to the version two of YOLO, it basically includes shallow layers in the CNN with higher resolution to support the detection of small objects. Based on SSD, the Deconvolutional SSD (Fu et al., 2017) added deconvolutional layers with the aim to involve contextual information. Thus, this SSD demonstrated an improved detection for small objects and for classes, which arise often together due to a common context.

Methods relying on automated feature learning clearly outperform methods with hand-crafted features in the stated benchmarks. Despite that, hand-crafted features are still relevant, mostly when the problem can be solved adequately with the available domain-knowledge or when there is not enough data available for a certain problem to train a deep neural network.

2.2 3D Object Pose Tracking

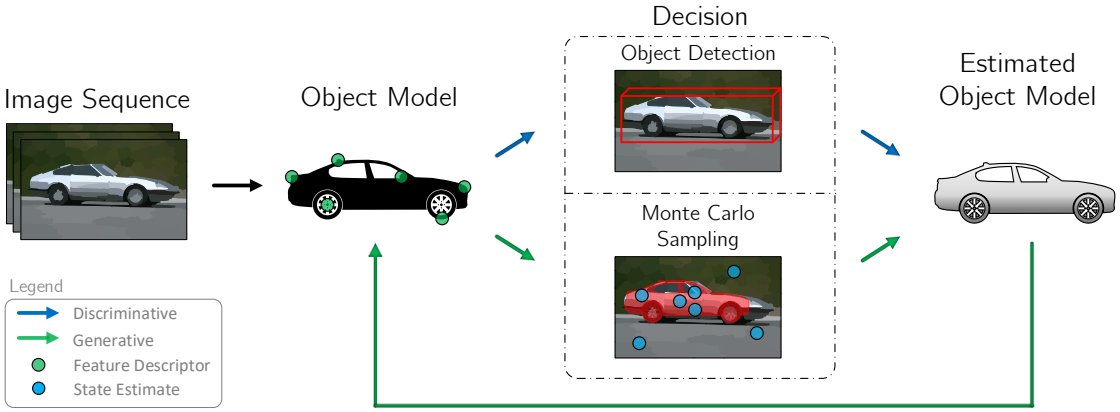


Figure 2.2: Example for an object pose tracking flow for generative and discriminative approaches, inspired by (H. Yang et al., 2011, p. 3824).

This section first motivates 3D-OPT and introduces different well-established 3D representation approaches. Afterwards, 3D-OPT methods are examined from different points of view in regard to their characteristics.
3D-OPT is motivated by its several applications that require navigation in 3D environments, like autonomous driving (Girao et al., 2016), robotics (Krainin et al., 2011) or Human Machine Interaction (Han et al., 2013). Due to low-priced sensors and higher computational power nowadays, navigation problems tend to be solved directly in the 3D space to tackle problems like scale-selection, occlusion, view-dependence or correspondence across views, known from 2D approaches (Tyagi et al., 2007). From researched 3D space representation techniques, for instance 2.5D elevation grids (Asvadi et al., 2015), voxel grids (Miyasaka et al., 2009; Broggi et al., 2013), octree-based data-structures (Azim and Aycard, 2014) and many more, the point cloud representation (Linsen, 2001; R. B. Rusu and Cousins, 2011) is the most established one. Despite the additional computation power of a 3D space, the respective algorithms have to allow fast response times to run on mobile autonomous agents, so that a fast online 3D-OPT is desirable.
Online 3D-OPT methods can be described by three characteristics: (i) tracking framework, (ii) object model representation and (iii) object search mechanism (H. Yang et al., 2011; Girao et al., 2016). Figure 2.2 illustrates an exemplary processing pipeline using these characteristics.

(i) Tracking Framework

Tracking frameworks of online 3D-OPT can be separated into discriminative (model-based) and generative (model-free) categories, based on whether object models are known beforehand or not.

In discriminative approaches, the object models are usually learned implicitly in an offline machine learning manner or given explicitly due to prior object appearance knowledge. Common established implicit learning methods are for example boosted learning (Zeisl et al., 2010), randomized learning (Breiman, 2001; Ozuysal et al., 2010; Godec et al., 2010), codebook learning (F. Yang et al., 2010) or CNNs (L. Liu et al., 2018). Having the object models, the method localizes the objects independently in each frame at runtime. Then, these single detections are matched to existing object tracks recursively by data association. Data association is also applicable for single object tracking, when same class objects occur close to each other – many works focuses on this task: One example is the Joint Probabilistic Data Association (Fortmann et al., 1983) method which assigns current measurements to available targets using a joint probabilistic score. Differently, Multiple Hypothesis Tracking (MHT) creates multiple hypotheses of all association hypotheses and chooses the maximum probability (Reid, 1979). Bayesian frameworks like the Particle Filter (Kitagawa, 1987; Arulampalam et al., 2002) (PF) perform data association inherently, by relating the measurement model to the object state estimate. Recent data association approaches deploy deep neural networks (Milan et al., 2017; Yoon et al., 2019) as well. The discriminative approaches recently gained attention with the success of modern 2D-OD methods – they are referred to as tracking-by-detection approaches. They provide the possibility to automatically reinitialize a lost tracking trail but a potentially large baseline between two detections lead generally to less tracking accuracy and more computationally load. Tracking approaches which do not belong to this class are labeled as recursive tracking methods – they make use of the strong temporal continuity constraint of the object pose between two frames, which reduces the tracking object search space but lead to tracking losses in challenging situations like occlusion (Lepetit, Fua, et al., 2005, pp. 66–72).
Generative approaches do not rely on pre-trained object models but construct these at runtime, to enable flexible applications (Ošep et al., 2016). Representative techniques rely on Mixture Models (Jepson et al., 2003), Kernel Density (Comaniciu et al., 2003) or Online Subspace Learning (Wang et al., 2012).
There are also hybrid methods that use the best of both, i.e. updating pre-build object models in an online manner, to account for appearance changes due to changing environmental conditions (illumination, occlusion, ...) (Shen et al., 2010; Lei et al., 2008). The discriminative and the generative frameworks are illustrated as different-colored arrows in Figure 2.2 and structure the underlying steps. The generative part are suppose to update the initial object model, while discriminative approaches mostly apply tracking-by-detection.

(ii) Object Model Representation
Object model representations are heavily researched for images, hence many established 2D approaches have been transformed and extended for 3D spaces – especially feature point detection in combination with local feature descriptors. Equally to

2D-OD of Section 2.1, object features are generally separated into hand-crafted and automatically-learned ones.
At first, 3D keypoint detectors for hand-crafted features are considered. They can be distinguished into fixed- or adaptive-scaled ones (Tombari et al., 2013). Well known fixed-scale detectors are for instance Intrinsic Shape Signatures (ISS) (Zhong, 2009), Keypoint Quality (KPQ) (Mian et al., 2010) or Heat Kernel Signature (HKS) (Sun et al., 2009). ISS basically measures saliency using the eigenvalues of the point scatter matrix and therefore tend to find well repeatable points quickly. KPQ extends the notion of ISS and additionally takes curvature properties into account, to find more keypoints with the drawback of also finding uniform areas. HKS describes the relation between two points as heat, which bases on the Laplace-Beltrami operator and takes local maxima as keypoints. The method provides more robustness against different invariances at the expense of a high memory complexity. Adaptive-scale detectors are among others KPQ Adaptive-Scale (KPQ-AS), MeshDoG (Unnikrishnan and Hebert, 2008) or 3D-SURF (Knopp et al., 2010). Additionally to KPQ, KPQ-AS also performs automatic scale selection for each keypoint using non-maxima suppression. MeshDoG utilizes Difference-of-Gaussian (DoG) derivations as a saliency measurement and can apply different kernels for the DoG operator. MeshDoG tends to find many keypoints around areas with high local curvature. 3D-SURF creates a scale space by a voxelizing model representation, to use box-filters for saliency computation and provide stable results by pruning keypoins in low-curvature-areas. 3D keypoints can then be robustly characterized by either Spatial Distribution Histogram (SDH) based or Geometric Attribute Histogram (GAH) based 3D keypoint descriptors (Guo, Bennamoun, et al., 2016; Hana et al., 2018). SDH based descriptors rely on histograms containing information about the spatial distribution of points on surfaces. The works of Rotational Projection Statistics (Guo, F. Sohel, et al., 2015) and Tri-Spin-Image (Guo, F. A. Sohel, et al., 2013) are supporter of the SDH group. The former follows the notion of gaining invariance against rigid transformations by encoding point distribution statistics for different model rotations. The latter provides transformation robustness by forming a spin image (Johnson and Hebert, 1999) for each coordinate axis and the descriptor out of these. In contrast to SDH descriptors, GAH based approaches build descriptors using geometric features of surface points (e.g. surface normals). The most common GAH method might be the Point Feature Histograms (PFH) (Radu Bogdan Rusu, Marton, et al., 2008; Radu Bogdan Rusu, Blodow, et al., 2008) and the run-time improving follow-up work Fast-PFH (F-PFH). F-PFH summarizes for a keypoint the relationship to its neighbor points for each feature dimension in a histogram, instead of considering the relation between all points in the region as before. Afterwards, the accumulation of the keypoints' histograms with other ones in a certain support region, form the descriptor. Another method called Signature of Histogram of Orientations (SHOT) (Tombari et al., 2010) encodes surface normal relations between the respective keypoint and points in a support region into histograms. Its amendment Color SHOT

(CSHOT) (Tombari et al., 2011) also takes texture information into account and therewith enhances accuracy with a small loss in runtime performance.
Recent works investigate automatically-learned keypoint detectors and descriptors, too. (Simo-Serra et al., 2015) proposes descriptor learning on natural image patches using a deep CNN (DCNN) with the aim that patches representing the same 3D point lead to similar descriptors, but with the disadvantage that these descriptors are not directly comparable to each other. In contrast, (Zeng et al., 2017) learns local 3D descriptors directly. Successive works like (H. Huang et al., 2018) added semantic similarity of points to their descriptors as well and make them directly comparable among themselves. They additionally learn from different views and multiple scales for more robustness.
In Figure 2.2, feature points and their descriptors are illustrated as green circles and used to represent an object model.

(iii) Object Search Mechanism
The approach to locate an object in the search space mainly depends on the chosen tracking framework. Tracking-by-detection techniques typically exploit the object localization mechanism of the applied 2D-OD (see Section 2.1). Other methods, including generative techniques, require an explicit object localization. Figure 2.2 depicts this associations as the "Decision" step. A well researched and general approach, that estimates probable object poses and reduces the search space, is the already introduced PF.
The PF framework seems as the natural choice for navigation in arbitrary environments. It does not linearize the search problem, supports multi-modal distributions and belongs to the non-parametric procedures. The PF reduces the object search region by randomized sampling. Further, compared to exhaustive search methods, the PF mainly considers promising search areas, takes temporal coherence into account and requires constant sampling effort, independent from the tracking object model. A disadvantage is the exponential grow of computation power with the number of search space dimensions, though. There are different approaches to tackle this problem: The Rao–Blackwellized PF (Doucet et al., 2000) for instance updates one part of the search space analytically and applies sampling to the rest. In (Fox, 2002), the proposed Kullback-Leibler Distance (KLD) PF bounds the approximation error while sampling using the KLD: if the particle density is focused on a small area, the amount of samples will be kept small. When the particles spread, the number will be increased. The particles themselves are non-dependent of each other, hence highly parallelizable (Choi and Christensen, 2013). Recent 3D-OPT works, that consider the PF framework, are for instance (Vatavu et al., 2015; Teuliere et al., 2015; Concha et al., 2018).

2.3 Combined Methods

The majority of combined works take place in traffic scenarios. The earlier work of (Leibe, Cornelis, et al., 2007; Leibe, Schindler, et al., 2008) performs aggregation of 2D-ODs and a space-time trajectory analysis, followed by hypothesis selection, similar to MHT. As the authors state, the approach can be extended to arbitrary 2D object detectors. One pitfall of their method is the overall higher runtime. The work of (Osep et al., 2017) directly tracks in point clouds for autonomous driving scenarios. They exploit the semantics of an cascaded boosting 2D-OD, together with the extended localization in world coordinates and couple them loosely in a Kalman-Filter for tracking. (Geiger, Lauer, et al., 2013) relates 2D detections from a modified Support Vector Machine (SVM) to 3D vehicle objects, with the goal to track them in the 3D space. Besides object appearance cues, they employ the intersection-over-union score of detection bounding boxes and object boundaries, to associate the data to each other. Markov Chain Monte Carlo sampling inherits their inference part. Zarzar et al. propose a double Siamese neural network for 3D-OPT in point clouds with the help of 2D-OD proposals of the point clouds' Bird Eye View image (Zarzar et al., 2019). Their approach leverages these images to find object tracking proposals in the related point cloud part with the help of an Region Proposal Network, which outperforms Kalman- and Particle Filter approaches in their traffic scenario tests. They also outline that even rough searches in the BEV image provide "outstanding improvements" for the 3D-OPT. The procedure of (Frossard and Urtasun, 2018) performs object detection using a DCNN with mini-batch images and point clouds as input. The detection proposals are then be utilized in two subsequent DCNNs, which provide matching and scoring results for 3D-OPT trajectories. Global trajectory optimization in a MHT manner eventually leads to a final matching result.
In contrast to previous mentioned works, (Pauwels and Kragic, 2015) apply a combined method in a visual servoing scenario. The 2D-OD procedure relies on SIFT keypoint extraction and an initial created 3D codebook for the object model. Their 3D-OPT takes advantage of a predefined 3D CAD object model, pre-computed surface normals and the optical flow using multiple cameras. The combined method then consists of a combined optimization problem, which determines the best object pose.

2.4 Outline

For 2D object detection, YOLOv3 yields the best tradeoff between accuracy and runtime in general object detection benchmarks. Still, this tradeoff is one disadvantage of YOLOv3 – Faster RCNN for instance would lead to more accurate object detections. YOLOv3 is real-time capable and like its predecessor YOLO9000 applicable to 9000 object classes, which makes it attractive for general navigation scenarios with unknown object classes. The code of all YOLO versions is open available, so

that further research and support from the research community can be presumed. Therefore, YOLOv3 is chosen as the 2D-ODs for this thesis. The preliminaries of using rigid and non-moving objects as well as a stereo camera setup have no influence on all the considered state-of-the-art 2D-ODs.
As the 3D scene representation, the point cloud format is chosen, since it is well-established for general real-time 3D applications. For the 3D-OPT technique of this work, a discriminative tracking framework forms the basis, due to the addressed beforehand given object model. This given model is assumed to be represented as a point cloud too, to simplify the object search approach. Although the discriminative DCNN methods lead in tests, when it comes to accuracy and runtime, a disadvantage is the availability of 3D training data, compared to the amount of training data available for general 2D object detectors. This restriction makes them unsuitable for a general object tracking application at the present time. Generative update mechanisms account for changes of the object appearance, so that a more robust object tracker can be expected, but this improvement is considered as future work here. The sparse 3D training datasets likewise encourages the use of hand-crafted 3D features for the object model representation. The benchmarks for 3D feature detections (Tombari et al., 2013) and 3D feature descriptions (Guo, Bennamoun, et al., 2016; Hana et al., 2018) approve the combination of ISS and CSHOT as a good decision, when it comes to descriptiveness, transformation robustness and computational efficiency for time-critical applications in potentially larger point clouds. All mentioned 3D components are part of the open source Point Cloud Library (PCL) framework (R. B. Rusu and Cousins, 2011), which provides comprehensive and constantly evolving functionalities.
Similar to the work of (Geiger, Lauer, et al., 2013), the 2D-OD in this work is utilized to yield a region proposal for the 3D-OPT. As a detected image object does not directly lead to the corresponding object in 3D, a KLD-PF is used to execute the remaining object search and thereby determine the final object pose. Further reasons to choose the PF are that it accounts for non-linear motion estimation of the camera itself and can support data association.
The commonly known and up-to-date multiple object tracking benchmark for 3D object tracking (Leal-Taixé et al., 2015) and also the mentioned 2D-OD benchmarks indicate the imperfect accuracies for the respective task as unsolved problems – especially when fast response times with at least two-digit frame rates are expected. Typically, accuracy is traded for computation time, like in YOLOv3 and the PF. This and the necessity for an accurate tracking in navigation scenarios, encourage the investigation of a potential benefit, when combining a 2D-OD and a 3D-OPT.

Fundamentals

This chapter introduces the theoretical and mathematical background used in this work. First of all, the visual mapping of 3D real world objects onto 2D images including depth features is reviewed as Image Formation. Second, the task of visual object detection in images is explained for the chosen method YOLO and its underlying CNN architecture. Third, the methods using the depth information are described, starting with the approaches of representing an object in the 3D space employing low and high level features. Subsequently, this section comprises the object pose tracking in the form of an adapted particle filtering and fundamentals to Visual Odometry, as the prior for the optical navigation task. Finally, the IPS is introduced as a device combining the previously outlined vision tasks.

3.1 Image Formation

Image Formation embraces the process of how 2D images of 3D objects are formed. It considers the including radiometric and geometric mechanisms as well as techniques for representing them in digital systems (Richard Szeliski, 2010, pp. 31–91). This section focuses on the geometric processes.
At first, the basic mathematical principles for the camera model used in this work are introduced, to image radiometric information of 3D objects. To retain depth information of the imaged objects as well, the geometric process of how to create images with depth information using a stereo camera setup are explained afterwards as Stereo Vision. Following, the algorithm SGM is reviewed, which post-processes the depth features to make them applicable.

3.1.1 Camera Model

The most frequently used camera model in Computer Vision applications is the pinhole camera model (illustrated in Figure 3.1) – the name refers to its presumed dot-sized aperture and the absence of a lens collimating the arriving light (Richard Szeliski, 2010, p. 49). Mathematically, it describes the projection of a 3D point $M \coloneqq [x, y, z]^T$ from the global Euclidean world coordinate system W on the corresponding 2D point $m \coloneqq [u, v]^T$ in the camera's image coordinate system I. By writing the two points in homogeneous coordinates $\tilde{M}$ and $\tilde{m}$, they can be related by the linear equation

$$\lambda \ \cdot \tilde{m} = P \cdot \tilde{M}, \tag{3.1}$$

where λ denotes a scale factor and P an affine projection matrix. The projection matrix P can be decomposed into the matrices K and E, which contain intrinsic and extrinsic camera parameters, respectively. The former depends only on the in-

ternal camera calibration, while the latter describes the rotational and translational transformation from W to the local camera coordinate system C.

$$P = K \cdot E = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \left[\; R_{3x3} \mid t_{3x1} \; \right], \text{ with } \alpha = \frac{f}{\delta}. \tag{3.2}$$

For simplicity's sake and due to common conventions, an unskewed camera image with quadratic pixels is assumed. In Equation 3.2, $c \coloneqq [u_0, v_0]^T$ denotes the principal point in the camera image, α the principal distance in pixel units, f the camera's focal length and δ the image's pixel size. In multi-camera setups, a 3D point can be transformed between the different local camera coordinate systems using a transformation matrix $E_{CC'}$, similar to the transformation E (see Figure 3.1). If the multiple cameras are rigidly mounted to each other, $E_{CC'}$ can be determined beforehand by stereo camera calibration (Hartley and Andrew Zisserman, 2003).
One drawback of the pinhole camera model is the long exposure time due to its small aperture, that restricts moving object and low light condition applications. The solution is the usage of lenses focusing the light and therefore allowing bigger apertures but naturally, imperfect lenses introduce mainly geometric distortions. This and other technical improvements of cameras used nowadays lead to more extensive camera models. Further details for commonly applied camera models can be found in (Foley et al., 1990).

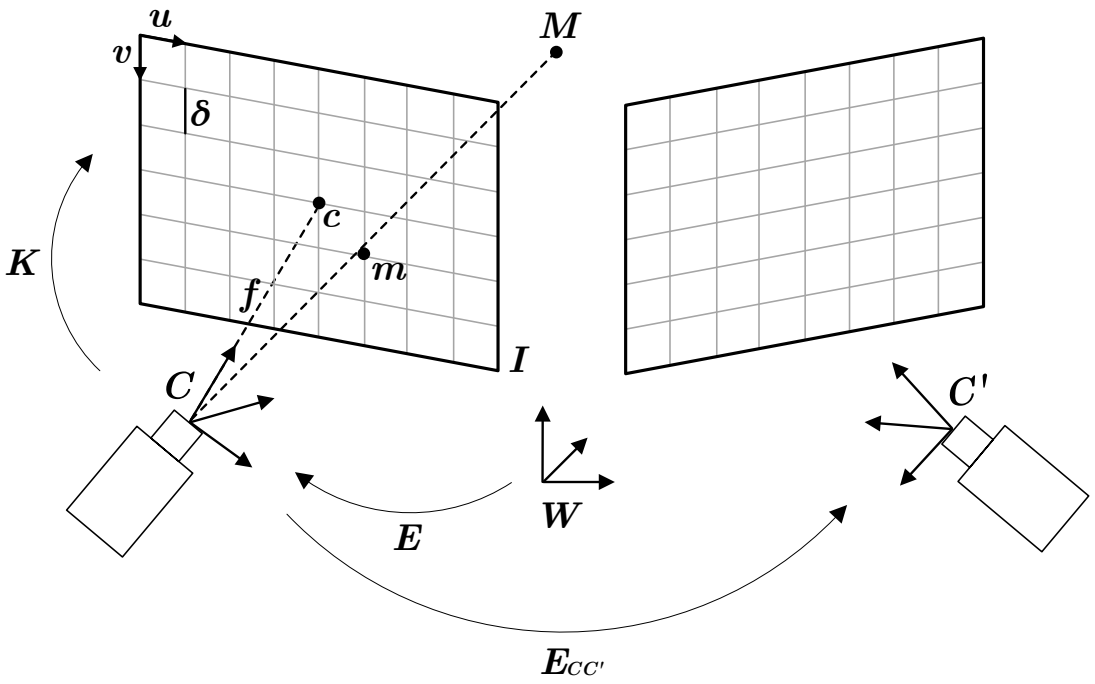


Figure 3.1: The pinhole camera model in a stereo camera setup, inspired by (Moulon et al., 2012).

3.1.2 Stereo Vision

Stereo Vision is mainly concerned with determining relative distances of objects to a stereo camera setup, whereby the binocular vision can be naturally expanded to general multi-camera systems. Taking a conjugate image pair of the same scene, one can induce depth information by means of disparity between corresponding image points m and m' of the same object point M (Hartley and Andrew Zisserman, 2003). A major problem Stereo Vision deals with is the correspondence problem of finding the correct m' to an arbitrary m. The search problem can be simplified by making use of epipolar geometry (Figure 3.2). The baseline B connects the camera coordinate systems C and C'. B's intersection with the image planes I and I' form the epipols e and e'. The plane, spanned by B, $\overline{MC}$ and $\overline{MC'}$, is called epipolar plane and its intersections with I and I' define the corresponding epipolar lines l and l'. The epipolar constraint then states that the point m' lies on the epipolar line l'.
A more efficient search is achieved by rectifying the images I and I'. Knowing the calibration between C and C' (i.e. the transformation $E_{CC'}$ in Figure 3.1), image rectification rotates the images I and I' to a common image plane, so that the epipolar lines l and l' lie on the same horizontal line and hence the search space for m' is reduced to the same vertical coordinate (Richard Szeliski, 2010, p. 530). As a result, the disparity d can be calculated as

$$d = |h(m) - h(m')|, \tag{3.3}$$

with $h(\cdot)$ as the horizontal coordinate of an image point. In a rectified image pair, the depth z to the point M is inverse proportional to d:

$$z = \frac{f \cdot B}{\delta \cdot d}, \tag{3.4}$$

where f is the focal length to the rectified images and δ the image's pixel size. So extracting depth can be reduced to disparity estimation. What remains is the need for a technique to find a m' to m, that also handles pitfalls like occlusion or ambiguous matches. Such an approach is referred to as Stereo Matching and a representative one is described in the next section.

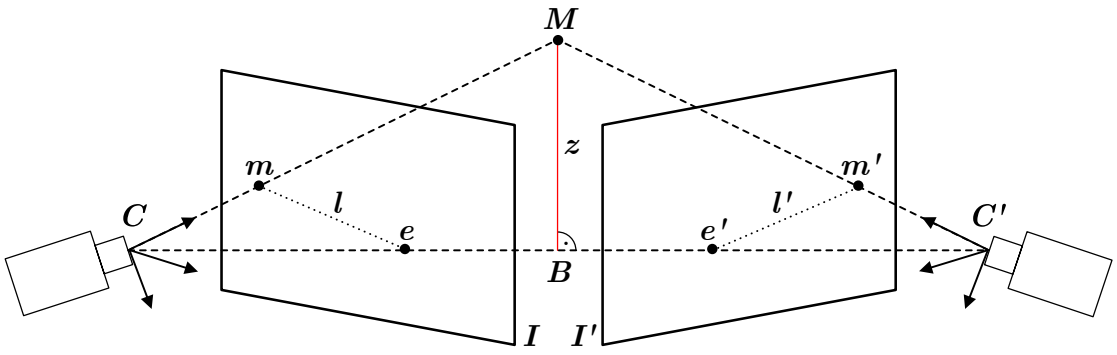
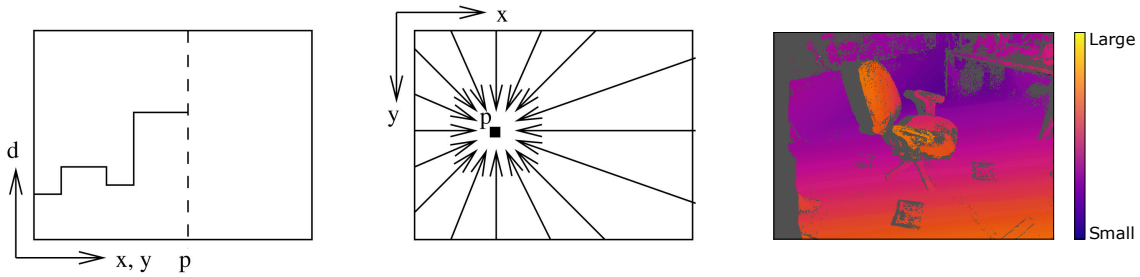


Figure 3.2: Stereo camera epipolar geometry, inspired by (Mariottini et al., 2004).

3.1.3 SGM

Semi-Global-Matching (SGM) forms the basis of current state-of-the-art stereo matching algorithms, when it comes to the best trade-off between accuracy and computational complexity. SGM "[...] uses a pixelwise, Mutual Information (MI)-based matching cost for compensating radiometric differences of input images" (Hirschmüller, 2008). The algorithm takes an image pair from a calibrated and rectified stereo camera setup as input and outputs a disparity image with applied "semi-global" smoothness constraints, which preserve sharp boundaries and a high grade of detail. These constraints are enforced by pathwise optimizations from different directions for each pixel of the disparity image (see Figure 3.3a). In more detail, the algorithm computes the matching costs between a pixel p and all possible disparity values for each pixel in the predefined directions. Next, it forms a matrix out of these values and uses dynamic programming to find the optimal path through the matrix with minimal cost regarding to the smoothness constraints (see Figure 3.3a). Subsequent post-processing steps moreover remove outliers and interpolate gaps.
SGM has two fundamental problems, though. Generally, it is unable to match pixels in homogeneous and low textured areas. Furthermore, it suffers from high memory complexity, which depends on the image size and the maximum allowed disparity value. The memory efficient GPU implementation eSGM overcomes the latter problem. Its memory complexity only depends on the image size, but to the cost of 50% more computing operations than SGM (Hirschmüller et al., 2012).
Figure 3.3b illustrates an exemplary colored disparity image of a chair. Points close to the camera relate to large disparities and far points to small ones. Grey colored pixels correspond to no disparity values, when no matching could be performed.
For further reading, a state-of-the-art Stereo Matching evaluation is provided in (Scharstein and R. Szeliski, 2019).



(a) Cost matrix with minimum cost path (left) and optimization along 16 paths in the disparity image (right).

(b) Example of a disparity map output from SGM.

Figure 3.3: Illustration of Semi-Global-Matching. Figure (a) by (Hirschmüller, 2008).

3.2 2D Object Detection

The commonly used abbreviation object detection indicates a task also known as object class detection (Zhang et al., 2013), object category detection (Aytar and A. Zisserman, 2011) or category-level object detection (Glasner et al., 2011), whereby the prefix 2D explicitly refers to images as the target domain. The term object addresses physical things that can be captured visually. The detection task is defined as the combination of two subtasks: object categorization and object localization. The former subtask aims to determine the existence of queried object category members. The latter identifies the position of the former task's result – depending on the algorithm, the output varies from a single point and an object contour mask. Object detection is a generalization task, which aims to find any object class members and is not to be confused with the related object recognition, which in contrast is limited to find a specific object (Zhang et al., 2013, p. 2).
The first section introduces CNNs as the generally most performant state-of-art architecture for object detection in images. Next, YOLO as the object detector used in this work is reviewed.

3.2.1 Convolutional Neural Network

Convolutional Neural Networks belong to the class of Artificial Neural Networks and are mostly applied to analyze images (Krizhevsky et al., 2012). CNNs were first successfully applied to object detection tasks around 2012 and head general object detection benchmarks ever since (see Section 2.1). The major advantages are its independence from prior hand-crafted knowledge and the well underpinned biological background (Fukushima, 1980; Hubel and Wiesel, 1968). Analogously to neural networks, CNNs consist of input, hidden and output layers. For image analysis tasks, the input layers are structured like the input images: each image channel corresponds to one input layer, each image pixel to one neuron and each pixel intensity of a channel to the neuron's input. This 3D layer arrangement differs from basic neural networks. Further, CNNs utilize three types of hidden layers: convolution, pooling and fully-connected layers (Figure 3.4). Convolution and pooling layers typically alternate and the last few layers are fully-connected, as in default neural networks. The eponymous convolution layer applies discrete convolutions between a kernel function k and the image I of size $m \times n$ (Equation 3.5a). Commonly, Rectified Linear Unit (ReLu) activation functions and its differentiable approximation are employed, respectively (Equation 3.5b).

$$(I * k)(m, n) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} k(i, j) \cdot I(m - i, n - j) \tag{3.5a}$$

$$ReLu(x) = \max(0, x) \approx \ln(1 + e^x) \tag{3.5b}$$

The second major difference to default neural networks are the shared weights in the form of k. The kernels k are also referred to as self-learned image filters. This sharing saves computation time, since a good filter for an arbitrary position can be assumed to be good for another position, too. The third difference is the local connectivity between the neurons – neighboring neurons react to neighboring image areas. Each applied filter in a convolution layer leads to one so called feature map as result. The CNN automatically learns filters that represent the image best, for instance low-level features like edges and corners in the first layers and high-level features consisting of the low-level ones like structures and textures in subsequent layers. The shared weights inherently lead to translation invariance, since the same filters are applied for all image regions. The pooling layers are meant to reduce the resulting feature maps to save computation time and to tackle overfitting. Most commonly 2×2 max-pooling is utilized to keep the maximum activation from a 2×2 area in the neuron layer. Figuratively, this is similar to keep only the approximative position of the respective feature in the image. All layers previous to fully-connected layers learn object representations. Multiple fully-connected layers in the end perform the actual classification task, whereas the number of neurons in the last layer corresponds to the number of learned object classes C. The final classification result is next calculated by a softmax function $\sigma(\cdot)$, which transforms the class activation $\underline{p}$ into a probability distribution:

$$\sigma(\underline{p})_i = \frac{e^{p_i}}{\sum_{j=1}^{C} p_j}, \text{ with } i = 1, ..., C \text{ and } \underline{p} = (p_1, ..., p_C). \tag{3.6}$$

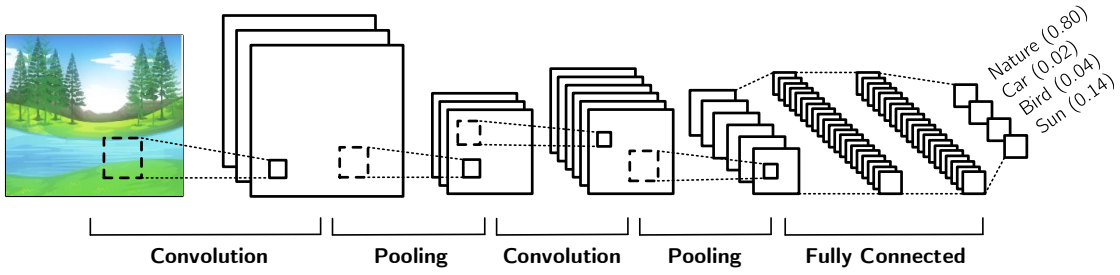


Figure 3.4: Typical structure of a CNN, based on (Britz, 2015; Brgfx, 2019).

3.2.2 YOLO

You Only Look Once and its ameliorations (see Section 2.1) are state-of-the-art real-time object detectors based on CNNs (Redmon, Divvala, et al., 2016). YOLO belongs to the one-shot detectors and requires only one pass through the CNN to calculate object detections. This single pass allows to take the global query image as contextual information into account. The underlying CNN structure supports pruning and addition of layers to trade off additional run-time against accuracy and vice versa.

Figure 3.5 and Algorithm 1 summarize the object detection process of the latest version YOLOv3 on a high level of abstraction. The CNN is trained beforehand on the desired object classes in a supervised manner. Optionally, also object size priors (anchor boxes) can be extracted from the training dataset to boost testing/inference runtime and accuracy. At inference time, first the image is cut into grids (Line 8). The grid size defines the nearest distance between two detectable objects, since YOLO is only able to detect one object per grid cell. Afterwards, the respective object detection proposal BB, together with its probability p_{obj} of detecting an arbitrary object and the respective conditional object class probability vector $\underline{p}_{classes} := [p(class_1|obj) \ ... \ p(class_n|obj)]^T$ are calculated for all grid cells and anchor boxes (Lines 9 – 14). BB consists of two image points which define a rectangular bounding box around the object. The CNN-backbone "Darknet-53" (Redmon, 2013–2016) implements a linear regression to calculate bounding boxes and their localizations inside the respective grid. Mainly the reduction of necessary operations, the larger feature maps for detection, the inclusion of detection in different scales and residual layers enhance the performance of this Darknet version. In Lines 15 and 16, scores p_{obj} below a threshold are neglected, the object class with the highest class probability $\max(\underline{p}_{classes})$ is chosen for the respective grid cell and only the detection with the maximum object class confidence score $p_{obj} \cdot \max(\underline{p}_{classes})$ for objects of the same object class within a small neighborhood retained. This neighborhood depends on the chosen grid size s. Finally, YOLO returns the remaining object detection bounding boxes with their corresponding object class confidence scores. Note that the steps of Lines 8 – 14 are performed implicitly as convolutions.

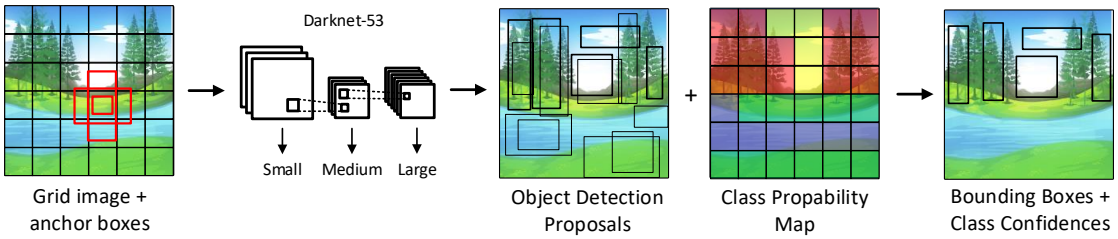


Figure 3.5: Illustration of the YOLOv3 architecture, based on (Redmon, Divvala, et al., 2016; Brgfx, 2019).

3.3 3D Object Representation

This section provides an overview for the chosen object representation techniques in the 3D space. Starting on an low level of abstraction, point clouds are reviewed as the medium to represent basic properties like position and color of real world object points. Subsequently, the more abstract features to describe objects in this work are presented – namely surface normals, keypoints and keypoint descriptors.

Algorithm 1: YOLOv3 inference algorithm (Redmon and Farhadi, 2018).

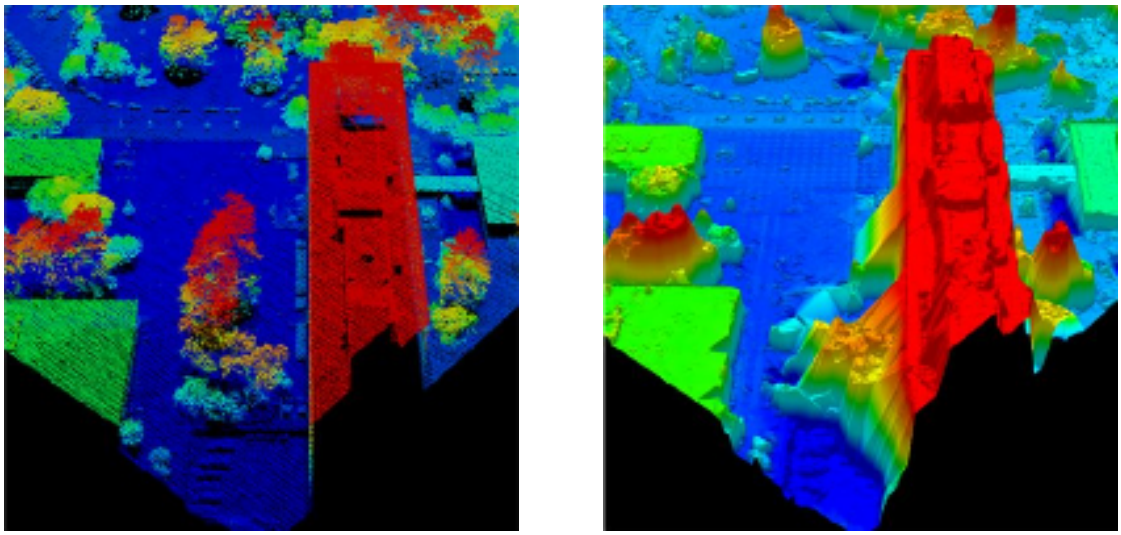
Input: img – query image, $aBoxes$ – anchor boxes, w – weights;
Parameters: s – grid size, τ_1 – objectness threshold,
τ_2 – prediction threshold;
Constants: $d_1 \coloneqq 32$, $d_2 \coloneqq 16$, $d_3 \coloneqq 8$ – downsampling factors;
Output: $objDetections$ - set of object detections;

$objDetections \leftarrow [\,]$;
$imgGrids \leftarrow \text{sliceImageToGrid}(img, s)$;
$F_{small} \leftarrow \text{calculateDownsampledFeatures}(imgGrids, w, d_1)$;
$F_{medium} \leftarrow \text{calculateDownsampledFeatures}(imgGrids, w, d_2)$;
$F_{large} \leftarrow \text{calculateDownsampledFeatures}(imgGrids, w, d_3)$;
for i ***in*** $imgGrids$ **do**
    for a ***in*** $aBoxes$ **do**
        $(BB, p_{obj}, \underline{p}_{classes}) \leftarrow \text{linRegression}(F_{small}[i], F_{medium}[i], F_{large}[i], w, a)$;
        if $p_{obj} > \tau_1$ **then**
            $objectDetections.\text{append}\big((BB, p_{obj}, p_{classes})\big)$;
$objDetections \leftarrow \text{nonMaxSuppression}(objDetections, \tau_2)$;
return $objDetections$;

3.3.1 Point Cloud

A point cloud can be interpreted a set of data points in the same coordinate system and is often intended to represent a sampled object surface. The point cloud's characteristics major differentiate depending on whether they are captured by active or passive sensors.
Light Detection and Ranging (LIDAR) denotes a popular active optical sensing method. The thereby actively illuminated light allows the sampling of uniform-textured or texture-less areas (e.g. in vegetation or urban areas) and provide precise point coordinate locations, without the need of post-processing. Depending on the illuminated surface, LIDAR light can penetrate certain materials and provide information from underlying object layers (for instance the surface within and under vegetation canopies), which supports dense sampled point clouds.[1] In comparison to passive sensors, active sensors are less sensitive to noise and therefore more precise, because of their own illumination source. On the other hand, stereo cameras as passive sensors are less expensive and can capture 2D scenes easily, without the need of moving parts (Harwin and Lucieer, 2012).

[1] Further conditions for the penetration capability are for instance the sensor lights' wavelength, the illumination power, the incidence angle and environmental conditions.



(a) LIDAR acquisition. (b) Stereo view acquisition.

Figure 3.6: Point clouds of the same scene acquired by LIDAR (a) and stereo imagery (b). The LIDAR point cloud is dense and sharp, the interpolated stereo view cloud sparse and rough. The color indicates the elevation of the scene (Basgall et al., 2014).

3.3.2 Surface Normal and Curvature Estimation

The pioneering work of (F. Stein and Medioni, 1992) introduces surface normals and object shape curvature information as basis for 3D object representations – they are still in use nowadays for higher level object features (see Section 3.3.3 or 3.3.4). Surface normals can describe the local orientation of a surface and the curvature behavior (Watt, 1993). Due to the surface sampling of point clouds, normals can only be approximated. A simple method is to determine the plane tangent through the following least-square plane fitting estimation (Berkmann and Caelli, 1994):
A surface normal of a specific point in a point cloud is defined as a vector perpendicular to the tangential plane at the points' surface. Each point $(x, y)^T$ on an arbitrary surface $ax + by + c$ defines two surface normals n_1 and n_2:

$$n_1 := (-a, -b, 1)^T \qquad n_2 := (a, b, -1)^T. \tag{3.7}$$

For each 3D point p_i, the covariance matrix C of the k nearest neighbors describes the local scattering. The eigenvectors v_j and eigenvalues λ_j of C indicate the direction of the scattering in the 3D space – they can be determined by the PCA (Pearson, 1901) of C (Equation 3.8). The two v_j corresponding to the two largest λ_j approximate the local surface tangential plane, while the v_j of the smallest λ_j estimates the normal, since all v_j are orthogonal to each other for the symmetric matrix C.
To determine the correct normal out of the two possibilities (Equation 3.7) and

avoid orientation inconsistency, an acquisition viewpoint v_p is needed, so that p_i has to satisfy the constraint in Inequality 3.8c. This constraint does not hold for merged point clouds with multiple viewpoints, though (Klasing et al., 2009).

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p}) \cdot (p_i - \bar{p})^T, \quad \text{with } \bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{3.8a}$$

$$C \cdot v_j = \lambda_j \cdot v_j, \quad \text{with j} \in \{0, 1, 2\} \text{ and } \lambda_0 < \lambda_1 < \lambda_2 \tag{3.8b}$$

$$n_i \cdot (v_p - p_i) > 0 \tag{3.8c}$$

Having the eigenvalues and -vectors, (Pauly et al., 2002) proposes to estimate the local curvature σ_{p_i} around p_i in the direction of its normal by:

$$\sigma_{p_i} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}. \tag{3.9}$$

Both, the surface normal direction and likewise the curvature depend on the beforehand chosen k, where k corresponds to the desired level of detail. A small k makes the procedure influenceable to noise and outliers. A large k might lead to a distorted curvature estimation and a loss of details (see Figure 3.7). More general than choosing a k, a search radius r can be provided to determine a k dynamically.

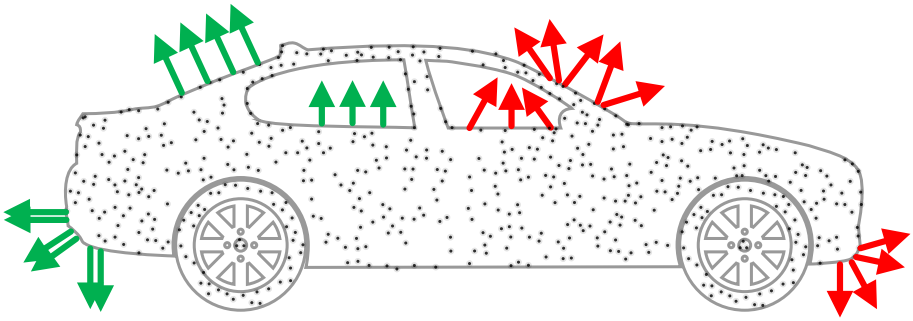


Figure 3.7: The green normals indicate an appropriate choice of the amount of nearest neighbors k. The k of the red ones is chosen too large – normals near edges are rotated and their directions scatter, which lead to smeared edges and a loss of fine details (Yousif et al., 2014).

3.3.3 ISS Keypoint Detection

Intrinsic Shape Signatures describe keypoints for 3D object recognition which are designed to be viewpoint insensitive, repeatable and discriminative (Zhong, 2009). The modified ISS algorithm used in this work is part of the point cloud library (R. B. Rusu and Cousins, 2011) and implements only the detection part. The ISS keypoint detection algorithm works directly on noisy, incomplete point clouds.

To obtain repeatable keypoint detections, ISS eigen-analyzes local point scatter statistics for each cloud point, because the statistic is assumed to be independent of the noisy sensor sampling process. The eigenvalues $\lambda_0, \lambda_1, \lambda_2$ for the local scatter matrix of a point p are calculated as in Equation 3.8, where p is here the center of C. A point p is considered an ISS keypoint, if it fulfills the following conditions:

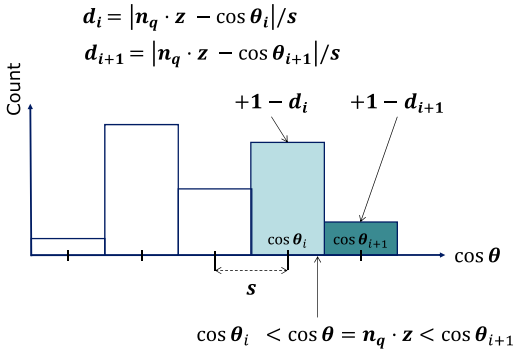
1. $\forall q \in \{q : |p - q| < r_n\} : \lambda_0(p) > \lambda_0(q)$,
2. $\nexists q_j \in \{q_j : |p - q_j| < r_{bn}\} : \forall q_k \in \{q_k : |q_j - q_k| < r_b\} : |N(q_j) - N(q_k)| < \tau_\theta$,
3. $\frac{\lambda_2(p)}{\lambda_1(p)} < \gamma_{21}$ and $\frac{\lambda_3(p)}{\lambda_2(p)} < \gamma_{32}$,

where r_n, r_{bn}, r_b, τ_θ, γ_{21} and γ_{32} are user-defined threshold parameters, $N(\cdot)$ depicts the surface normal of a point and $\lambda_i(\cdot)$ denotes the i-th eigenvalue of a point's local scatter matrix. The first condition performs a non-maxima suppression in the local neighborhood and keeps points with the largest neighborhood scattering in all directions. The second condition filters surface boundary points – in benchmarks (see Section 2.4), this condition leads to more robust keypoints with a low runtime overhead. The third condition excludes points with similar spread along their principal directions, for instance points in homogeneous areas.

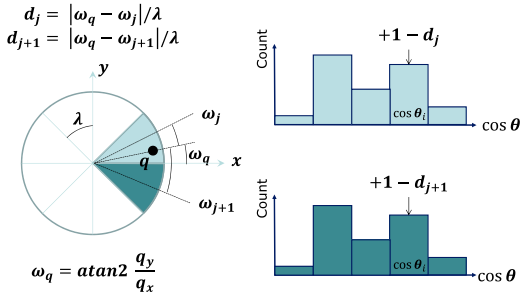
3.3.4 CSHOT Feature Description

Color Signature of Histograms of Orientations (Tombari et al., 2011) extends the SHOT 3D feature descriptor (Tombari et al., 2010) with color/texture information. As the name implies, SHOT combines potentially highly descriptive signatures and robust histograms of surface normal orientations, while focusing on maintaining computational efficiency.
Like the ISS keypoint detector in Section 3.3.3, SHOT eigen-analyzes the local scattering to create a repeatable local reference frame for each point p. In the next step, this frame is partitioned in bins along the local azimuth, elevation and radius distance directions, which are defined in terms of the local eigenvectors. In the Figures 3.8b – 3.8d, the different shaded areas in the circles represent bins for the respective direction, with the respective bin sizes s, λ, ψ or R. Then, for each bin a histogram of the local neighborhood normal orientations is calculated. These histograms contain the cosine values of the angles θ between each local neighbor normal n_q and the local z-axis, as illustrated in each subfigure of Figure 3.8, when ignoring all other parameters for now. All histogram bins are next interpolated between p and each neighbor point q, by adding a value $1 - d$ to each bin count. The respective distance d thereby is calculated in terms of the position of q inside its bin and in terms of the bin spacings, which means that d is normalized by the respective distance between two adjacent bins (Figure 3.8). After the interpolation process, the histograms are accumulated to form the SHOT signature. The result is additionally normalized to sum up to one by dividing each bin by its inverse local point density and its volume, to acquire invariance against point density differences.

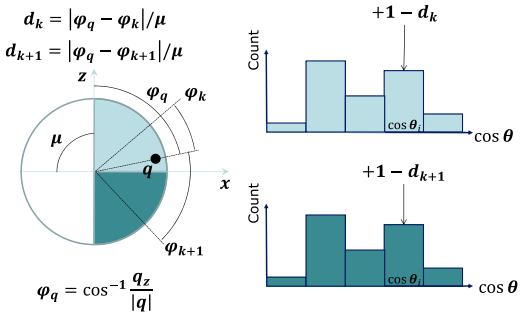
The color signature follows the same bin structure and consists of histograms, where colors of the $CIELab$ space (CIE, 1977) are compared regarding their L_1 norm. The SHOT signature concatenated with the color one then results in the final CSHOT descriptor. Each CSHOT descriptor can be considered as a high dimensional vector. The lower the Euclidean distance between two CSHOT descriptors, the higher their matching score.



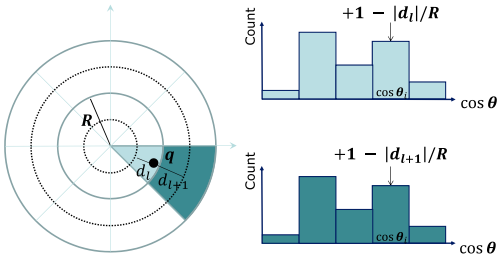
(a) Interpolation on normal cosines.



(b) Interpolation on azimuth.



(c) Interpolation on elevation.



(d) Interpolation on radius distance (d denotes the Euclidean distance).

Figure 3.8: SHOT quadralinearly histogram interpolation (Salti et al., 2014).

3.4 3D Object Pose Tracking

"Tracking an object in a video sequence means continuously identifying its location when either the object or the camera are moving" (Lepetit, Fua, et al., 2005, p. 1). As stated in the scope of this work in Section 1.3, the tracked objects are assumed to be rigid and non-moving – only the camera is moving. Regarding the literature, 3D object tracking denotes the task to recover all 6DoF in a 3D space, more precisely, the translation and rotation parameters for each direction. The 6DoF define the relative camera position and orientation to the scene, or equivalently, the 3D pose

of an object relative to the camera (Lepetit, Fua, et al., 2005). In Computer Vision, the combination of translation and rotation of an object is referred to as its pose. In this work, the terms "3D object tracking", "3D tracking" or "tracking" denote synonyms for 3D object pose tracking. The prefix 3D is used to explicitly refer to the Euclidean 3D space and to dissociate these methods from the 2D image space applications.
First the extended particle filter is reviewed and afterwards Visual Odometry needed for the evaluation.

3.4.1 KLD Particle Filter

The notion behind the Particle Filter is to estimate a true state S^t of a dynamical system using sensor measurements M^t, where the state space at time t is represented by an underlying probability distribution $\hat{P}(S^t)$ approximated by a set of discrete samples (particles) (Kitagawa, 1987; Arulampalam et al., 2002). In an object tracking navigation scenario, $P(S^t)$ describes the true object/camera pose probability distribution. A state $S^t := (\underline{s}, \underline{w})_t$ consists of samples $\underline{s}$ and corresponding weights $\underline{w}$. The samples $\underline{s}$ represent translation and rotation estimations of the true object position along each 3D space axis and their weights $\underline{w}$ indicate the likeliness. Since $P(S^t)$ is unknown, it is approximated through $\hat{P}(S^t)$ by evaluating the samples distributed in the state space, whereby the particles focus on regions with high likelihood. As a member of the Bayesian frameworks, the PF aims to estimate the Maximum Likelihood Estimate, i.e. the best object pose guess of the sampled state space, conditioned on the data collected so far. The following equations describe the relation between a S^t and the sensor measurements M^t:

$$M^t = f(S^t) + N_1^t, \tag{3.10a}$$
$$S^{t+1} = g(S^t) + N_2^t, \tag{3.10b}$$

where $f(\cdot)$, $g(\cdot)$ denote non-linear functions and N_1^t, N_2^t random noise (for instance zero-mean Gaussian noise). The function $f(\cdot)$ can for example be dependent on the similarity of the sample's neighborhood to the tracked object, while $g(\cdot)$ can predict the next state estimate using the momentum of the particles movement over time. Both contribute to the weight of a particle. For each state, c_i^t with $i = 1, ..., n$ stores the cumulative weight to lower computational complexity, where $c_n^t := 1$. New samples $\underline{s}^{t+1}$ are drawn from $\underline{s}^t$ based on different sampling methods – the most common one is importance sampling (Yilmaz et al., 2006, p. 21):

Selection. Select n samples $\hat{s}_{i=1,...,n}^t$ from S^{t-1} by generating a random number $r \in [0, 1]$, finding the smallest $j = 1, ..., n$ such that $c_j^{t-1} > r$ and set $\hat{s}_i^t = s_j^{t-1}$.

Prediction. For each selected sample s_i^t with its w_i^t, apply Equation 3.10b.

Correction. Compute new weights w_i^{t+1} to the new predicted samples s_i^{t+1}, by applying Equation 3.10a with the new measurements M^{t+1}.

The basic PF algorithm always handles the full amount of samples n. The work of (Fox, 2002) introduces an adaptive number of samples in each selection step at runtime, based on a KLD evaluation. They propose to add new samples, if the samples scatter in the search space and remove low-weighted ones otherwise. They have shown that

$$n \approx \frac{k-1}{2\epsilon} \cdot \left\{ 1 - \frac{2}{9 \cdot (k-1)} + \sqrt{\frac{2}{9 \cdot (k-1)}} \cdot z_{1-\delta} \right\}^3 \tag{3.11}$$

can ensure, that with probability $1 - \delta$ the KLD between the MLE of the samples and the true distribution is approximately less than the desired error bound ϵ. The approximation of the true distribution is only dependent of k, $z_{1-\delta}$ and ϵ, where the first denotes the number of bins in the discretised state space containing at least one sample (i.e. non empty bins) and the second the upper $1 - \delta$ quantile of the standard normal distribution $\mathcal{N}(0, 1)$. Figure 3.9 illustrates the KLD PF process.

3.4.2 Visual Odometry

The term Visual Odometry (VO) refers to the process of determining the ego-motion of an agent, where only the input of mono/multi cameras fixedly attached to it are used (Nister et al., 2004). VO operates incrementally using successive images to form a trajectory and requires an adequate illumination, static scene elements and a scene overlap in the images, i.e. a sufficient frame rate to ego motion speed ratio. Its worth to mention the closely related problems of Structure from Motion (Longuet-Higgins, 1981; Harris and Pike, 1988) and Visual Simultaneous Localization and Mapping (Durrant-Whyte and Bailey, 2006). Actually, VO is a particular case of the former, focusing on images sequentially and in real time. However, VO distinguishes from the latter by enforcing local consistency of the estimated trajectory and not taking global consistency constraints into account (Scaramuzza and Fraundorfer, 2011). Its local consistency point of view leads to drift and the trajectory diverges over time, though. A common method to tackle this problem is windowed-bundle adjustment, which minimized the re-projection errors of common reconstructed 3D points of the last m images (Mouragnon et al., 2006).
Stereo cameras (or multi cameras in general) are preferred over mono cameras for VO, as the latter has to perform the task of 3D structure computation in addition to the relative motion, to infer depth information, and the absolute scale of the environment is unknown using two images. For the scale determination, it requires three images and either knowledge of 3D structures or a trifocal tensor (Hartley and Andrew Zisserman, 2003, pp. 365–406).
Works to VO can be separated into three categories: feature-based methods, appearance-based methods and hybrid methods, that combine both. The first ones are based on feature detectors, finding salient and repeatable features, that are matched

between the frames. Appearance-based methods exploit intensity information of all pixels in the whole image or of sub-regions. Independent of which pixels are actually used, the task is to calculate a 3D transformation $E := [R|t]$ between the images (see Figure 3.10) – similar to the problem of determining $E_{CC'}$ from Figure 3.1. To find a common transformation/motion between the images, all single pixel transformations can be accumulated or outliers can be rejected beforehand. The actual method to do so, depends on whether the correspondences between features $\underline{f}^{k-1}$ and $\underline{f}^{k}$ are stated in 2D or 3D:

2D-to-2D. $\underline{f}^{k-1}$ and $\underline{f}^{k}$ are stated in image coordinates and the matrix E is calculated for instance using Nistér's five-point algorithm (Nistér, 2004).

3D-to-3D. $\underline{f}^{k-1}$ and $\underline{f}^{k}$ are stated in 3D, for example using Stereo Vision. At each time step, 3D points are triangulated. A general solution is to find the transformation, that minimizes the L_2 distance between both 3D point sets, e.g. by using the Iterative Closest Point algorithm (Besl and McKay, 1992).

3D-to-2D. $\underline{f}^{k-1}$ is stated in 3D, while $\underline{f}^{k}$ is considered as the corresponding re-projections onto the image. This is more accurate than the former methods, since it searches the transformation minimizing the image re-projection error, instead of the 3D point position error, also known as the perspective-n-point problem (Fischler and Bolles, 1981).

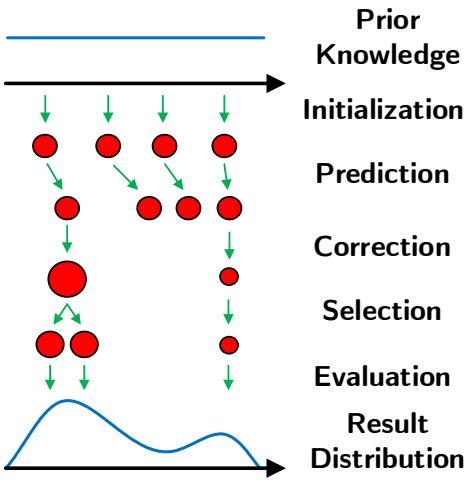


Figure 3.9: Visualization of KLD Particle Filtering. Red circles denote particles and green arrows transitions between the algorithmic steps. Based on (S.-r. Yi and Song, 2018, p. 4).

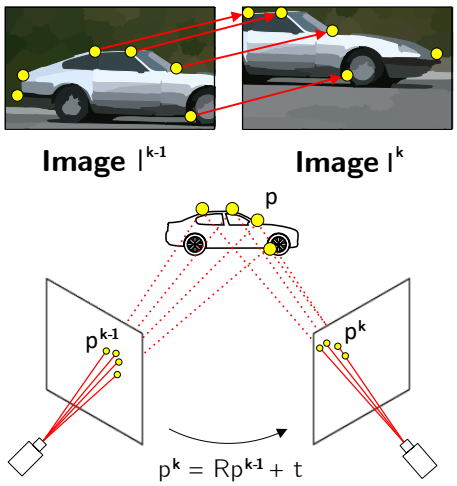


Figure 3.10: Visualization of feature-based Visual Odometry. Yellow circles denote feature points. Based on (Scaramuzza, 2015).

3.5 IPS

The Integrated Positioning System combines multiple sensor systems to estimate the ego-motion and reconstruct the environment, without the need for environmental assumptions or global referencing (Grießbach et al., 2012). The Figure 3.11 illustrates an exemplary IPS output. It is mainly developed for the deployment in closed areas like buildings and the outer space, where navigation by global systems fail.
As the core components, the IPS utilizes an Inertial Measurement Unit (IMU) and a stereo camera system. The former estimates a translation and orientation (pose) change between two time steps, within a local reference coordinate system. The latter captures stereo image pairs. The stereo images and the respective pose change are then fed into an ego motion estimation, that performs Visual Intertial Odometry (VIO). In VIO, pose estimations of the IMU and the VO are combined to approach more accurate results. In the IPS VIO implementation, the IMU pose change estimations are utilized as a prior, to restrict the search area for the pixel matching in the VO and it outputs a trajectory as a function of the time steps. The combination of IMU and VO leads to saved computation resources and higher precision, by using low cost hardware. Simultaneously, a depth map is created out of the stereo image pair – the processes are introduced in the previous Sections 3.1.2 – 3.1.3. The depth images and the trajectory form point clouds (Section 3.3.1), to reconstruct the environment piecewise. The IPS additionally assigns timestamps to each created data and synchronize them, whereby a central clock provides the first ones.
Further sensors can be deployed in the IPS, to accomplish higher precision, reliability and integrity. The Figure 3.12 depicts the sensor data processing chain on a high level of abstraction and the Figure 3.13 shows two versions of portable IPS systems.

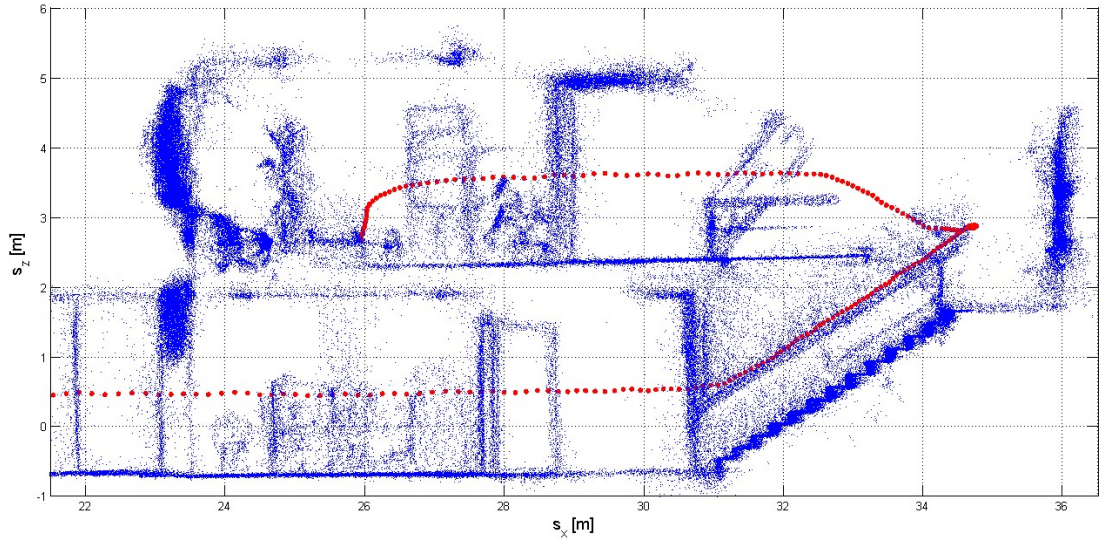


Figure 3.11: Exemplary IPS output (side view) – blue dots represent a point cloud and red ones the trajectory over time (Grießbach et al., 2012).

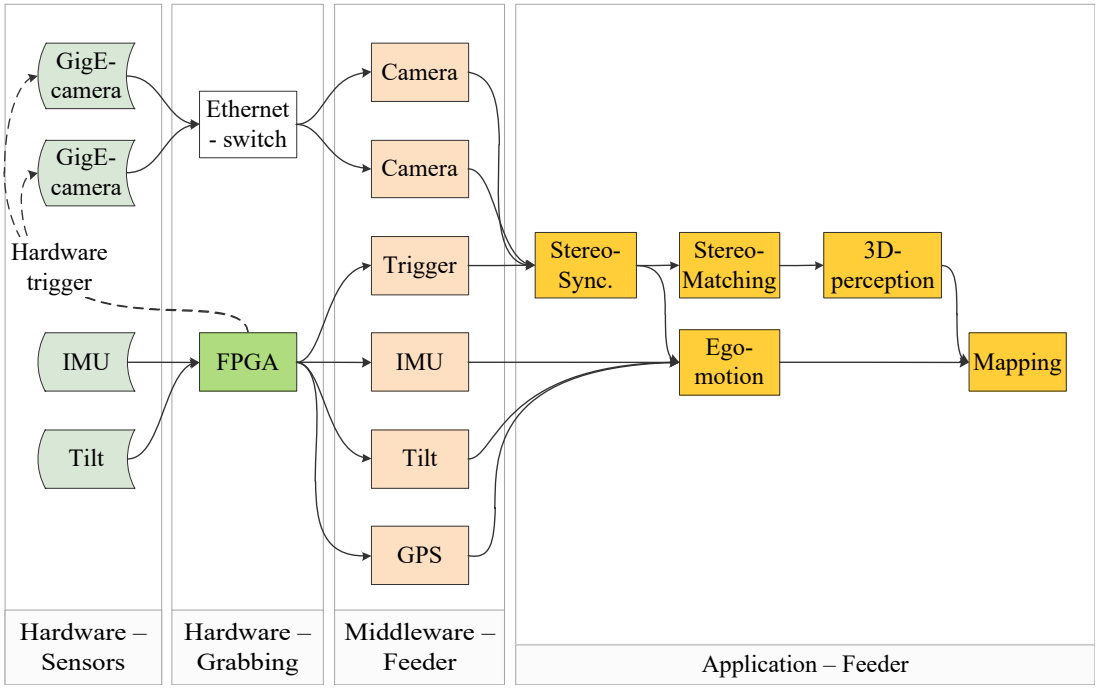


Figure 3.12: Illustration of the IPS sensor data processing chain (Grießbach et al., 2012).

Figure 3.13: Visualization of the IPS. There are IPS versions in different sizes for different application scenarios. The left image illustrates the used prototype for this work, the right image a smaller commercial prototype (Börner et al., 2017).

Method

This chapter proposes a combination of a 3D object pose tracker and a 2D object detector, referred to as CODAPT, as a part of Research Question (1). Besides, CODAPT serves as the method to evaluate the Research Questions (2) and (3), whether the 2D object detector can improve the accuracy and runtime performance of a 3D object tracker and vice versa, the 3D object tracker can improve the accuracy of the 2D object detector. First, an overview is given to the functional principle of CODAPT. Afterwards, the functional behavior of the single components and their interaction are specified and justified in detail.

4.1 Overview

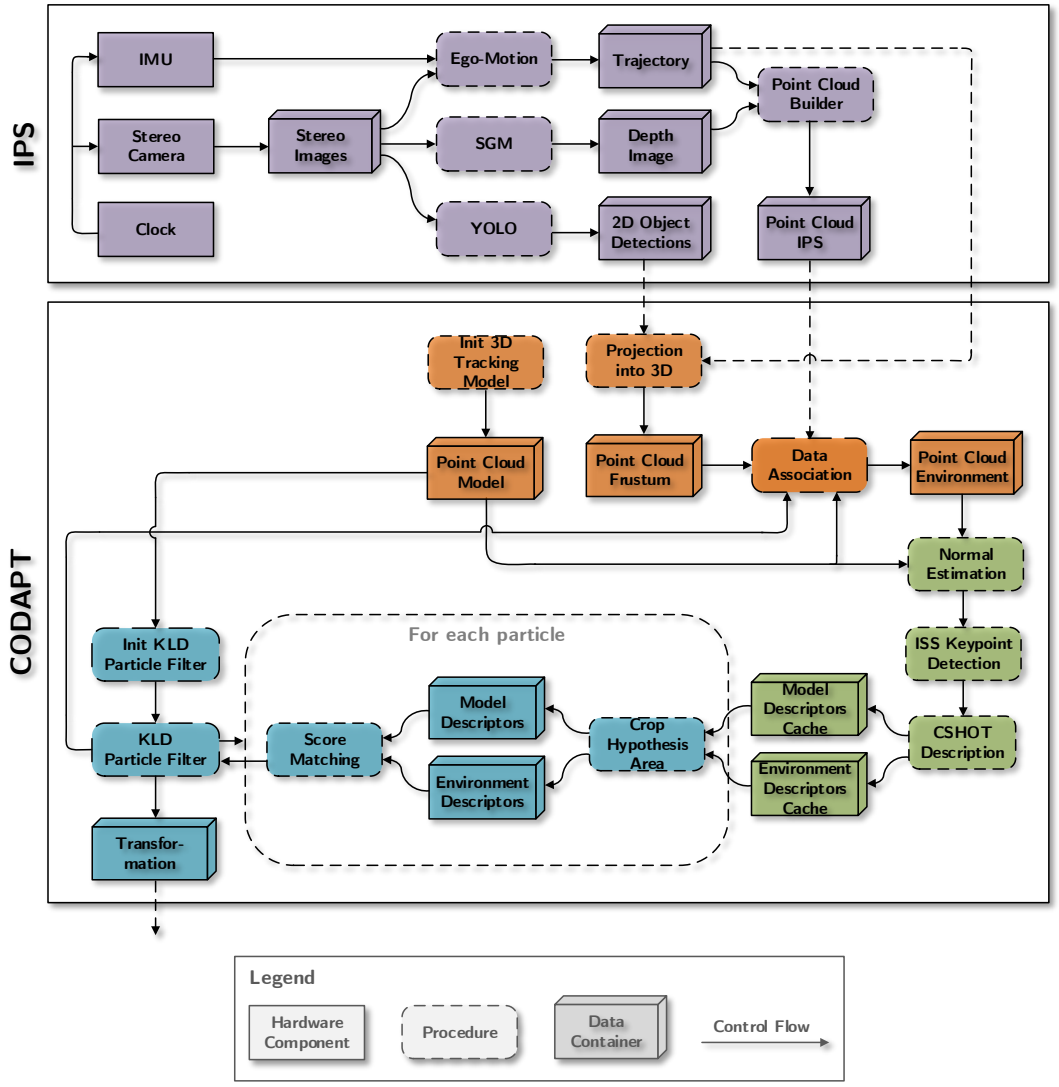


Figure 4.1: CODAPT system architecture. Violet components belong to the input parameter processing by the IPS, orange ones to point cloud preprocessing, green ones to feature detection and description, and the blue components to KLD particle filtering.

The Figure 4.1 summarizes the system architecture and the processing flow of CODAPT for one time step t, whereby initialization procedures are performed once. An IPS session is defined for $t = 0, ..., n$. For each component group, necessary parameters, input-output-relations and brief discussions are considered in the respective subsections.
The IPS is used to acquire stereo images and related pose changes during an IPS session, to form the input parameters for CODAPT: a trajectory, a point cloud and 2D object detections for a time stamp t. The process is described in Section 3.5.
The components of CODAPT in Figure 4.1 are divided into three groups: point cloud pre-processing, feature detection and description, and KLD particle filtering. In the first group, the object model point cloud pc_{model} of the object to track and a point cloud from the environment pc_{ips} captured by the IPS, are prepared. First, the beforehand given object model point cloud is loaded from a storage device into pc_{model}, whereby it contains information for the initial starting position for the respective tracking scenario and its object class. Next, each 2D object detection and the related IPS localization are projected into a point cloud $pc_{frustum}$, which forms a frustum. Then, the data association follows. The main goal of this component is to find and match an object detection to the currently tracked object at t and discard others. Therefore, the frustum point cloud is tried to be associated to the last known object tracking result $pc_{lastTrackingResult}$ from $t - 1$. As initial tracking result, the loaded object model point cloud is used. An object detection is successfully associated to the tracked object, if their object classes equal and $pc_{frustum}$ contains a certain amount of $pc_{lastTrackingResult}$. In the case of a successful association, $pc_{ips} \cap pc_{frustum}$ then results in the environment point cloud pc_{env} for further processing. Otherwise, pc_{ips} is passed unmodified as pc_{env} to the next component.
The second group of CODAPT contains the modules for the feature detection and description. Since the last two components both rely on surface normals, first surface normals are estimated (Section 3.3.2) and added to pc_{model} and pc_{env}. Afterwards, ISS keypoints are detected (Section 3.3.3) for the point clouds and CHSOT descriptors calculated (Section 3.3.4) for these keypoints. The descriptors of pc_{model} are cached over the whole tracking procedure, while the ones for pc_{env} are cached for one tracking iteration, until a new environment frame is available.
Finally, the third group contains all components related to the KLD particle filter. At $t = 0$, the PF and its particles are initialized. For $t > 0$, the PF performs the steps introduced in Section 3.4.1. Each particle represents a hypothesis for the object pose within the current pc_{env}. So for each particle, its hypothesized area within the environment point cloud is cropped and the respective subset of the descriptor cache is obtained. Each particle then compares its model descriptors to the environment descriptors, to score the matching. The pose hypothesis from the particle with the highest matching score leads to the final transformation for the camera movement between $t - 1$ and t. The PF then feeds the point cloud for the tracking result back to the data association component for the next time step $t + 1$.

4.2 CODAPT Design Specification

According to Figure 4.1, the system architecture of the proposed method CODAPT is separated into four groups. In this section, these grouped components are specified and design choices justified. Appendix D provides a full list of external requirements.

4.2.1 Input Parameter Processing

Algorithm 2: CODAPT input parameter processing.

1 **Input:** - ;
2 **Parameters:** (Appendix B.1);
3 **Output:** $objDetections$ - object detections, pc_{ips} - point cloud,
4 $traj$ - trajectory;
5
6 $t \leftarrow$ generateTimeStamp();
7 **if** $t == 0$ **then**
8 $traj \leftarrow$ initTrajectory();
9 **return**;
10 $img_C, img_{C'} \leftarrow$ acquireStereoImage(t);
11 $\Delta pose_{imu} \leftarrow$ getIMUEstimation($t-1, t$);
12 $traj(t) \leftarrow$ VIO($\Delta pose_{imu}, img_C, img_{C'}$);
13 $depthImg \leftarrow$ SGM($img_C, img_{C'}$);
14 $pc \leftarrow$ buildPointCloud($depthImg$);
15 $pc_{ips} \leftarrow$ transformPointCloud($pc, traj$);
16 $objDetections \leftarrow$ YOLO(img_C);
17 **return** $objDetections$, pc_{ips}, $traj$;

The IPS functionality is created and provided by the DLR – the functionality remains unchanged, only parameters are adjusted for this work. A full list of the used IPS parameters can be found in Appendix B.1. Ground truth values are denoted with the index gt in the following. This section follows the Algorithm 2, which is visualized in the Figure 4.2.
The Lines 6 – 12 denote the output of the IPS hardware components and the VIO, which are explained in Section 3.5 and 3.4.2, respectively. The initial entry of $traj(0)$ is defined as zero for all translation entries, which describes the start at the origin of the local IPS coordinate system. The orientation values for $t = 0$ describe the initial IPS orientation.
Simultaneously to the VIO, SGM produces a depth image with gray value information out of the stereo image pair (see Section 3.1.3). The parameters for SGM can be found in Table B.1.6. The introduced memory efficient GPU implementation of SGM (eSGM) is used for CODAPT to allow larger disparity values in close range areas. A general limitation of SGM is its weakness in homogeneous areas, where

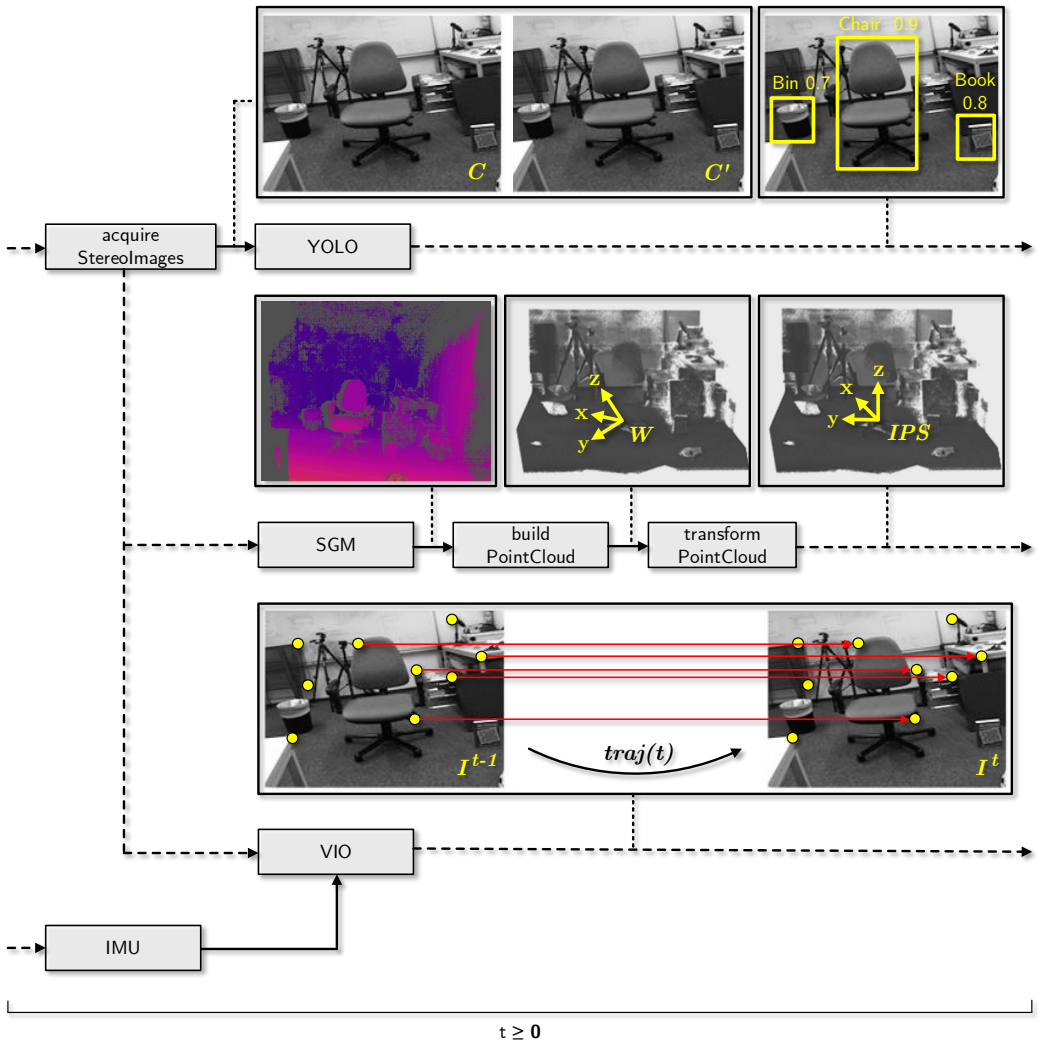


Figure 4.2: Visualization of the CODAPT input parameter processing. The notation follows the one of Figures 3.1 and 3.10.

no depth values can be calculated. This limitation can be neglected here, since the proposed ISS keypoint detection ignores homogeneous areas by design (see Section 3.3.3). For denser point clouds and a potentially more robust CSHOT feature description (the more points in the range of CSHOT features, the better), the depth maps are post-processed using a 3×3 median filter. This small kernel is assumed to preserve contour and texture features, which are necessary for the ISS keypoint detection. Despite the supplementary computational overhead, the gain of a richer feature description is more favorable, though.

The depth image is then transformed into a point cloud representation (Line 14) with as few loss of information as possible: a leaf size of 0.01 m (Table B.1.7) defines the maximum point cloud resolution δ_{PC} and the minimum allowed two points per

voxel are chosen to filter outliers and noise. The depth of the point cloud in looking direction is limited to a maximum of 10 m. Due to the low baseline of 0.23 m of the applied IPS, a reliable stereo vision is only possible in close range. The work of (Irmisch, 2017) shows the relation between a disparity deviation of ± 1 px and resulting distance deviations as a function of viewing distances for a comparable IPS setup. These results also underline the necessity of the applied parabola subpixel interpolation of SGM. The resulting point cloud pc is then transformed into the local coordinate system of the IPS as pc_{ips}, using $traj$ (Line 15). With this transformation, CODAPT can take advantage of the already available IPS' navigation capability and hence restrict the object tracking search space.
In Line 16, the left camera image is meanwhile fed into YOLO, to determine known object classes. In this work, pre-learned weights and object classes are used with the proposed default YOLOv3 configuration of Table B.1.8. The object detections, the transformed point cloud and the modified trajectory are finally passed to the point cloud pre-processing group.

4.2.2 Point Cloud Pre-Processing

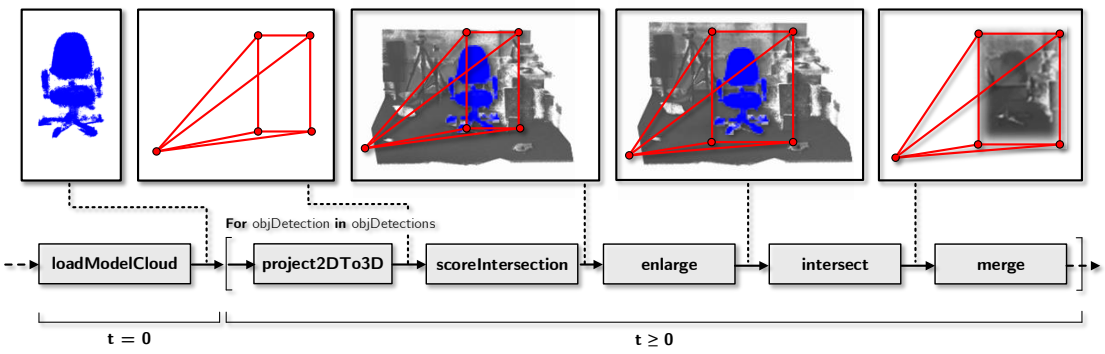


Figure 4.3: Visualization of the CODAPT point cloud pre-processing.

This section follows the Algorithm 3, illustrated in the Figure 4.3. The algorithm takes basically the output of Algorithm 2 as input. As parameters, the projection matrix E_{CI} between the left camera's image plane and the IMU is needed (see Table B.1.5) and the ones listed in Appendix B.2.
At first, the algorithm prepares the variables in the Lines 7 – 12. The point cloud pc_{env} is initialized and the current time stamp t acquired from the input point cloud. At $t = 0$, the object model point cloud to track is loaded from a storage device and have to fulfill three preconditions. First, the respective object model point cloud is assumed to have the same leaf size 0.01 m as pc_{ips} to accomplish equality, when it comes to CSHOT descriptor comparability. It is further assumed to be acquired independently to the current tracking dataset, to avoid model overfitting. As the

Algorithm 3: Point Cloud Pre-Processing.

```
1  Input: pc_ips - cloud from the IPS, objDetections - 2D object detections,
2         traj - IPS trajectory, lastTrackingPose - last tracking pose;
3  Parameters: E_CI - homographic projection matrix from C to IMU,
4              (Appendix B.2);
5  Output: pc_model - object model cloud, pc_env - environment cloud;
6
7  pc_env ← initPointCloud();
8  t ← getTimeStamp(pc_ips);
9  if t == 0 then
10     pc_model, objClass_model ← loadTrackingObjectModel();
11     return;
12 E_CIPS ← traj(t) · E_CI;
13 pc_lastTrackingResult ← applyTransformation(pc_model, lastTrackingPose);
14 for objDetection in objDetections do
15     objClass_det ← getObjectClass(objDetection);
16     if objClass_det == objClass_model then
17         pc_frustum ← project2DTo3D(objDetection, E_CIPS);
18         intersectionScore ← scoreIntersection(pc_frustum, pc_lastTrackingResult);
19         if intersectionScore ≥ τ_intersectionScore then
20             pc_enlargedFrustum ← enlarge(pc_frustum, intersectionScore);
21             pc_intersection ← intersect(pc_enlargedFrustum, pc_ips);
22             pc_env ← merge(pc_env, pc_intersection)
23 return pc_model, pc_env;
```

third precondition, the point cloud has to contain its starting pose, as presumed in Section 1.3. Chapter 5 describes how to satisfy the latter constraint. In Line 12, the multiplication of the homogeneous transformation between the left camera's image plane to the IMU of the IPS and the homogeneous transformation between the IMU to the local IPS coordinate system stored in *trans*, leads to the transformation from the image plane to local coordinate system E_{CIPS}. The $pc_{lastTrackingResult}$ is prepared by applying the $lastTrackingPose$ to the pc_{model} – if there is no $lastTrackingPose$, the last known object position (respectively the initial object position) is taken.
Subsequently, each object detection is prepared for the data association. Therefore, the object class of the object detection is compared to the one of the tracking object model in Line 16. Since all other object classes are rejected by YOLO in advance, this treatment ignores low probable but correct detections, however the inclusion of the less probable object classes is neglected here and left for future works. If both classes equal, the object detection is projected into a point cloud $pc_{frustum}$

using E_{CIPS} (Line 17), as depicted in Figure 4.3. Together with the IPS' capturing position, this point cloud consists of five points and forms a frustum. The lines from the IPS position point to the object detection corners are extended to 10 m, to match the maximum IPS point cloud depth. Next, the data association step follows in Line 18. The data association is considered as success, if the following inequality holds:

$$\frac{\#(pc_{frustum} \cap pc_{lastTrackingResult})}{\#(pc_{lastTrackingResult})} \geq \tau_{intersectionScore}, \tag{4.1}$$

with a predefined $\tau_{intersectionScore}$ and $\#(\cdot)$ counting the number of points in a cloud. To compare both point clouds to each other, their convex hulls are calculated as an approximation and intersected. The points of the result point clouds with respect to the overall points of $pc_{lastTrackingResult}$ then lead to the intersection score. The intersection score threshold is chosen as 0.4 to incorporate imperfect object detection- and object tracking results, which might not exactly fit the true object position. Furthermore, for 0.4, the probability of multiple object detection hits raises, but on the other hand, the missing rate is assumed to be lowered. A miss is assumed to be worse than multiple association hits, since even multiple hits can still shrink the considered point cloud area. This data association is discussed in Chapter 7.
In the case of a successful data association, the frustum potentially needs to be corrected, to prevent a too small or displaced object detection, which would result in a insufficient coverage of the tracking object in the current pc_{ips} frame. Therefore, the 3D bounding box is enlarged by the factor of $1.0 - intersectionScore$ in x and y direction, i.e. the four 3D bounding box points are shifted, respectively (see Figure 4.3).[1] The intersection between the convex hulls of the enlarged frustum point cloud and pc_{ips} then forms the point cloud section with the detected object. In the case of multiple associated object detections, all are merged together to the final pc_{env} point cloud (Lines 19 – 22).
Both, the loaded object model point cloud pc_{model} and the cropped environment point cloud pc_{env} are passed as results to the feature detection and description component group. The basic point cloud operations for convex hull wrapping, convex hull intersection, transformation and merging are utilized as independent modules from the PCL framework.

4.2.3 Feature Detection and Description

This section follows the Algorithm 4, illustrated in the Figure 4.4. The algorithm takes the output of Algorithm 3 as input. The needed input parameters for this group are listed in Appendix B.2.
The algorithm performs the processing pipeline of Figure 4.4 once for the object model pc_{model} at $t = 0$ (Lines 6 – 11) and periodically for pc_{env} for every $t > 0$ (Lines

[1] The notation of x and y directions are chosen for simplicity. In the local coordinate system of the point cloud frame, the direction vectors for the shift are determined by connecting the upper and the lower points of the 3D bounding box, respectively. Each point is then shifted away from the respective connected point.

Algorithm 4: CODAPT feature detection and selection.

1 **Input:** pc_{model} - object model cloud, pc_{env} - environment cloud;
2 **Parameters:** (Appendix B.2);
3 **Output:** $descriptorCache_{model}$ - model CSHOT descriptor cache,
4 $descriptorCache_{env}$ - environment CSHOT descriptor cache;
5
6 **if** $t == 0$ **then**
7 $pc_{modelN} \leftarrow$ estimateAndAddNormals(pc_{model});
8 $res \leftarrow$ calculateCloudResolution(pc_{modelN});
9 $pc_{issModel} \leftarrow$ detectISSKeypoints(pc_{modelN}, res);
10 $descriptorCache_{model} \leftarrow$ performCSHOTDescription($pc_{issModel}, res$);
11 **return**;
12 $pc_{envN} \leftarrow$ estimateAndAddNormals(pc_{env});
13 $res \leftarrow$ calculateCloudResolution(pc_{envN});
14 $pc_{issEnv} \leftarrow$ detectISSKeypoints(pc_{envN}, res);
15 $descriptorCache_{env} \leftarrow$ performCSHOTDescription(pc_{issEnv}, res);
16 **return** $descriptorCache_{model}$, $descriptorCache_{env}$;

12 – 15), since the recorded environment changes by the movement of the IPS, but the object model remains the same for an IPS session. In the processing pipeline, first surface normals are estimated (Section 3.3.2) and added to the respective point cloud. A search radius parameter $r = 3 \cdot leafSize(pc_{env}) = 0.03$ m is chosen to preserve fine details, as depicted in Section 3.3.2. The viewpoint is consistently set to the origin of the local coordinate system to counteract the multiple viewpoint problem.
Having the normals, the ISS keypoint detection follows. Its parameters are set dependent on the resolution of the respective point cloud. The resolution of a point cloud pc is calculated as:

$$res(pc) = \frac{1}{\#(pc)} \sum_{i=1}^{\#(pc)} |pc(i) - knn(pc(i))|, \tag{4.2}$$

where $pc(i)$ returns the ith point of the iterable pc, $knn(\cdot)$ denotes the nearest neighbor of a point determined by the k-nearest neighbor algorithm and $|\cdot|$ the Euclidean distance. Resolution-depending parameters ensure keypoints in sparse point cloud areas, e.g. if no points can be created due to bad illumination, and prevent too many keypoints in dense areas – especially a resolution dependent non-maximum suppression is used. A disadvantage is the additional computation time, since the resolution is recalculated for each new point cloud.
Subsequently, the detected ISS keypoints are passed as a point cloud to the CSHOT description. The only configurable parameter is the search radius, similar to the one

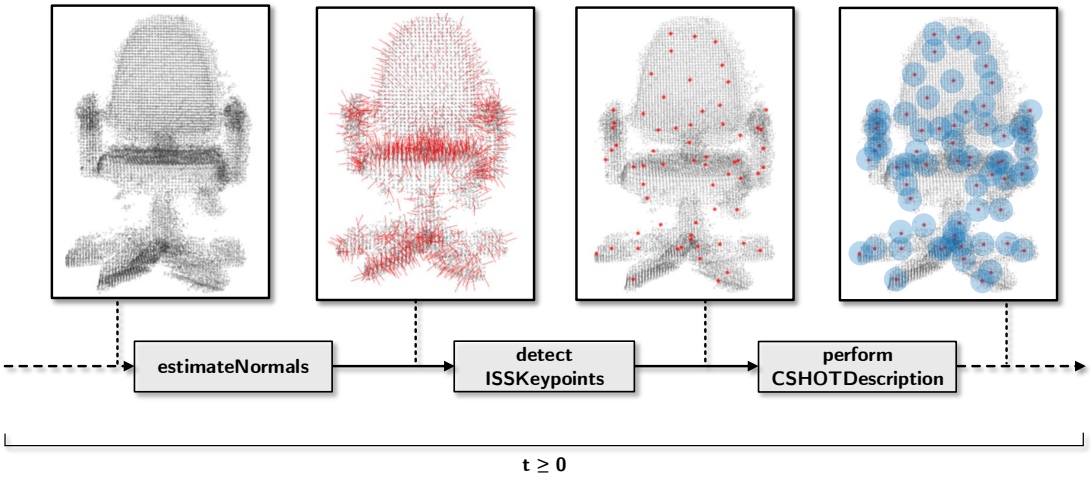


Figure 4.4: Visualization of the CODAPT feature detection and description.

of the ISS keypoint algorithm and set to the same value. The resulting environment CSHOT descriptors are cached as $descriptorCache_{env}$ in form of a mapping, where the respective ISS keypoint map to the corresponding descriptor. This treatment avoids the recalculation of descriptors for each particle in the KLD particle filter, described in the next Section, and therefore saves computation time in exchange for memory space. This mapping is not necessary for $descriptorCache_{model}$, since always all descriptors are used.
The functions for the normal estimation, the ISS keypoint detection and the CSHOT feature description are given as independent modules by the PCL framework.

4.2.4 KLD Particle Filtering

This section follows the Algorithm 5, illustrated in the Figure 4.5. The algorithm takes the output of Algorithm 4 as input, together with the initial pc_{model}. The needed input parameters for this group are listed in Appendix B.2. In the following, a particle is considered as a data container, which consists of values for its weight and the 6DoF. The 6DoF describe translation and rotation of the tracking object center of mass in relation to the origin of the IPS coordinate system and are referred to as particle hypothesis.
At $t = 0$, the KLD PF initializes the particles given the starting position of the object model to track. Following the object tracking benchmark of (Wu et al., 2013), a random pose noise is applied to each particle, to counteract the strong starting pose bias of the tracking. A particle's random shift is calculated as in Equation 4.3, where *obj* denotes the object to track. Each dimension is considered independently to avoid dependencies among one another. For the respective translational shift, a number between zero and ten percent of the object extend in the respective dimension is generated. The object size involvement makes the generated shift amount invariant to the object scale and the environment scale, respectively. For the rotational shift,

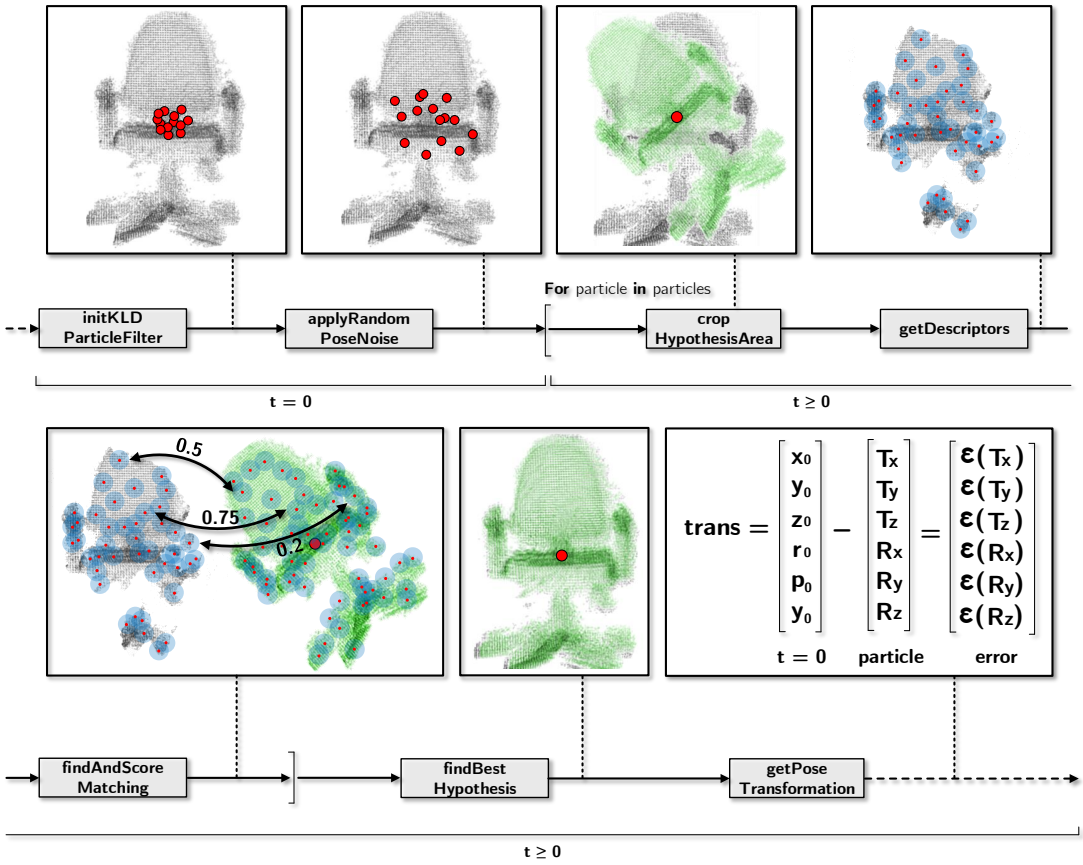


Figure 4.5: Visualization of the CODAPT KLD particle filtering.

a number between zero and ten percent of a full 2π rotation is generated. The random numbers are drawn from a uniform distribution to avoid prior knowledge of the tracking object pose. The calculated shifts are then added to the respective particle hypothesis (Lines 7 – 10).
For $t > 0$, first the already known pc_{issEnv} is gathered from the environment descriptor cache. Following, the algorithm processes each particle in the Lines 12 – 22. For each particle, first its hypothesis area is intersected with the environment point cloud pc_{issEnv} for the further analyses. The more precise the intersection, the better the particle's hypothesis and therefore also its score. Same hypothesis areas would lead to similar scores, which will be explained later. The hypothesis area is gathered by applying the particle hypothesis to the initial tracking object. For the intersection cloud, two methods are implemented: an approximative faster one and a more precise slower one. The former describes the hypothesis area by a 3D bounding box, whereby the rotations remain preserved. This procedure allows a faster intersection for cuboid-like object shapes and for applications, where the object orientation is of secondary importance. But this method is disadvantageous for the complement use cases, because a cuboid representation drops object shape details and the intersection of a cuboid with the environment is the same, as when the cuboid was rotated by $k \cdot \pi$ along any coordinate axis, for a $k \in \mathbb{N}_{>0}$. Both rea-

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta roll \\ \Delta pitch \\ \Delta yaw \end{bmatrix} = \begin{bmatrix} \text{rand}(\{-1,1\}) \\ \text{rand}(\{-1,1\}) \\ \text{rand}(\{-1,1\}) \\ \text{rand}(\{-1,1\}) \\ \text{rand}(\{-1,1\}) \\ \text{rand}(\{-1,1\}) \end{bmatrix} \cdot \begin{bmatrix} |\max_x(obj) - \min_x(obj)| \cdot \text{rand}([0,0.1]) \\ |\max_y(obj) - \min_y(obj)| \cdot \text{rand}([0,0.1]) \\ |\max_z(obj) - \min_z(obj)| \cdot \text{rand}([0,0.1]) \\ \text{rand}([0, \frac{2\pi}{10}]) \\ \text{rand}([0, \frac{2\pi}{10}]) \\ \text{rand}([0, \frac{2\pi}{10}]) \end{bmatrix} \quad (4.3)$$

sons lower precision in the intersection and will later lead to a worse matching. The latter intersection method is the already mentioned intersection of the convex hulls. This method preserves the object shape details and therefore is expected to lead to more precise intersections, but at the expense of computation time. The latter approach is visualized in Figure 4.5. At this point, the cached CSHOT descriptors are obtained from the $descriptorCache_{env}$ for the intersection result $pc_{issEnvCrop}$.
In the next step, the best corresponding object model descriptor is searched for each CSHOT descriptor of the cropped environment and their matching scored (Lines 16 – 22). Since a CSHOT descriptor is represented as a vector, the best corresponding match can be defined as nearest neighbor in the Euclidean vector space and their distance to each other as matching score, scaled between zero and one. A score of zero means perfect matching, while a score of one means no match. All scores above a threshold $\tau_{matchingScore}$ of 0.5 are dropped to prevent results, where the tracking fails or where the tracking score is too low to be considered as tracking match. The remaining scores are accumulated and its inverse plus a little positive number (to avoid a division by zero here) set as the weight of the corresponding particle, to account for the inverse relationship between the weights and the CSHOT matching scores (Line 22). This manual weighting corresponds to the correction phase of the PF framework. The PF framework performs the selection and prediction steps automatically before each further particles iteration. In these phases, zero-meaned Gaussion noise is added according to Equation 3.10.
After reviewing the scoring process, the importance of a precise intersection method in Line 13 clears up: The more matching descriptors the model-environment-intersection brings up, the higher the particle's weight. But the best particle weight have to be unique, since a unique tracking result is expected. Non-unique best particle hypotheses emerge, when multiple particles lead to the same subset of environment descriptors after the intersection or in the case, that another subset of environment descriptors lead to the same score, which means that the local surface structure (surface normals) and the local colors are very similar to the ones of the tracking object for multiple surface keypoints. The first case appears in the addressed approximative intersection procedure. The second case arises especially for common shapes and colors[2] or for objects of the same object class in general. To counteract this pitfall, the data association step was introduced and the particle filter framework chosen, which applies its movement prediction to stay on the tracked object.

[2]For instance cuboid-forms and grayish coloring in urban areas, when CODAPT would be applied to an optical satellite.

Despite that, a residual risk for ambiguous results remains. In this case, one of the best particles is chosen randomly.
After iterating all particles, the particle with the highest weight is considered as the best object tracking hypothesis and its 6DoF parameters describe the transformation (Lines 23 – 26). Since the tracking object is rigid and non-moving during the IPS session, the difference between the particle's pose transformation and the tracking object start position equals the tracking error for the respective time stamp (see last frame of Figure 4.5). The empty return in Line 27 is interpreted as no valid transformation and therefore as no tracking result for a time step.

Algorithm 5: CODAPT KLD particle filtering.

```
1  Input: descriptorCache_model - model CSHOT descriptor cache,
2         descriptorCache_env - environment CSHOT descriptor cache,
3         pc_model - initial model point cloud;
4  Parameters: (Appendix B.2);
5  Output: trans - transformation of the tracking object;
6
7  if t == 0 then
8      particles ← initKLDParticleFilter(pc_model);
9      particles ← applyRandomPoseNoise(particles);
10     return;
11 pc_issEnv ← getKeys(descriptorCache_env);
12 for particle in particles do
13     pc_issEnvCrop ← cropHypothesisArea(pc_issEnv, particle, isApproximated);
14     descriptors_envP ← getDescriptors(descriptorCache_env, pc_issEnvCrop);
15     score ← 0.0, matchFound ← false;
16     for descriptor_envP in descriptors_envP do
17         descriptor_model ← findNearestNeighbor(descriptorCache_model);
18         tempScore ← scoreMatching(descriptor_envP, descriptor_model);
19         if tempScore ≤ τ_matchingScore then
20             score ← score + tempScore;
21             matchFound ← true;
22     particle.weight ← 1.0 / (score+10^-5);
23 if matchFound then
24     particle ← findBestHypothesis(particles);
25     trans ← getPoseTransformation(particle);
26     return trans;
27 return;
```

Evaluation - Design

This chapter proposes the approach to evaluate the accuracy and runtime performances of CODAPT, with respect to the uncombined single methods. The sections follow the setup of the Figure 5.1. First, an overview and design decisions for the evaluation are introduced. Following, the data collection for the evaluation is explained at first for the real-world datasets and subsequently for the simulated datasets, including the used simulation software. Finally, the used metrics, to make the performance related research questions operationalizable and to gather statistics, are presented and justified.

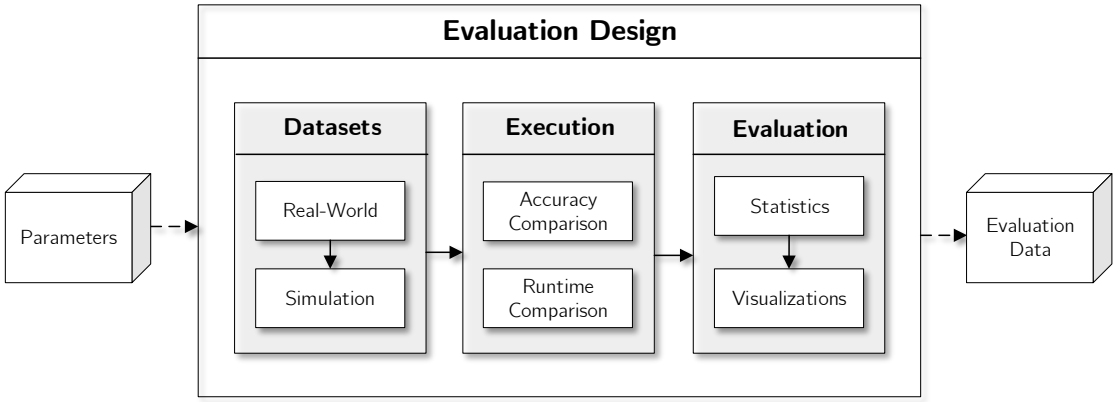


Figure 5.1: Evaluation setup visualization. Arrows denote data dependencies between components, where the arrow's targets depend on its sources.

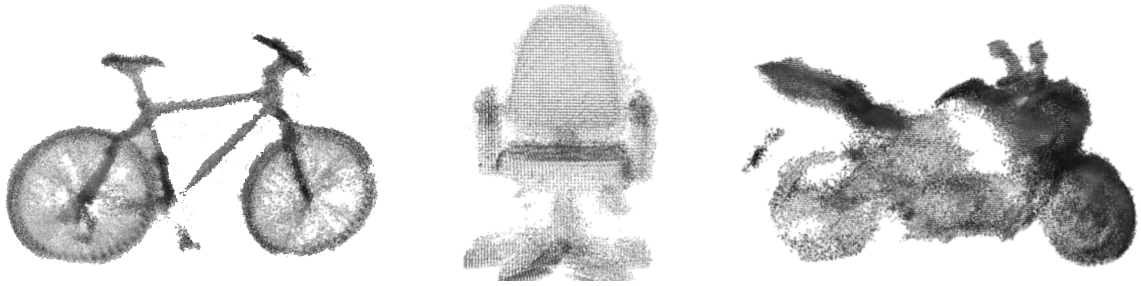
5.1 General Approach

This section first comprises necessary preparations to perform the evaluation. Subsequently, specifics for the evaluation execution are discussed.

5.1.1 Preparation

The research questions target to determine the accuracy and runtime performance of CODAPT for the 3D object tracking and the 2D object detection tasks, in comparison to the respective single uncombined methods. To determine the quality of the results of CODAPT, ground truth values are needed. Since the tracking objects are assumed to be rigid and non-moving, the true pose of the tracking object can be derived from the IPS trajectory, knowing the object model and its starting pose. The object model and its starting pose are determined as follows: Right before an evaluation dataset is recorded, an independent IPS session is performed with the same start- and end position, without moving the objects. The independent session

makes sure, that the object can be recorded from each side to gather a complete model and that the point clouds differ from the ones in other sessions, to avoid model overfitting. Identical start- and end points and also the non-moved objects lead to an identical local IPS coordinate system and unchanged object poses within the sessions. For each desired tracking object, its respective point clouds are merged, until a complete object is obtained. Next, the object is cut out and saved as a separate point cloud. The coordinates of the points of this point cloud describe the pose in the mentioned local IPS coordinate system. This procedure has to be repeated before each IPS session.



(a) Bicycle. (b) Chair. (c) Motorbike.

Figure 5.2: Exemplary tracking object point clouds.

Having the model point clouds with their poses, the next step is to get a ground truth IPS trajectory for an evaluation IPS session, to determine ground truth object poses and therefore ground truth values for the 3D object tracking. In Section 4.2.1 it is anticipated, that the point clouds processed by the IPS are transformed into the local IPS coordinate system, using the IPS trajectory. A further advantage resulting from this is, that the determination of the ground truth trajectories for each tracking object can be simplified. However, the ground truth trajectories of the IPS needs to be determined. The missing means to determine approximately ground truth values (e.g. by multiple lasers to measure pose changes accurately over time or by constantly visible, pre-located tags) limit the evaluation. More precisely, the proposed simplification idea holds only for a trajectory, which fulfills the following condition:

$$\epsilon(traj(n)) \leq \delta_{pc}, \tag{5.1}$$

where $\epsilon(\cdot)$ denotes the error of a pose change estimation, $traj(n)$ the trajectory for the last time stamp n of an IPS session and δ_{pc} the resolution of an arbitrary point cloud pc. Since the error accumulates over time, $traj(n)$ contains the whole pose estimation error. To approach the constraint stated in Inequality 5.1, aids are applied, which are proposed in Section 5.2.1. The remaining error is discussed and considered in the discussion of this work. Assuming an nearly optimal trajectory

from now on, the tracked object stays on a fixed pose in the local IPS coordinate system for the whole IPS session. This means that the tracking method has to keep the focus on the fixed object pose. The advantage is that the tracking pose error can be measured directly, since the initial object pose is assumed to be the ground truth pose. The disadvantage is the neglection to exploit particle movement capabilities of the PF between two point cloud frames. The tracking task remains valid though, because the PF has no capability to learn or remember the last tracked object pose. Implicitly, the probability for sampling particles in a time step $t + 1$ is increased in this area, but this holds for each possible tracking match. The PF can still be distracted by noise, similar objects, background clutter and others. In addition, this evaluation simplification gives the isolated tracking and CODAPT the same advantage.
What remains are the ground truth values for the 2D object detections and the datasets. The former are determined manually with the tool LabelImg (Tzutalin, 2015) for each image frame of an IPS session. The latter is explained in Section 5.2.

5.1.2 Execution

Having all the addressed data prepared, CODAPT can be executed, respecting some considerations.
First, there are two options to run CODAPT: As an IPS feeder[1] or as standalone application. The first one gets the data at runtime from the IPS, the last one loads the data from a data storage. For the evaluation, the standalone approach is chosen. This allows an unbiased runtime measurement, since all hardware resources can be used dedicatedly by the evaluation procedure.
The parameter container in Figure 5.1 contains the following evaluation parameter groups:

- *Filepath.* Paths to the recorded IPS sessions and the tracking object models.
- *Method.* Comprises a flag to toggle between CODAPT and the the uncombined single methods.
- *Runtime.* Variable parameters to decide the quality of the PF: maximum number of applied particles, number of iterations per tracking frame and the chosen PF hypothesis cropping method.
- *Ground truth.* Ground truth 2D object detections (positions and object classes) for each image frame.

[1] During this work, the CODAPT software is also integrated as a feeder into the IPS software framework. The denotation IPS feeder refers to a single component of the IPS software framework.

Since the stated runtime parameters exert a major influence on the accuracy and the runtime, six parameter setups are proposed for the evaluation in Table 5.1. These configurations are meant to cover combinations, from where real-time performance can still be expected (*Fast*) to parameters, where an approximately accurate result is expected (*Accurate*). To evaluate the PF cropping method, this parameter is introduced as an independent one and denoted with a suffix (*-C*) for the convex hull cropping. Non-varied parameters are kept constant over the whole evaluation, to provide comparability between the results.

Table 5.1: Six different parameter setups for the evaluation.

Label	**Parameters**	**Label**	**Parameters**
Fast	• 200 maximum particles • 1 PF iteration per frame • 3D box PF hypothesis crop	Fast-C	• 200 maximum particles • 1 PF iteration per frame • Convex hull PF hypothesis crop
Default	• 500 maximum particles • 1 PF iteration per frame • 3D box PF hypothesis crop	Default-C	• 500 maximum particles • 1 PF iteration per frame • Convex hull PF hypothesis crop
Accurate	• 1000 maximum particles • 2 PF iterations per frame • 3D box PF hypothesis crop	Accurate-C	• 1000 maximum particles • 2 PF iterations per frame • Convex hull PF hypothesis crop

As mentioned in Section 4.2.4, the initial tracking object starting pose is randomly shifted along each coordinate axis independently, to reduce the strong bias of the predefined object pose.
The evaluation is performed for each dataset and each tracking object twice – first with CODAPT and subsequently with the single uncombined methods. As an outcome for each experiment, five types of raw data are recorded. At first runtime measurements. The second recorded data is the tracked 3D pose deviation to the starting position pose for each tracking frame, i.e. the pose tracking error. Thirdly, raw YOLO object detections are saved. Fourthly, YOLO object detections are logged, where the CODAPT data association is successfully performed. Finally, the tracked object pose is back-projected onto the image plane of the IPS and saved as 2D object detection of the object tracker and CODAPT, respectively.

5.2 Datasets

To compare the performances between CODAPT and the single uncombined methods, several datasets are proposed in the next two sections. Since the IPS is used as data source and as part of the evaluation, own IPS-specific datasets are recorded.
The design considerations rely on (Wu et al., 2013). They state that "for better evaluation and analysis of the strength and weakness of tracking approaches, we propose to categorize the sequences by annotating them with the 11 attributes [...]". This attribute annotation operationalizes the properties and their variability of a dataset, regarding an object tracking task. All datasets aim to vary the attribute manifestations, to reduce their effects on the evaluation results. Since not all attributes fit to 3D object tracking scenarios, a subset is selected and adopted (see Table A.1).
In addition to the dataset attributes, different environments for the datasets are considered to cover potential fields of application for CODAPT. The proposed ones are *indoor* and *outdoor* urban areas and *wasteland*. The first class represents areas within buildings and mainly contain office objects. The second one is supposed to provide vegetation background clutter and vehicle objects. The third class comprises background clutter by soil and stones and is assumed to represent desert-like areas on earth and in the outer space.
Last but not least, the object classes for tracking are chosen with respect to YOLO's detectability and object attribute variability. The proposed object attributes are *size*, *color*, *textureness* and *curviness* – their definition and a quantization can be found in Table A.2. The proposed quantification is chosen to divide the YOLOv3 objects in roughly similar sized classes. Moreover, these properties are assumed to influence the CODAPT procedure, since the first one corresponds to the amount of found feature points and the latter to the CSHOT feature description.
The Figure 5.3 illustrates and exemplary dataset. The applied symbolism is explained in Table A.3. Assuming an already calibrated and initialized IPS, the IPS session starts at $t = 0$ on the starting point. While $0 < t < n$, the predefined path is walked with the IPS in the hand, where the IPS cameras capture the environment and all objects are visible as often as possible, to provide valid tracking frames. The objects are viewed from different camera poses during an IPS session. At $t = n$, the IPS session ends and the IPS is parked at the starting point with the same pose, to determine a close loop error and estimate the influence on the evaluation results. Objects of same object classes are placed next to each other in each dataset, since it is expected, that the proximity challenges the tracking procedure the most. The actual object of an object class to track is chosen randomly.

5.2.1 Real-World

Three real-world datasets are proposed in this work. They are illustrated and specified in the Appendix A.2. The distances mentioned in the illustrations are gauged

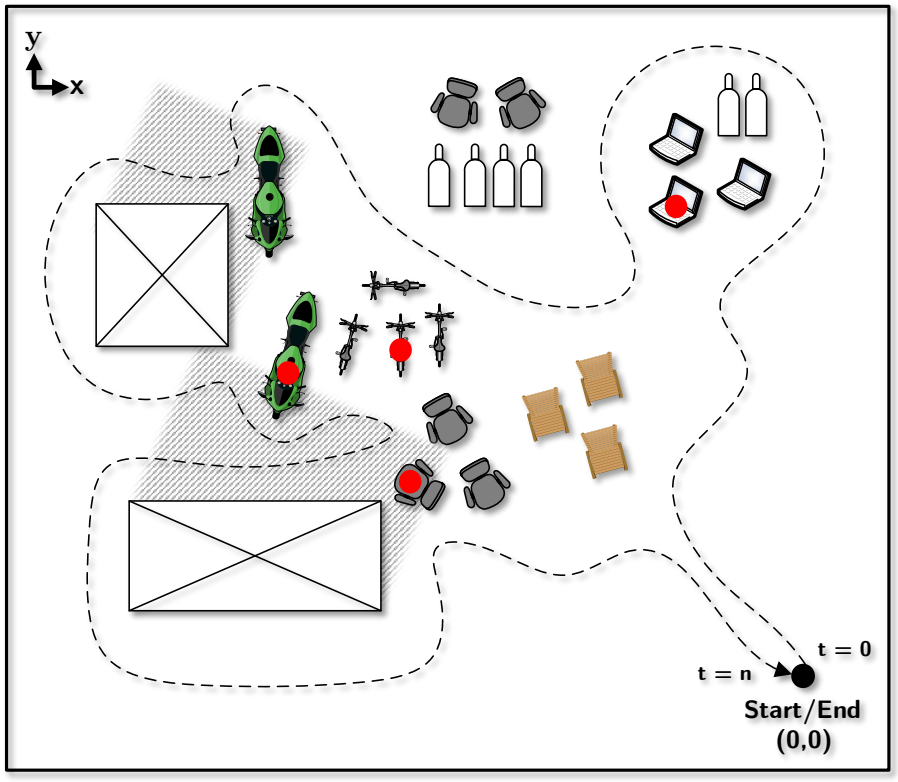


Figure 5.3: 2D bird's-eye view of an exemplary IPS object tracking session.

using a measuring tape and the respective IPS starting pose are marked to ensure, that IPS start- and end-pose equal.
Due to the error introduced by the IPS methods and the nature of imperfect measurement equipment, confounding variables are introduced for the evaluation, which have to be minimized and discussed. The Table 5.2 summarizes these error variables. The *image noise* and the error introduced by imperfect *IPS calibrations* are naturally and hardly optimizable here. The SGM error is by design to allow a real-time capable stereo matching process and can also be barely influenced or measured. The quality and the quantity of the YOLO object detections are already considered in CODAPT (data association) and can be identified during the evaluation. To tackle influencing environmental conditions, the mentioned dataset and object class attributes are introduced and the environment scenarios varied. To support the evaluation simplification assumption of a nearly ideal IPS trajectory and to reduce this error, two aids are applied. The first one is to keep the overall traveled distance of the IPS session small. The proposed real-world datasets comprise an average distance of about 46 m per IPS session. Following the announced IPS close loop error of 0.5% with respect to the overall traveled distance from (Grießbach et al., 2012), an error of 0.23 m can be expected. A closed-loop error is defined as $\epsilon(traj(n))$.

Table 5.2: Confounding variables for the real-world CODAPT evaluation datasets.

Variable	Description
Image Noise	Although a loss-less data compression of the proposed TIFF file format is used, the CCD elements of the cameras and the image post-processing introduce image noise.
YOLO	All object detections with a probability below a pre-defined threshold are neglected. Additionally, the calculated bounding boxes can be shifted or misfit in general.
IPS Calibrations	The calibrations between the two camera sensors and the IMU of the IPS are assumed to be naturally not perfect.
SGM	Due to discrete and limited optimization paths, a post-processing filter and regions where pixels cannot be matched, information get lost.
Trajectory	The real-world trajectory of the IPS contains errors introduced by inaccuracies of the VIO.
Environment Conditions	Different light conditions (for CSHOT), textureness (for CSHOT and VIO), context information (for YOLO) and similar tracking object classes can distort the evaluation results.

The second aid is to add predefined checkpoints with known position. When the IPS pose equals for each sampling point (when taking the sampling point transformations to each other into account), the trajectory between a pair of sampling points can be straightened and the closed-loop error between these points set to zero. The procedure is referred to as trajectory smoothing and is explained in the Appendix A.3 in detail. Additionally, simulated datasets are proposed to tackle the presented error sources.
All real-world datasets comprise 6 IPS sessions with an overall tracking time of about 14 minutes, 278.7 walked meters and 14 object trackings. The Figure shows two exemplary images of the proposed real-world datasets, captured by the left camera.

5.2.2 Simulation

Based on the real-world datasets, three simulated datasets are proposed here and specified in the Appendix A.1. The simulated datasets are meant to support the real-world ones, since real-world datasets are time consuming to create, while the simulated ones can be put together in a simulation software. They also allow a fast environmental parameter variation and provide ground truth data. The simulation software used for this work bases on a customized version of the open-source software OpenSceneGraph (Burns, 1998–2019). The software facilitates the creation of virtual 3D environments out of object meshes and textures. It also provides capabilities to specify camera sensors, camera noise effects, camera movements and light variations. The software is provided by the DLR (Irmisch et al., 2019).

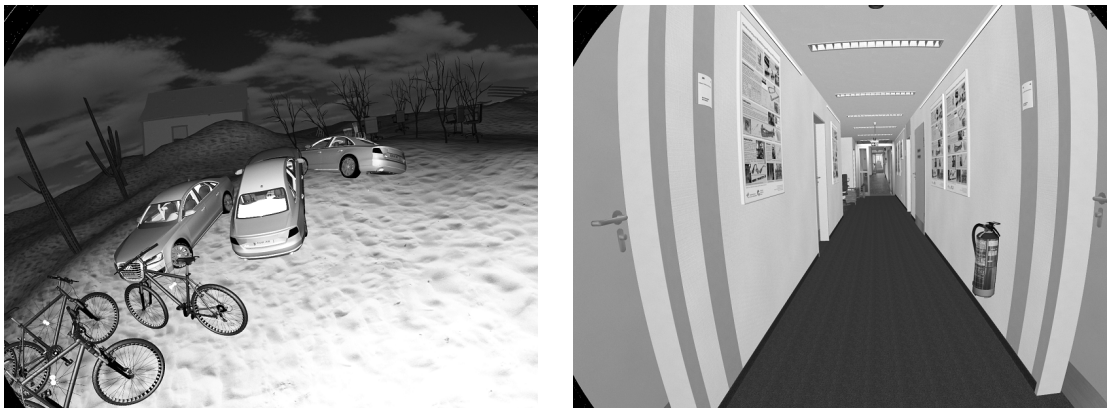
(a) (b)

Figure 5.4: Exemplary tracking object image recordings of the proposed real-world datasets (increased brightness for better visualization).

The simulation datasets tackle the confounding variables of the real-world datasets stated in Table 5.2. Each of the datasets utilizes a real-world trajectory of the IPS, which is recorded independently in each case. Since the true camera movement is known in the simulation, a ground truth trajectory can be employed. To counteract the influence of environmental conditions, different pre-build object models and environment scenarios are selected. In addition, each simulated dataset is build twice: with default illumination and darkened, whereby the IPS has a light attached on its front. The IPS calibrations can be predefined beforehand and applied to the simulation without any alterations. The image noise, YOLO and SGM remain untouched, to prevent unrealistic conditions. Nevertheless, simulated 3D environments lack of realism and therefore might worsen results, which is still an open research problem. All simulated datasets comprise 6 IPS sessions with an overall tracking time of about 25 minutes, 1566 walked meters and 52 object trackings. The Figure 5.5 (a) and (b) shows two exemplary images of the proposed real-world datasets captured by the left camera.

5.3 Metrics

In this section, metrics are proposed that are used to operationalize the accuracy and runtime performances for 2D object detections and 3D object tracking, respectively. Besides, the introduced randomness of the proposed method in Chapter 4 is considered, as well. The selected metrics for the 2D object detections base on the work of (Wu et al., 2013). They are also adopted to the 3D object tracking evaluation.



(a) (b)

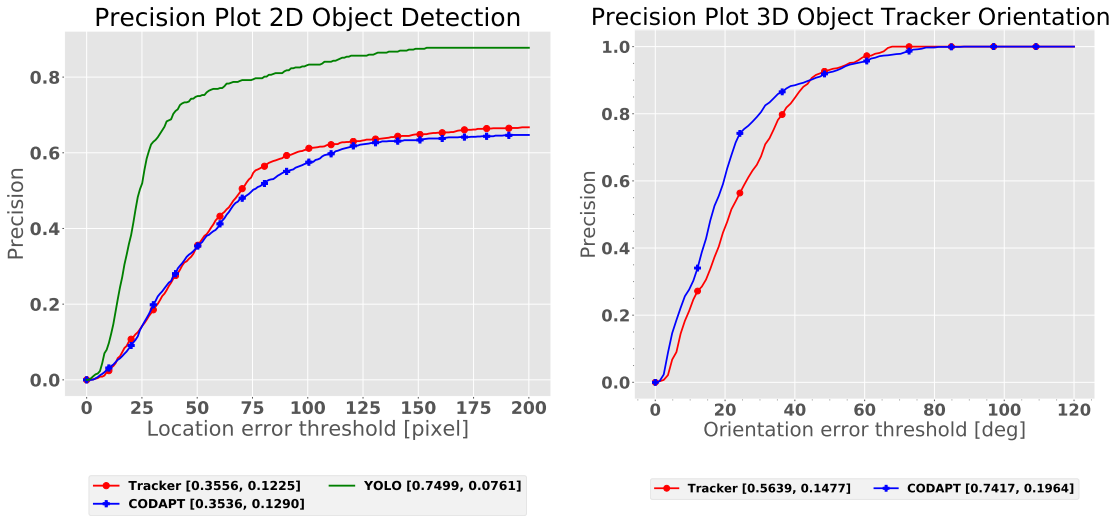
Figure 5.5: Exemplary tracking object image recordings of the proposed simulation datasets (increased brightness for better visualization).

Precision Plot. A widely applied metric for precision is the center location error, that is defined as the average Euclidean distance between the center locations of the tracked targets and the manually labeled ground truth values. The average center location error over all frames of an IPS session is then used to summarize the overall precision error of a session. One issue for this procedure occurs, when the target is lost and arbitrary areas are determined as detection or tracking result. As a result of that, the proposed object location can be random and the average error does not measure the precision performance reliably. The proposed fix is to use the percentage of frames, whose estimated object location lies within a predefined threshold distance of the ground truth object location (true positive). Otherwise a frame can be seen as false positive, here. This ratio also fits to the precision score used for statistical classification tasks:

$$Precision = \frac{\#(TruePositives)}{\#(TruePositives) + \#(FalsePositives)} \tag{5.2}$$

To measure the precision of a 2D object detection, the bounding box center of mass is utilized. For a 3D object tracking result, translational and orientational precision errors are examined in separate plots, whereby the respective absolute errors are provided by the proposed evaluation method directly. Unless otherwise stated, a single graph represents the mean scores over all tracked objects and all prepared scenarios. Each graph is also summarized by a score and the standard deviation (STD) at a predefined x-axis-threshold. For the 2D cases, a threshold of 50 pixels is chosen, for the 3D translational cases 10 cm and for the 3D orientational ones a value of 20 deg (based on (Wu et al., 2013)).

The Figure 5.6 illustrate exemplary precision plots. The respective axis of abscissas denotes threshold values for the used unit and the related ordinate value represents the number of frames, with respect to all number of frames of an IPS session, that yield results with lower or equal precision than this threshold value. The higher the printed curve for low threshold values, the better. In the legend, the brackets embrace the summation precision value and its standard deviation for the chosen threshold. The higher the first score and the lower the second one, the better.



(a) 2D object detection center offset.

(b) 3D object tracking orientation error.

Figure 5.6: Exemplary precision plots.

Success Plot. In addition to the precision metric, the area overlap (Intersection over Union - IoU) is another metric used here. Given the area A_{obj} of the tracked object and the ground truth area A_{gt}, the IoU score is defined as

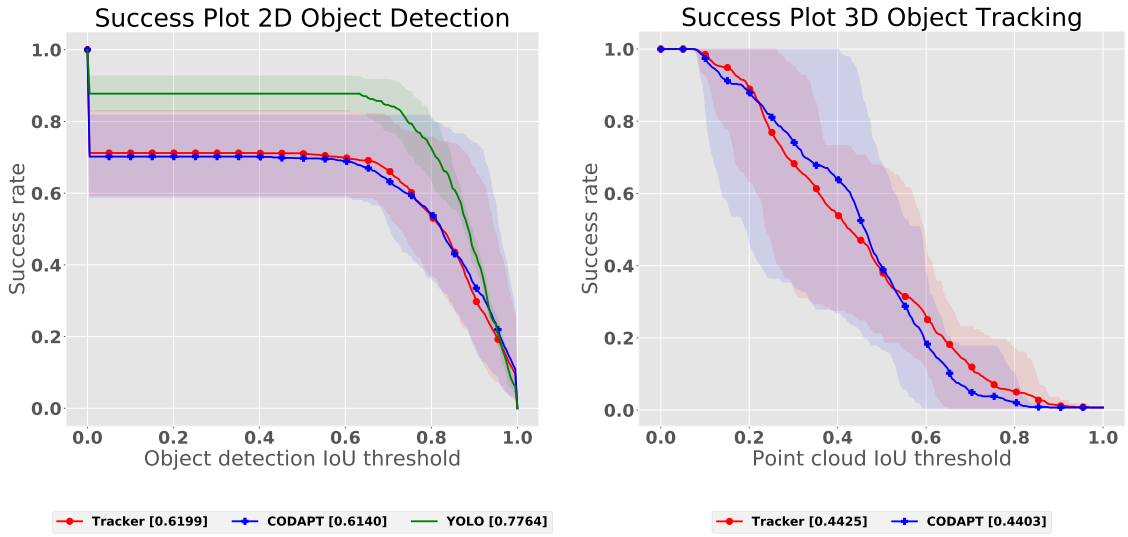
$$IoU(A_{obj}, A_{gt}) := \frac{A_{obj} \cap A_{gt}}{A_{obj} \cup A_{gt}}, \tag{5.3}$$

where $\cap$ and $\cup$ denote the intersection and union of two areas, respectively. In the 2D case, the object area is provided as a bounding box, in the 3D case as the convex hull of the tracking object point cloud. To aggregate the performance for a whole IPS session, the number of frames are counted, that fulfill the threshold IoU score at an x-coordinate (similar to the precision plot). The success plot then illustrates the ratios of successful frames at the IoU-thresholds, varying from 0 to 1. Analogous to the precision plot, a single graph represents the mean scores over all tracked objects and all prepared scenarios by default. In addition, for each graph at each point, the minimum and maximum success rates are provided, to describe a boundary for the performance (only for plots with $\leq$3 graphs). These information are meant to extend the standard deviation of the precision plots. The miss rate is legible at $x = 0 + \epsilon$ for a small ϵ, which refers to the next data point.

In contrast to the precision plot, the Area-Under-Curve (AUC) score summarizes each success plot. The AUC score is defined as the area under a plotted curve $f(x)$:

$$AUC(x) \coloneqq \int_{0.0}^{1.0} f(x)\ dx. \tag{5.4}$$

The Figure 5.7 illustrates exemplary success plots. The higher the success ratios for low thresholds, the better. The shaded areas correspond to the graph of the same color and illustrate the extend between minimum of maximum at each x-coordinate. In the legend, the brackets embrace the summation AUC scores. The higher the AUC score, the better.



(a) 2D object detection IoU error.

(b) 3D object tracking IoU error.

Figure 5.7: Exemplary success plots.

Runtime Plot. The runtime plot illustrates the measured execution times to run the code of CODAPT, in comparison to the ones of the single 3D tracking task. The on the axis of abscissas plotted CODAPT components are preselected: Those running once per IPS session (for example 3D tracking model initialization) or that do not contribute to the overall runtime (for instance descriptor cache access) are neglected. The axis of ordinates represents the measured runtimes in seconds per tracking frame or per frame and particle, by default. The latter normalization concerns only the components *PF Hypothesis Area Crop* and *Particle Scoring-Matching*, relying on the number of used particles. For the normalization, the average number of actually used particles is utilized. Frames, where no tracking can be performed (e.g. due to missing point cloud fragments or when the tracking object is out of view), are filtered out, to obtain comparable runtime values. The total runtimes per frame are summarized in the last bar group.
The bars represent the mean execution time of a component, with respect to a whole IPS session. The black strokes indents the minimum and the maximum values, respectively. The Figure 5.7 illustrates an exemplary runtime plot.

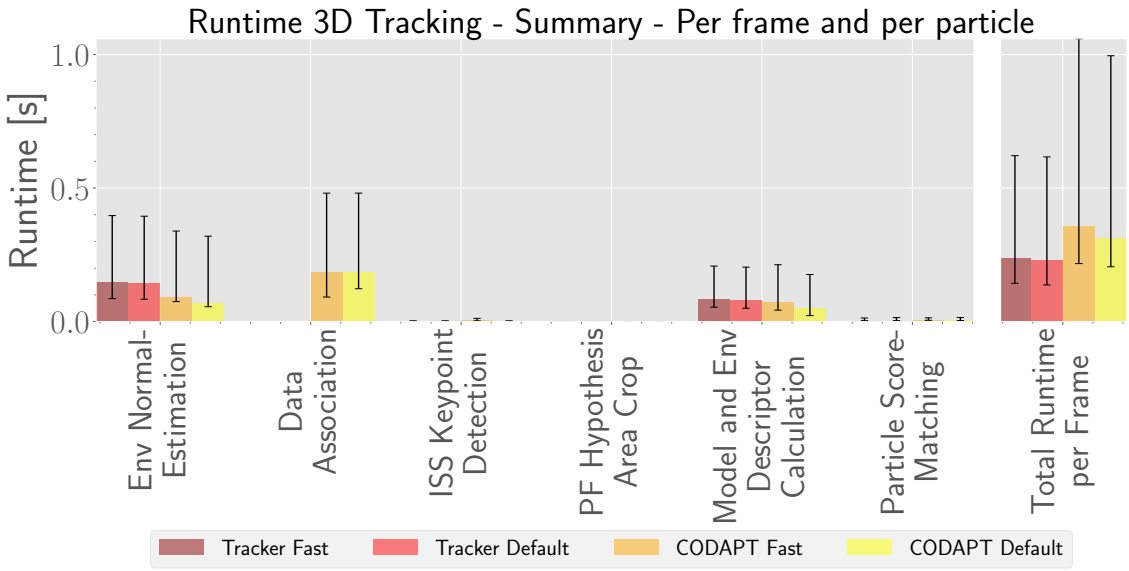
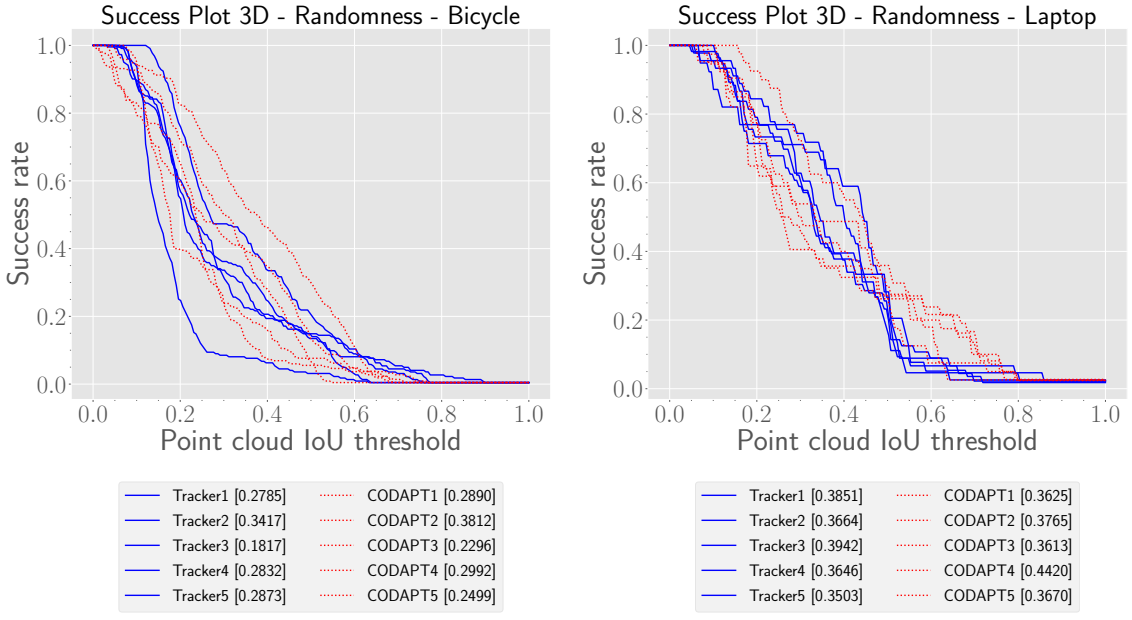


Figure 5.8: Exemplary runtime plot.

Randomness Plot. The randomness plot is used to reveal the influence of the non-deterministic parts of the proposed method (especially the PF) on the evaluation results. The goal is to indicate, whether the used method leads to stable evaluation results or not. Therefore, the STDs of the plotted samples are calculated.
To create a plot, a single evaluation configuration is executed five times and analyzed regarding a chosen metric.
The Figure 5.9 depicts exemplary randomness plots for 3D success.



(a) (b)

Figure 5.9: Exemplary randomness plots indicating a high influence of randomness.

Evaluation - Results

In this section, the results of CODAPT running on the proposed datasets are evaluated and analyzed using the stated metrics. The structure follows the formulated research questions.
First, the outcomes are preanalyzed for the subsequent sections, to reduce the amount of data and as auxiliary to examine the Research Question (1). Following, the Research Question (2) is investigated. Therefore, the 3D accuracy and runtime performances of CODAPT are compared to the ones of the isolated 3D tracker. Finally, the 2D accuracy performances of the isolated trackers are compared to the one of the chosen 2D object detector YOLO.
For the sake of completeness, the Appendix C lists additional experiment results.

6.1 Preliminary Evaluation

The results are evaluated with respect to randomness, parameter setups, the data acquisition method and different object and environment attributes.
If not otherwise specified, the 3D success plot is mainly analyzed in the following, because its summarizing AUC score is more accurate than one threshold score of a precision plot (Wu et al., 2013).

6.1.1 Randomness

The proposed method involves an influence of randomness – especially through the PF and the manual object pose initialization noise (see Chapter 4). This randomness might overlay other performance impacts and therefore has to be quantified. The randomness is expected to influences the tracker and CODAPT equally. Since the method execution and the result calculation is time consuming for all proposed object trackings, four tracking setups are chosen, that differ the most in parameter setups, object and scene properties:

Table 6.1: Tracking setups for the randomness plots.

Label	Dataset Acquisition	Scene	Object	Parameter Setup	Details
(a)	Simulation	Desert	Bicycle	Default	No illumination changes
(b)	Simulation	Desert	Laptop	Default	No illumination changes
(c)	Real	Office	Chair	Fast-C	No illumination changes
(d)	Real	Office	Chair	Accurate	No illumination changes

Table 6.2 summarizes the main results of the Figures 6.1a – 6.1d and quantifies the randomness in terms of sample standard deviations (STD – see Equation 6.1).
A similar amount of randomness is visible for both methods. While the STD of the CODAPT AUC score is with 3.46% slightly lower than the trackers one with 4.52%, the difference of about 1% is considered as too small to be meaningful. Noticeable are some graphs with a steep descent in the beginning, like *Tracker3* in Figure 6.1a.

This indicates an early loss or at least a partly loss of the object track. This might be the main interference factor, that distort the randomness plots. Notwithstanding that, a considerable randomness influence of up to 6.19% AUC success score is present here. In other words, the randomness can influence up to 22% of the AUC success score and is therefore considered as high.

$$STD(x_1, ..., x_n) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}, \text{ with } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{6.1}$$

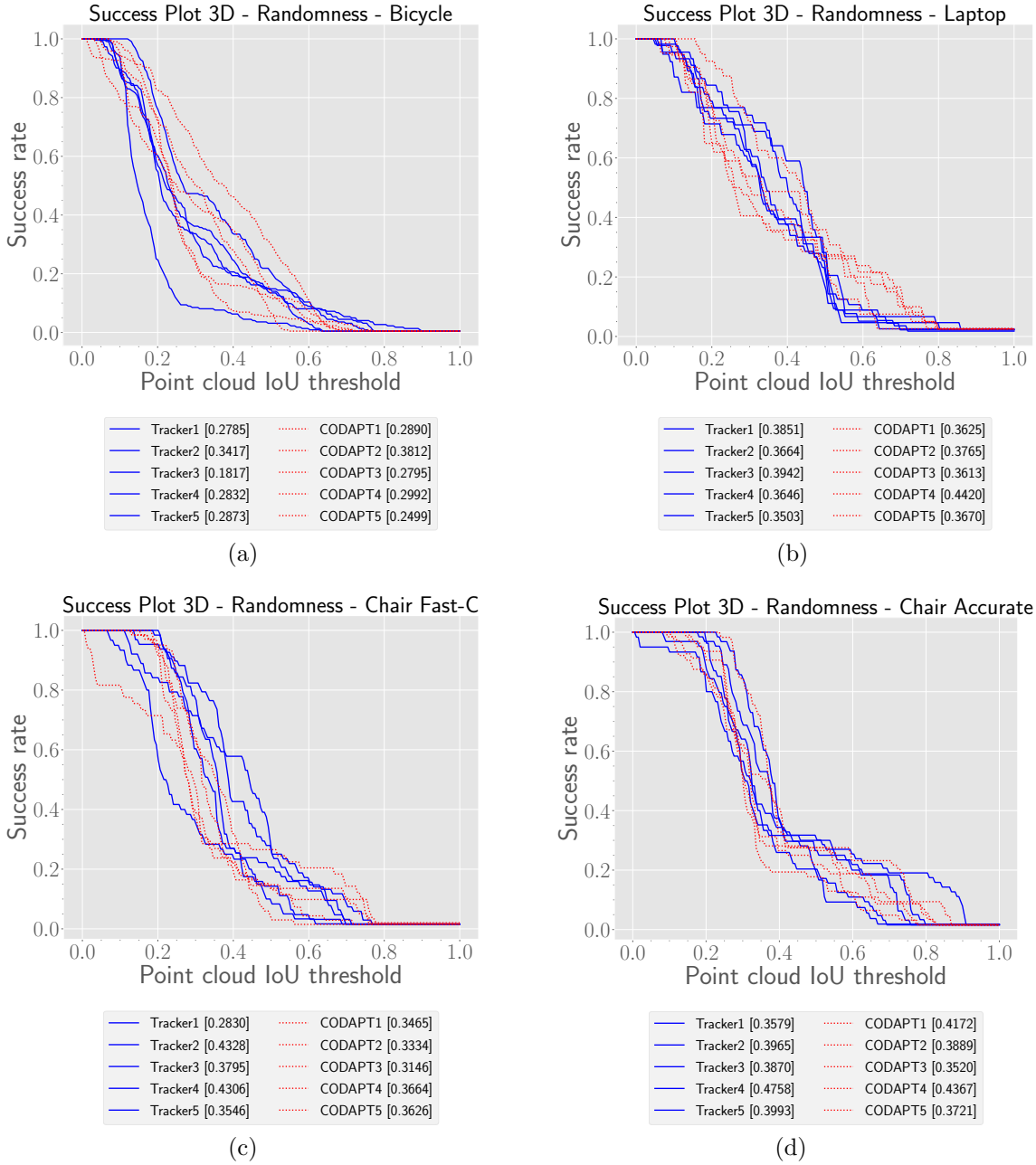


Figure 6.1: Influence of randomness on the tracking accuracy.

Table 6.2: Summary of randomness plot results.

Label	Tracker AUC STD [%]	CODAPT AUC STD [%]
(a)	5.78	4.91
(b)	1.75	3.41
(c)	6.19	2.13
(d)	4.37	3.40
Mean	4.52	3.46

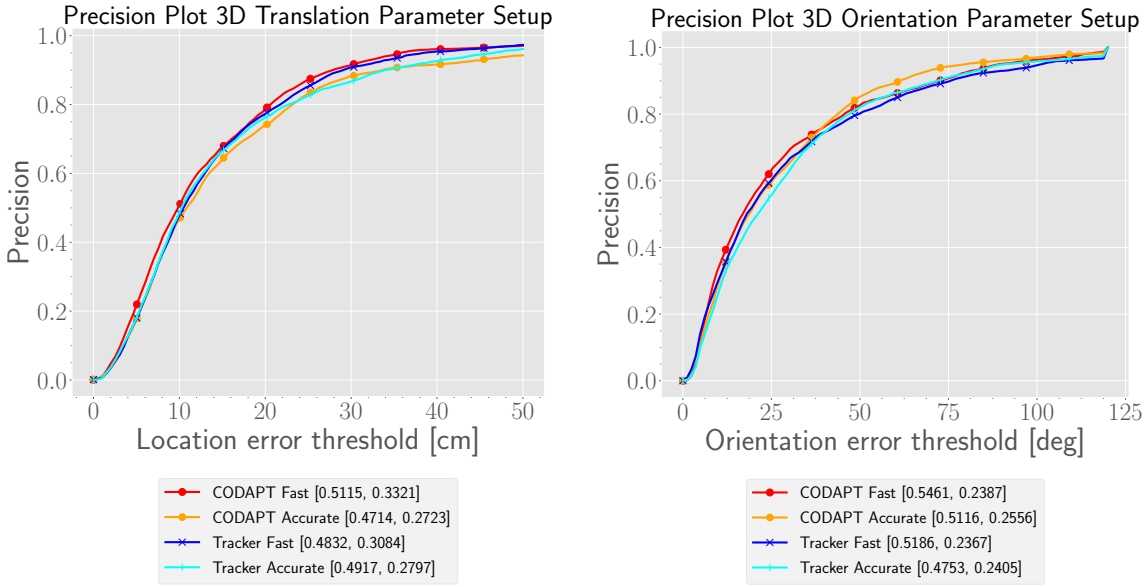
Figure 6.2: Exemplary tracking loss (result in blue) after 23.4 s of overall 123.4 s.[1]

6.1.2 Parameter Setup

In Section 5.1.2, six parameter setups are proposed: three setups with varying PF accuracy (*Fast*, *Default* and *Accurate*) and further three with a different cropping method. Here, the three parameter setups are examined with respect to their accuracy performances and the cropping methods are neglected for now. Therefore, results with corresponding parameter setups (e.g. *Fast* and *Fast-C*) are summarized as one graph. It is expected, that more particles and more PF iterations lead to more accuracy, since the tracking search space is sampled denser. 3D precision plots are employed to compare the precision performances.
In Figure 6.3a, the *Accurate* parameter setup leads to a lower mean accuracy and a lower STD for CODAPT (-4.01% and -5.98%) and a higher mean accuracy and a lower STD for the tracker (+0.85% and -2.87%), compared to the *Fast* parameter setup. The further trend indicates a slightly higher accuracy for the *Fast* parameter setups.
In contrast, the Figure 6.3b reveals a lower mean accuracy and a higher STD for CODAPT (-3.45% and +1.69%) and also for the tracker (-4.33% and +0.38%), when comparing the *Accurate* parameter setup to the *Fast* ones. In the later trend, the *Accurate* parameter setups increase the orientational accuracy.
Since the particles of the PF treat the translation and orientation as equal dimensions of the search space, the respective accuracies can also be treated similarly. These results are contrary to each other. Neither the translation nor the orientation accuracy results clearly indicate an accuracy gain for a specific parameter setup. Hence, an increase from 200 to 1000 particles and from one to two PF iterations per image does not lead to more accuracy for the tracking methods, here. This disagrees

[1]Tracking scenario: Appendix A.1.1, no illu. changes, CODAPT Accurate-C.

the hypothesis, but might be a hint that 200 particles are sufficient for the proposed tracking methods or the search space resolution is set too coarse.



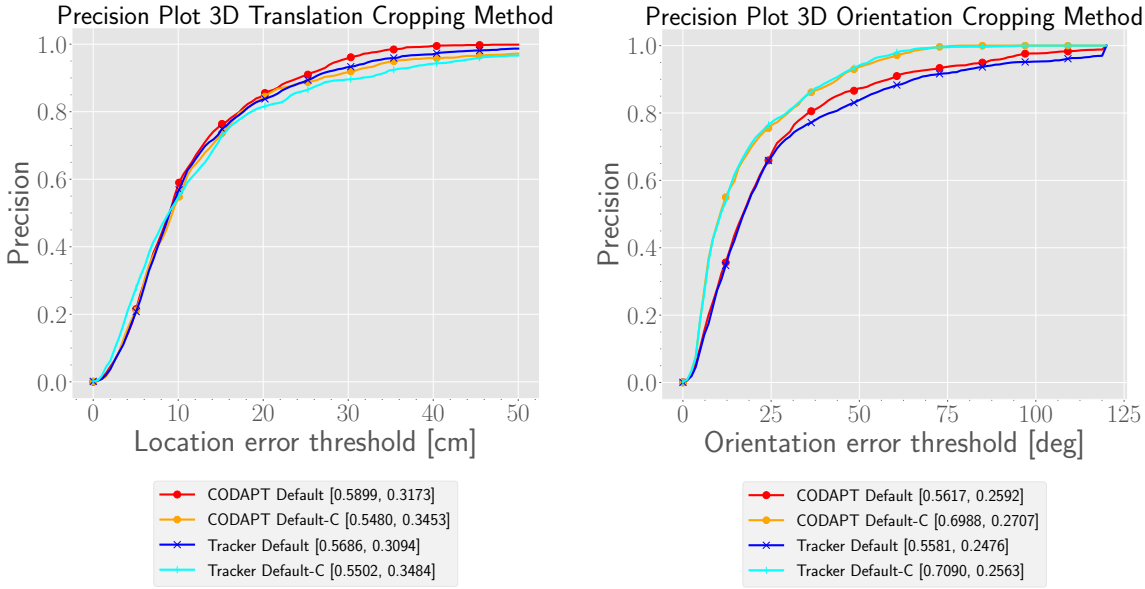
(a) (b)

Figure 6.3: Comparison of the 3D precision plot results for the proposed parameter setups.

6.1.3 Cropping Method

In Section 4.2.4, two approaches to crop point clouds are proposed, with respect to varying accuracy: an approximative 3D box crop and a more accurate convex hull crop. It is assumed, that the box crop cannot preserve the orientation information of the 3D object tracking as good as the convex hull crop, due to its drastic object shape simplification. Further, this simplification might lead to a less precise tracking. Hence, the 3D translation and orientation precision plots are considered for validation of the hypothesis, using the *Default* and *Default-C* parameter setups.
Figure 6.4a shows a small translational precision loss for the convex hull crop. At $x = 10$ cm, the precision differs 4.19% for CODAPT and 1.84% for the tracker. The standard deviation raises by +2.8% and +3.9%. For $x \leq 8$ cm, the tracker convex hull crop outperforms the box crop by up to +8%. In the other cases, the box crop leads to ≈5% more translational precision for CODAPT and the tracker. Although the results tend to get worse with the convex hull crop, there is no reasonable explanation at this point, than the influence of randomness.
On the other hand, Figure 6.4b indicates a significant difference for the orientational precision. At $x = 20$ deg, the convex hull crop leads to a precision gain of +13.71%

for CODAPT and +15.09% or the plain tracker. This trend continues for $x \neq 20$ deg. The STD is slightly worse for both convex hull graphs, but with 1.15% and 0.87% negligibly.
Overall, the results confirm the hypothesis, that the convex hull crop preserve orientational precision better than the box crop. Contrary to the hypothesis, the convex hull crop worsen the translational precision.



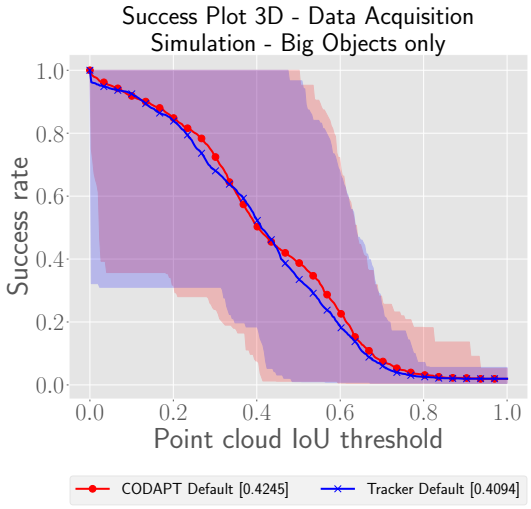
(a) (b)

Figure 6.4: Comparison of the 3D precision plot results between the box and the convex hull cropping method.

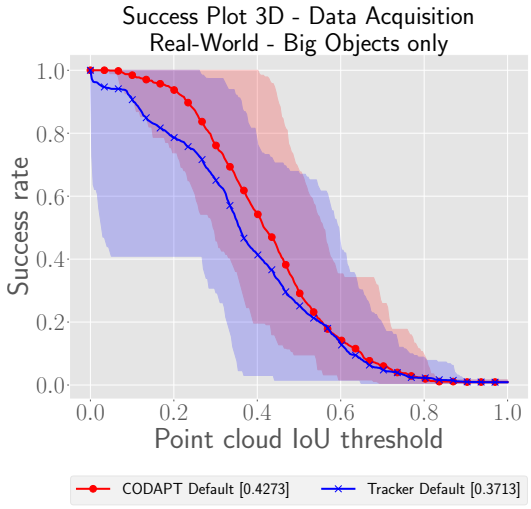
6.1.4 Dataset Acquisition

In Section 5.1.2, two types of datasets are proposed: real-world and simulated ones. The real-world results include confounding variables (Table 5.2), which are supposed to decrease the tracking performance. In this Section, the influence of the erroneous IPS trajectory and the IPS calibration is investigated, by evaluating all real-world and all simulated results. The influence of the former parameter is visible in the 3D precision plot, when comparing both datasets. Also, both parameters influence the overall performance, summarized by a 3D success plot.
When first observing the evaluation results, it was conspicuous, that the real-world results performed striking bad in contrast to the simulated results (see Figure C.2.1 in Appendix C.2). Manual inspections of the real-world results revealed, that both tracking methods suffer from many tracking losses, when tracking small objects from some meters apart. The ratio between distance to the object and object size to the stereo-camera baseline seems to be too small, to obtain enough 3D points to model these objects. At this point, enlarging the point cloud resolution, the

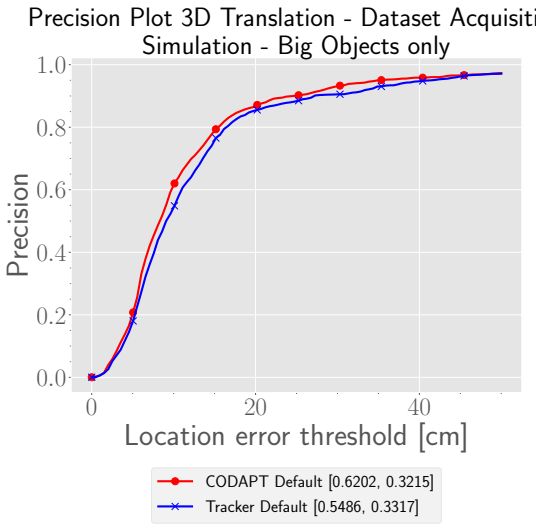
camera resolution or the baseline might be corrective actions. The influences of these actions are discussed later. Although a tracking is performed anyway, this bad performance influences the performance of the remaining results and make the datasets not comparable anymore. Another indicator is Table C.1.2 in Appendix C.1. It shows that trackings of the smaller real-world object classes *laptop* and *monitor* do not lead to as much overall associations, as for the other object classes. This is either due to a bad detection rate of YOLO or a bad tracking performance of the tracking methods. In either case, the worse performance is undesired. In the simulated dataset, the small object classes are designed bigger by default, to avoid this pitfall. To obtain comparable evaluation results, the small object classes laptop and monitor are dropped for both datasets in the remaining evaluation.
First, the erroneous IPS trajectory is considered, using the Figures 6.5c – 6.5d. Comparing both plots at $x = 10$ cm, the simulated precision is more than twice as high, than in the real-world data (+35.41% CODAPT and +30.41% tracker), despite the slightly higher STDs. As stated in Section 5.2.1, a closed-loop translation error of 0.23 m can be assumed as the trajectory error for the end of an IPS session. Since the plots do not distinguish between tracking start and end, the error is assumed for the whole IPS session as a conservative guess. Allowing this additional 0.23 m error (i.e. consider the precision of the real-world plot at $x = 33$ cm), the trackers precision raises to ≈85% and CODAPT's to ≈98%. This increase would close the performance gap to the simulation data. It can be therefore assumed, that the IPS trajectory error is responsible for the biggest part of this performance gap.
Since the real-world translational precision performances already outperforms the simulated ones, when only incorporating the IPS trajectory error, it can be no error attributed to other error sources.
Summarizing, an influence of the erroneous IPS trajectory is clearly visible in the results, leading to a maximum degradation of about 31% – 36% of the tracking performance scores. Effects beside the IPS trajectory error cannot be quantified using the results.



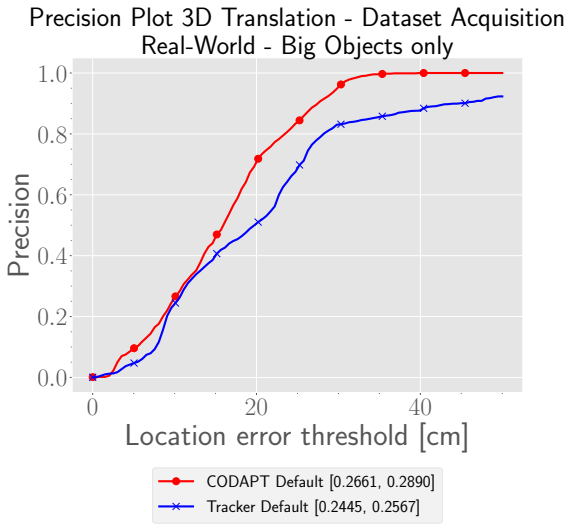
(a)



(b)



(c)



(d)

Figure 6.5: Evaluation results comparing the dataset acquisition methods (real-world vs. simulated).

6.1.5 Dataset and Object Attributes

The Section 5.2 proposes different tracking object classes and tracking scenarios, to suppress specific object and dataset scene parameters, that are assumed to influence the tracking methodology (see Appendix A). It is presumed, that the illumination changes do not influence the tracking much, since the geometric part of the CSHOT descriptor is of higher dimension, than the color part and therefore higher weighted, which makes the CSHOT descriptor robust against color changes. The same holds for the background clutter of the scenes, because this is assumed to be the strength of the tracking in the 3D space, in contrast to tracking in the 2D space. The object attributes on the other hand might affect the quality (object curviness, textureness and color) and the amount of descriptors (object size), which might lead to an observable change in the tracking results. All object classes, that have the highest contrast regarding the respective attribute, are evaluated. The attribute valuations can be found in Appendix A.
The Figure 6.6 depicts the results. As expected, the scene attributes have no noticeable influence on the performance. Contrary to expectations, the same holds for the object attributes. All object classes lead to a reasonable amount of CSHOT descriptors, that ensures a proper tracking. Further, object color and textureness are not as influencing as expected – probably also due to the addressed color change robustness of the CSHOT descriptors. The impact of object curviness is not strong enough to distort the normal estimation process and therefore the CSHOT descriptors, because the neighborhood-radius of the normal estimation process is with the configured 0.03 m too small, to get noticeably influenced.

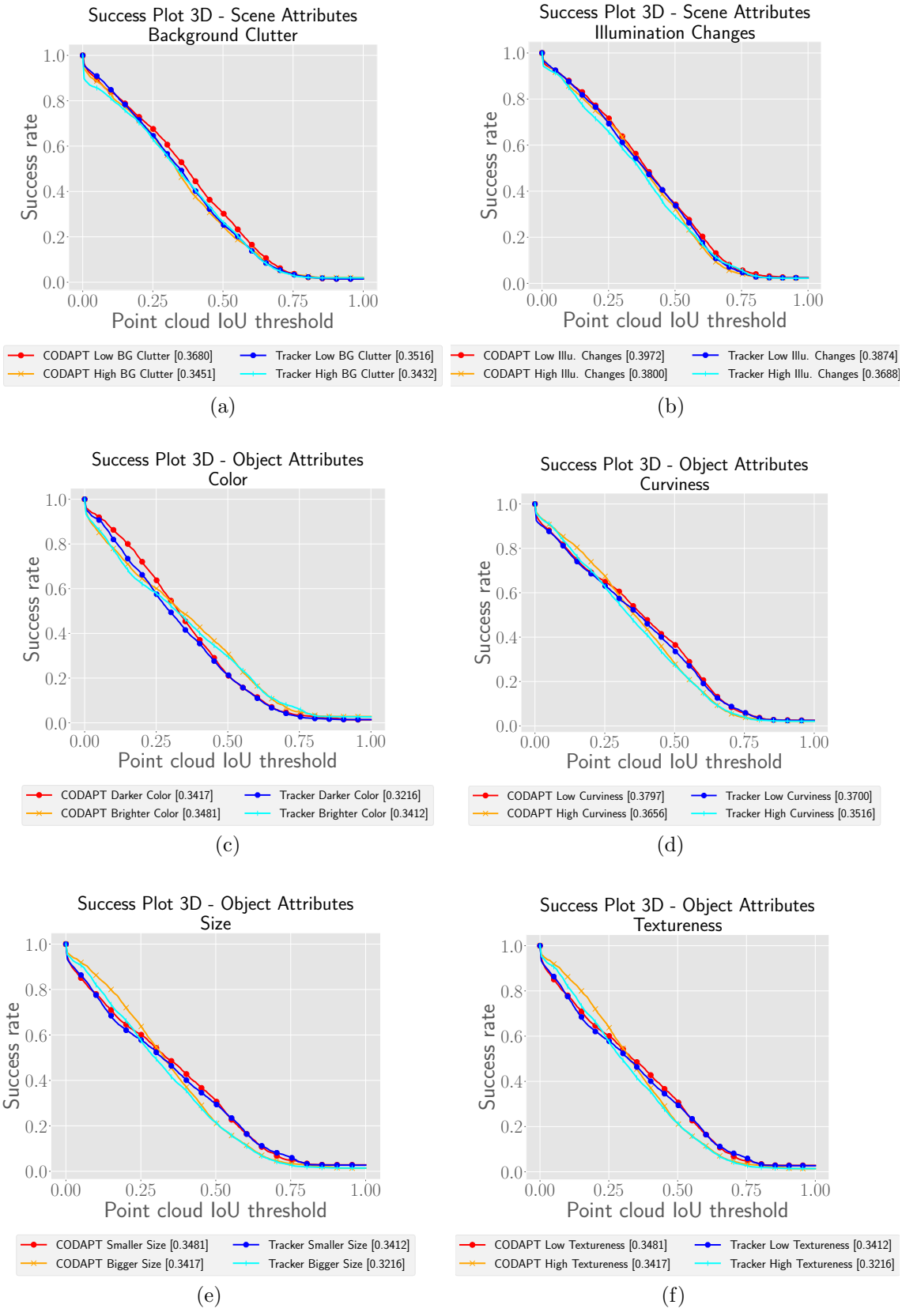


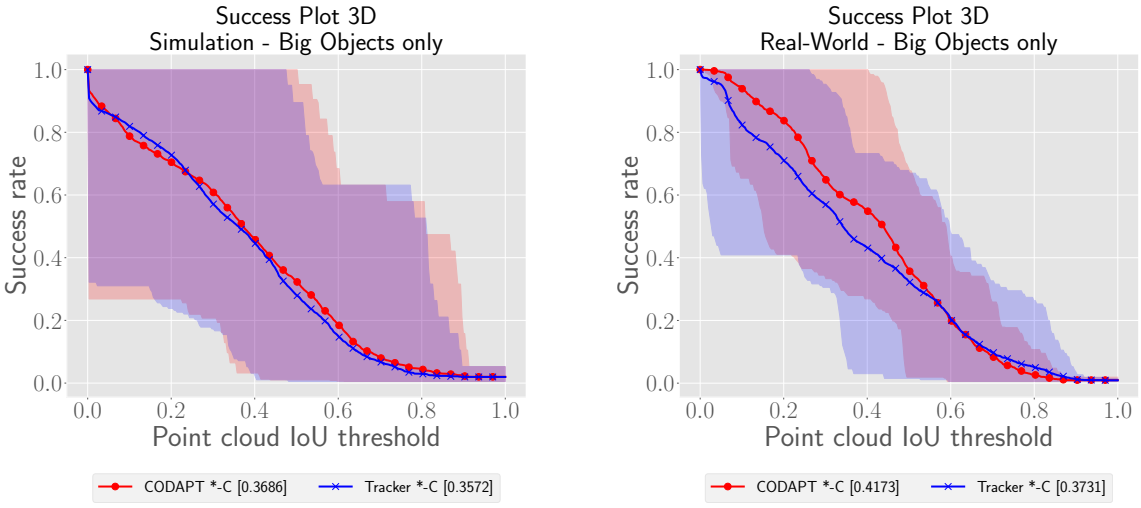
Figure 6.6: Comparison of dataset and object attribute results.

6.2 3D Object Pose Tracking

In this section, the results of the 3D object pose tracking are summarized. First, the overall 3D tracking accuracies are compared, using the success and precision metrics. Second, the runtime of CODAPT is investigated.

6.2.1 Accuracy

The 3D object pose tracking performances for the proposed datasets are investigated here to answer the question, whether the CODAPT approach can improve the isolated 3D object tracker.
Before analyzing the results, the pre-evaluation outcomes are considered. The pre-evaluation reveals that the performances of the different parameter sets do not differ in general, but the convex hull crop lead to a more accurate 3D orientation accuracy. In addition, results differ for simulated and real-world datasets. Hence, the data of the superior convex hull crop is examined in the following for the two dataset acquisition methods separately, whereby all respective parameter setups are involved and only big objects are considered.
At first, both methods are compared to each other regarding the success plots in the Figures 6.7a – 6.7b. Second, the translational precision plots in the Figures 6.8a – 6.8b are analyzed similarly. Finally, the orientational precision plots in the Figures 6.9a – 6.9b are evaluated and matched to the context of the plots before.



(a) (b)

Figure 6.7: 3D object pose tracking success performance results.

The Figures 6.7a – 6.7b show the 3D success plots for both datasets. In the synthetic data, both methods perform similar regarding the AUC score and their aberrations to the mean. A significant difference is not visible. However, CODAPT performs better than the tracker in the real-world plot (AUC +4.42%). At this point, this

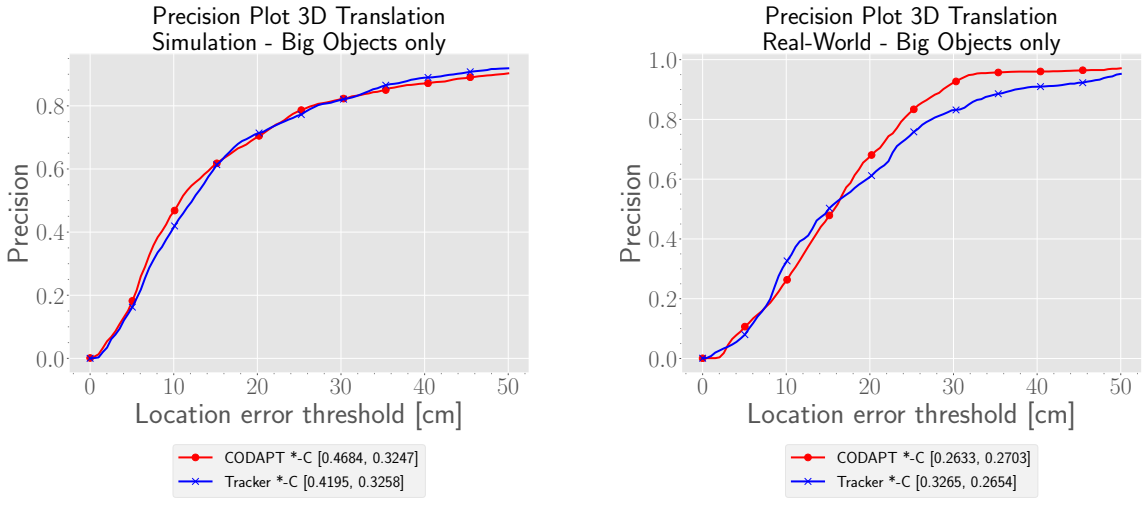
could be the influence of randomness but also a hint, that CODAPT takes advantage of the object detections. Although the amount of associated object detections of CODAPT are similar for both datasets regarding only big objects (see Tables 6.3), YOLO provides more precise bounding boxes in the real-world scenarios (see results in Section 6.3).

Table 6.3: Weighted mean of the associated object detection rates per dataset acquisition method, regarding the Tables C.1.1 and C.1.2.

Dataset	Weighted Mean $\bar{x}$
Simulation	0.66
Real	0.64

$$\text{with } \bar{x} = \frac{\sum_{i=1}^{n} ratio_i \cdot \#(GT\ detections)_i}{\sum_{i=1}^{n} \#(GT\ detections)_i}$$

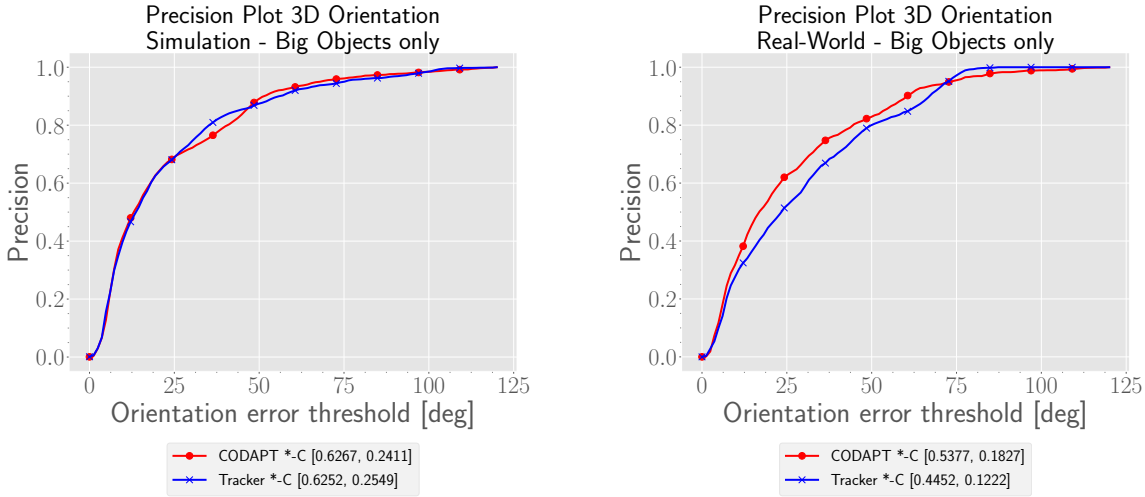
The tighter bounding boxes can make a difference, if the tracking begins to drift. By providing less points in the surrounding that can distract the tracker, the tracker is forced to stay in the provided cropped area, which leads to a higher likelihood, that the tracker retrieves the tracking object. This assumption is supported by the real-world graph, since CODAPT's performance is higher for IoU thresholds below 0.5 (e.g. around +10% at 0.4), i.e. if the tracking does not fit the true object accurately anymore. Too imprecise (or too loose) bounding boxes are not assumed to make a difference for CODAPT's accuracy in this datasets, because similar objects of the same object classes are mostly placed nearby, to distract the 3D object tracking. This holds especially for the simulation datasets.
For both methods, the aberrations to the mean are noticeable smaller in the real-world datasets. Since this is not only the case for CODAPT, this behavior is attributed to the less object trackings in the real-world dataset. Otherwise, the simulation results should perform better, since they are boosted by the ideal IPS calibration and trajectories.



(a) (b)

Figure 6.8: 3D object pose tracking translational precision performance results.

The translation precision plots in the Figures 6.8a – 6.8b reveal similar results. The STDs are similarly higher in the simulation plot, than in the real-world plot (at $x = 10$ cm). In addition, CODAPT reaches higher precision scores at $x = 10$ cm in the simulation plot and at $x = 33$ cm in the real-world plot, than the tracker (both $\approx$+5%) and both methods gain higher scores in the real-world results at the same x-coordinates (about +20% CODAPT and +19% tracker). On the other hand, the difference between both methods in the simulation plot is only pointwise and do not reveal a clear behavior. The same holds for the real-world plot: Below $x = 17$ cm, the performances of both methods alternate and no superior method is identifiable here. Noticeable are the real-world precision scores above $x = 17$ cm. As in the related success plot, CODAPT seems to obtain better results (at $x = 30$ cm +10%), when the tracking is already distracted. This underlines the assumed working principle for CODAPT, as well. As mentioned in the metrics introducing, too high errors are insignificant, because a completely lost object tracking can lead to arbitrary results. The joining lines at $x = 50$ cm indicate (similar as in the other graphs) the same level of randomness for both methods.



(a) (b)

Figure 6.9: 3D object pose tracking orientational precision performance results.

The orientational precision plot results of the Figures 6.9a – 6.9b fit to the precision results: Both methods perform overall better in the simulated data (+8.9% CODAPT and +18% tracker), due to the ideal IPS trajectory and calibration. Also, their performances differ to each other in the real-world data. On the other hand, the STD of CODAPT in the real-world data is higher than the trackers one, which overlays a part of the mean performance difference. Since the other plots do not indicate remarkable higher STD of CODAPT, this number here is accounted to the influence of randomness. The superior performance of CODAPT in the real-world cases for $10 < x < 70$ deg (e.g. about 9% at $x = 25$ deg) fits to the previous results and also supports the assumption, that CODAPT can make use of tight bounding

boxes, if the tracking is already slightly distracted. Although the Appendix C.2 lists results for all available object trackings, it indicates the same behavior of CODAPT. Summarizing, the results clearly indicate that CODAPT can exploit 2D object detections, to keep partly distracted object trackings within bounds. This performance gain of CODAPT amounts pointwise up to +10% and is supported by all applied metrics. Also, this earning is too large to be fully attributed to the influence of randomness. The bounding boxes have to be tight to improve CODAPT, since they would not exclude nearby distracting objects otherwise.

6.2.2 Runtime

In this Section, the runtime of the proposed tracking methods and their parameter setups are evaluated. The second part of Research Question (2) is answered here, whether the CODAPT approach can speed up the isolated 3D object pose tracker. The results in this section are supported by additional plots in the Appendix C.3, which separate simulated and real-world datasets.
Please note that the results for the *Accurate* and *Accurate-C* parameter setups are left out, because these would not provide an added value. The insights regarding the runtimes between the tracker and CODAPT are fully covered by the plotted parameter setups. Hence, the graph is simplified to improve readability.
The runtime plots can be seen in the Figures 6.10 and 6.12. The first plot shows the runtimes per frame, to normalize the different object tracking runs and make them comparable. The second plot depicts the runtimes per frame and per particle, to keep the respective groups in scale and find results, that are overlaid by overall high runtimes in the first plot.

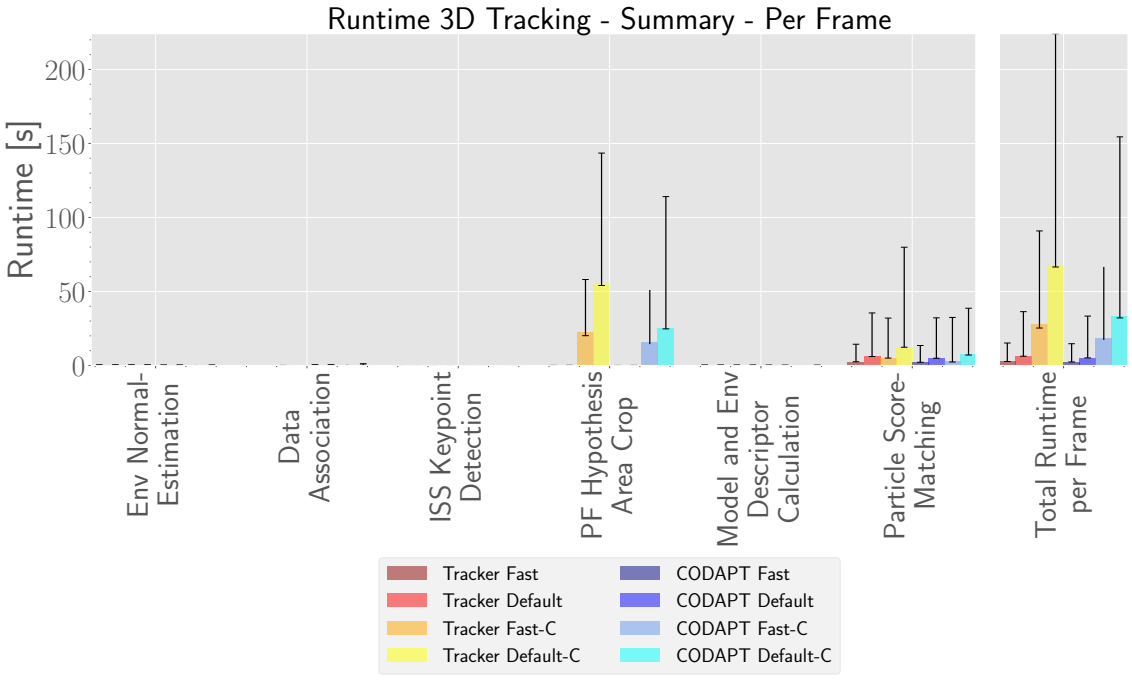


Figure 6.10: Runtime performance results (per frame).

First of all, the runtimes per frame in Figure 6.10 are investigated. It is clearly visible, that operations involving particles dominate the groups. In comparison, the other component runtimes carry no weight. As expected, the *PF Hypothesis Area Crop* component illustrates higher runtimes for both cropping methods and compared among themselves, runtimes proportional to the applied number of particles. The lower CODAPT runtimes of this component confirm, that the idea of the cropped region to process works: the *Fast-C* CODAPT setup runs 24% faster than the trackers one (≈19 s against ≈25 s per frame). The difference increases for the *Default-C* setup to 55%, since every particle benefits from the less amount of points to process (≈25 s against ≈55 s). Their runtime also scatter similar. The scattering enlarges with a higher amount of applied particles in general, due to the effect of randomness. The more particles are available, the more spread the different matching accuracies of the particles. A higher matching accuracy might probably lead to more CSHOT descriptors to compare, what could explain the runtime variations. The barely visible box crop runtimes are separately outlined in Figure 6.11 and behave similar to the convex hull crop method. For the *Fast* parameter setup, the difference amounts 21% (about 0.015 s to 0.019 s) and for the *Default* one 43% (about 0.02 s to 0.035 s). The advantage of CODAPT is also underlined by the Figures C.3.1a and C.3.1c in the Appendix C.3.

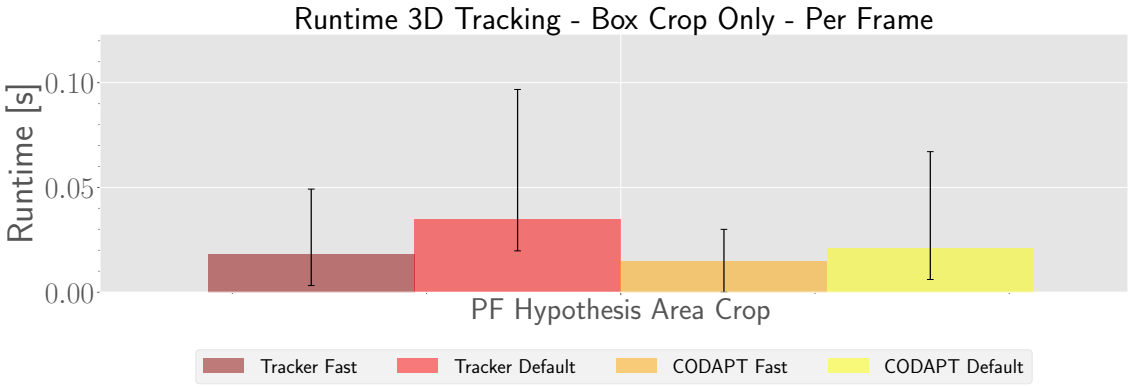


Figure 6.11: Runtime performance results for the box crop method (per frame).

In the *Particle Score-Matching* component, CODAPT seems to save runtime, as well. But due to less differences within the same CODAPT and tracker parameter setups, this behavior is attributed to randomness within the tracking. Considering the *Total Runtime per Frame*, the biggest part of the runtime difference can be attributed to the *PF Hypothesis Area Crop*. The respective runtimes of CODAPT are lower than the ones of the tracker. The more particles are applied, the more CODAPT benefits. The longer-running convex hull crop method profits the most in absolute numbers: for the *Fast-C* parameter setups, the saving per frame amounts ≈37% (≈19 s to ≈30 s) – for the *Default-C* setups, ≈51% (≈33 s to ≈68 s). The difference for the box crop methods amounts 20% (≈2 s to ≈2.5 s) for the *Fast* setup and 17% (≈5 s to ≈6 s) for the *Default* one. The Figures C.3.1a and C.3.1c in the Appendix C.3 reveal similar results, again.

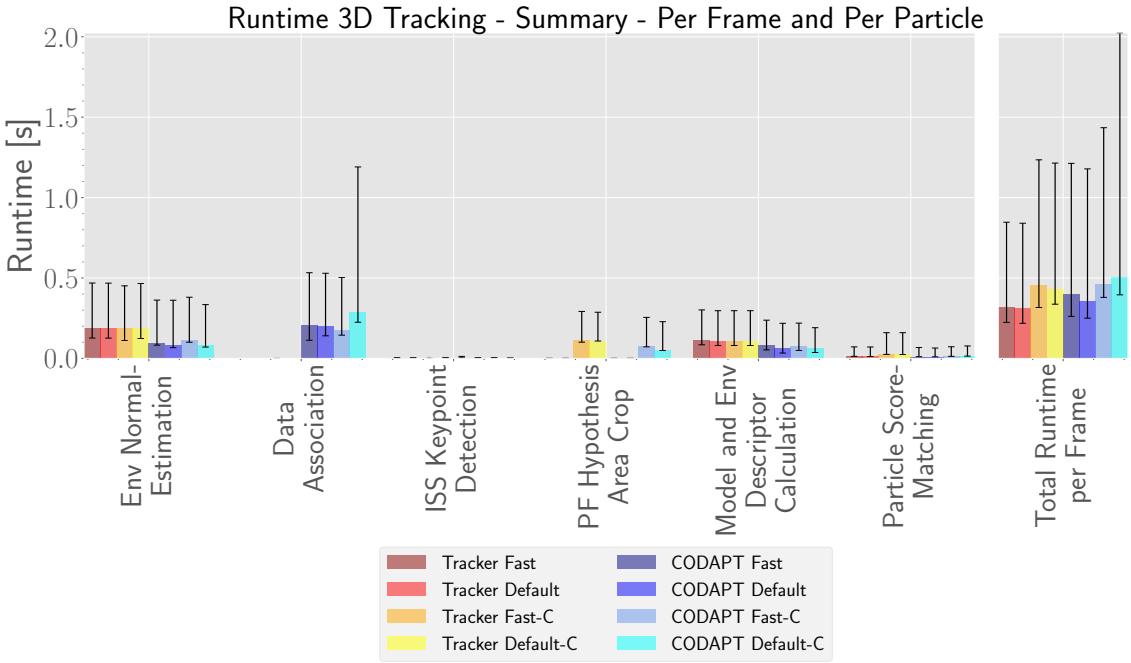


Figure 6.12: Runtime performance results (per frame and per particle).

The view of the runtimes per particle and per frame in Figure 6.12 reveals additional runtime savings of CODAPT. Since the *Env. Normal-Estimation* of the environment frame is performed after the environment cropping of CODAPT, less normals have to be estimated. The component group *Data Association* is the overhead of CODAPT, but small in comparison to the overall runtime. Since the CODAPT data association does not differ using different parameter setups, the runtime changes are attributed to the randomness of the tracking. Dependent on the tracking position, more or less object detections can be processed per frame. However, the detected objects in a specific frame for different parameter setups are constant, since YOLO works deterministic. Furthermore, CODAPT performs better in the *Model and Env. Descriptor Calculation* component. For a restricted point cloud, less ISS keypoints can be found in general and therefore, less CSHOT descriptors have to be calculated for each environment frame. Nevertheless, the runtime savings for the less features are barely visible. The *Total Runtime per Frame* summarizes the runtime savings and overheads for CODAPT: Considering one particle and one frame, the benefit of the object detections are canceled out by the data association and the total runtimes do not differ significantly.
Summarizing, the tracking clearly benefits from the CODAPT approach to save runtime. Besides minor runtime savings, the PF convex hull crop benefits the most. The overall CODAPT runtime savings depend on the ratio of remaining 3D points (with respect to all 3D points of an environment frame) and likewise on the number of particles, processing these points. The results show that up to 37% mean runtime saving could be reached for 200 particles and up to 51% for 500 particles.

6.3 2D Object Detection

This section points out, whether the 3D object tracker can improve the accuracy of the 2D object detections. Therefore, the results of the 3D object tracker and likewise CODAPT are projected onto the image plane, to make them comparable to the detections of YOLO. As in the 3D evaluation, the convex hull method is examined using all respective parameter setups combined. Additionally, only big objects are considered, to make the results relatable to the 3D ones. Simulated and real-world datasets are considered separately as before.
In the following, each method is considered separately for the success plots in Figures 6.13a – 6.13b and then compared to each other. The same procedure is performed for the precision plots in Figures 6.13c – 6.13d. In the end, the 2D scores are evaluated in the context of the 3D scores.
First, the YOLO object detector is considered briefly. Since it is trained on real-world images, it is expectably that YOLO performs better on them, than on the simulated ones (+16.18% and lower aberrations to the mean). This is mainly attributed to the gap between simulation realism and the real-world's one – this topic is picked up in the next chapter. Also, YOLO shows a lower miss rate in the real-world cases (≈17%), than in the simulated ones (≈29%). Besides, the results show the ability of YOLO to generalize on different data, as well.
Next, CODAPT and the isolated tracker are examined. The real-world success accuracies outperform the simulated ones with a difference of +23.15% for the tracker and +24.29% for CODAPT in the default success plots – both methods gain likewise. This is due to the fact, that the proposed datasets with the big objects are different challenging. In the simulated data, objects are partly further afar and out of view for a longer time, which makes the tracking more difficult and it also leads to more misses. The success plot for the simulated data shows a high miss rate of ≈50% for both methods, whereas the miss rate in the real-world cases are with ≈23% much lower. This is part of the discussion in Chapter 7.
The three methods are now compared to each other regarding their success score. CODAPT and the tracker perform similar in the simulation dataset concerning mean success performance, but the high miss rate does not allow a proper comparison of the method's mean performance aberrations. In the simulated data, YOLO performs clearly better, than both tracking methods (≈ +13%). In the real-world data, the tracking methods seem to can keep pace with the mean performance of YOLO (YOLO ≈ +5%), but YOLO's performance scatters less, though. As in the 3D evaluation, the real-world results are biased by the influence of the erroneous IPS calibration and trajectories, but in a different way: since the 3D object tracking results are back-projected onto the image plane using the inverse of the already applied transformation (to project the scene into the local IPS coordinate system), the projecting error cancels out again. What remains is the error of the 3D object trackings, that emerge as a consequence of the projecting error. This error cannot be quantified by the used metrics, but incorporating it would most probably lead to higher success scores of the tracker and CODAPT. Having this in mind, both

tracking methods probably outperform YOLO in the real-world scenarios. Apart from that, YOLO captures the tested object classes more often and more correct in the simulation dataset.
In the following, the precision plots in the Figures 6.13c – 6.13d are analyzed, starting with YOLO. Here, YOLO leads to similar results as in the success plots: Higher precision scores and lower STDs in the real-world cases, compared to the simulated ones (+15.35% and +13.83%).
CODAPT and the tracker perform similar again in the simulation results, regarding mean performance and STD. In the real-world plot, CODAPT seems to reach a slightly higher mean precision (+3.92%) than the tracker, but also a higher STD (+2.39%) – the difference is too low to be considered as significant. In both datasets, YOLO outperforms the both trackers. Although YOLO performs slightly weaker in the simulation data at $x = 50$ pixels, YOLO achieves clearly better results for all $x < 40$ pixels. In the real-world results, CODAPT and the tracker cannot stick with YOLO's performance ($\approx +32\%$). Even when incorporating the error in consequence of the projecting error, the precision performances would probably not match up to YOLO's – the difference is too large. In both plots, YOLO's bounding boxes are mostly more precise.
The 3D success scores of Section 6.2.1 can be compared with the 2D ones. It stands out, that nearly all 2D success scores are remarkably higher. Besides the already mentioned error in consequence of the projecting error, one reason for the higher scores in the 2D scores is accounted to the effect, when projecting a 3D space onto the 2D space (see Figure 6.15). Although these exemplary object trackings mainly fail in the 3D space (low point cloud IoU scores), both seem to succeed in the 2D space (higher bounding box IoU scores). This behavior leads to same results as a proper tracking in the 2D space here, but they are undesired, since they distort the performance analysis. This effect is also expected to influence the 2D precision.
The Figures in Appendix C.4 contain plots with small objects as well and support the results of this section. Although the real-world precision plots (from this section and the appendix) seem to support the assumption of the 3D evaluation, that 2D object detections improve the performance of CODAPT, the projection issues are considered as too large to make a clear statement here.
All in all, YOLO mostly performs more stable and outperforms the tracker and CODAPT clearly in the simulation dataset. YOLO also produces more precise object detections with a lower miss rate. When incorporating the errors as the consequence of the projection errors made by the IPS, the trackers might keep pace with YOLO's performance and can even lead to superior results. The Figure 6.14 illustrates exemplary situations, where either YOLO or the tracker performs better. Despite, this has only been shown for not too challenging data. Further, the results of the trackers are biased by 3D-2D-projection effects and probably too high. Due to the same effect, a significant difference between CODAPT and the tracker could not be stated.

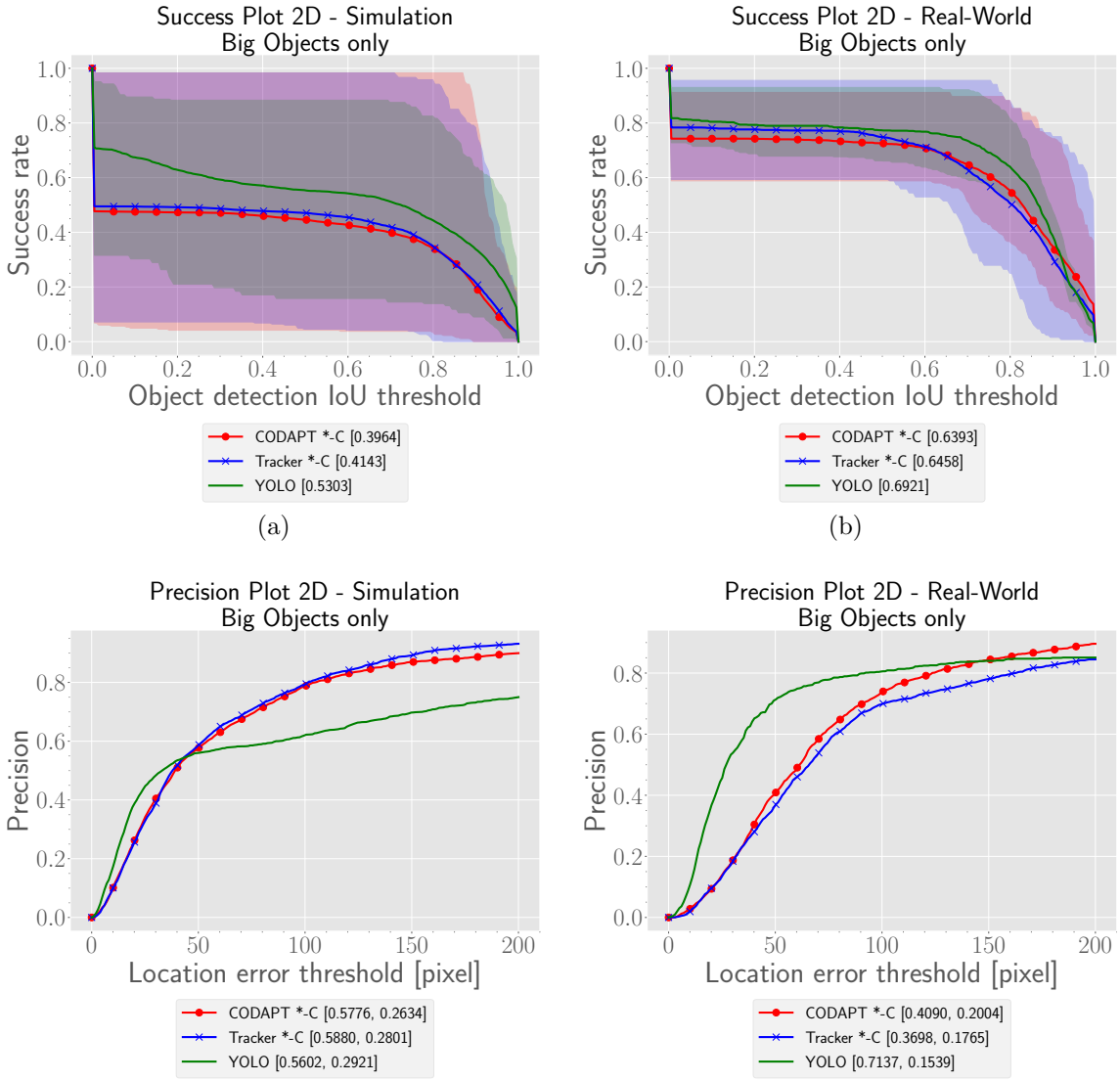
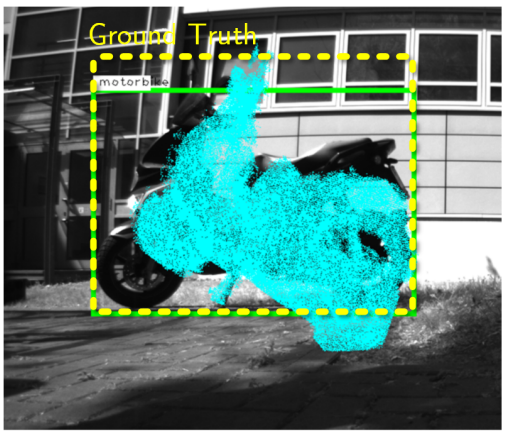
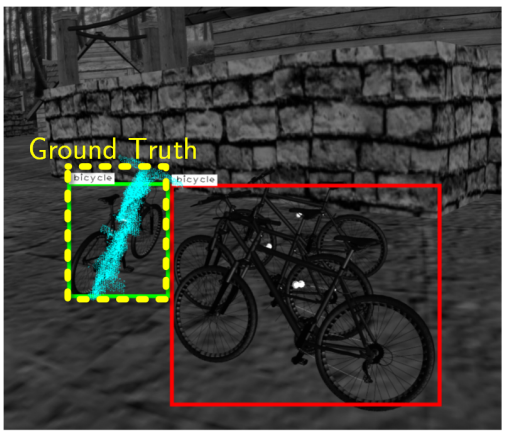


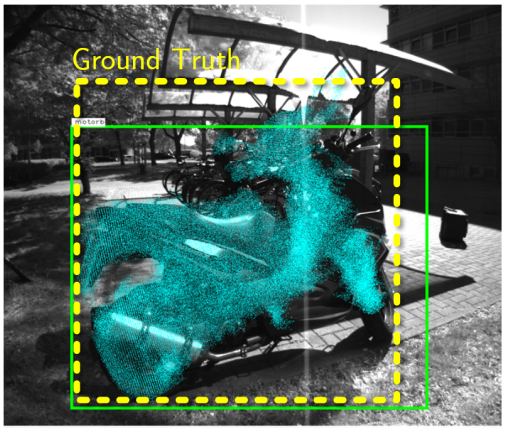
Figure 6.13: 2D object detection accuracy results.



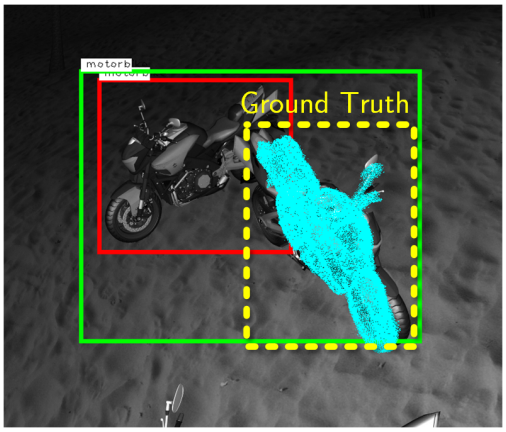
(a) YOLO performs better.



(b) YOLO performs better.

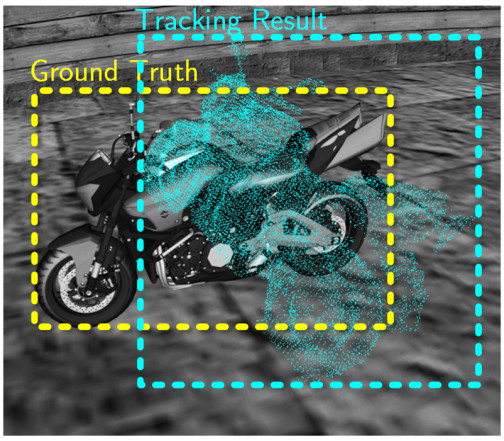


(c) Tracker fits the tracking object more precisely.

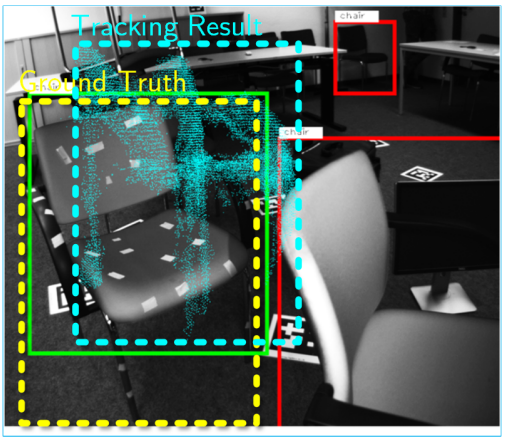


(d) Tracker performs better, because YOLO is distracted by objects nearby (big YOLO bounding box).

Figure 6.14: Exemplary situations comparing the results of YOLO and the tracking methods. The green bounding boxes denote associated YOLO object detections, the red ones non-associated YOLO detections. The blue mask depicts the current best object tracking hypothesis and the yellow dashed line the actual ground truth position.[2]



(a)



(b)

Figure 6.15: Exemplary tracking situations, that produce low 3D success scores but higher 2D success scores. The dashed blue bounding box represents the back-projected tracking result, the rest remains as in Figure 6.14.[3]

[2]Tracking scenarios: (a) Appendix A.2.1, second run, CODAPT Default-C; (b) Appendix A.1.2, no illu. changes, CODAPT Default-C; (c) Appendix A.2.1, second run, CODAPT Default-C; (d) Appendix A.1.1, with illu. changes, CODAPT Accurate-C.

[3]Tracking scenarios: (a) Appendix A.1.2, no illu. changes, Tracker Accurate-C; (b) Appendix A.2.3, without illu. changes, CODAPT Fast.

Discussion

This thesis utilizes proven state-of-the art components of the literature, considers verified conclusions of other works and proposes thereout a combined 3D object pose tracker and 2D object detector method named CODAPT. CODAPT is used to investigate a mutual improvement for both tasks and designed against the backdrop of requirements like general applicability, high precision and fast response times, to conform the needs of optical navigation scenarios. It allows the tracking of general rigid and non-movable objects, to make inferences of self-movements in 3D scenes. The performances are then evaluated by using well established metrics. To ensure evaluation results of high generality, the datasets are permuted with respect to different reasoned object and environment properties.
The following discussion considers each research question separately. First, the Research Questions (2) and (3) are discussed. Involving these outcomes, the method CODAPT proposed in the course of Research Question (1) is examined afterwards. Finally, limitations of this work are mentioned.

7.1 3D Object Pose Tracking

The second research question investigates the influence of the 2D object detector on the accuracy and the runtime of 3D object tracker, when combining both. The results clearly indicate that the 2D object detections can improve the accuracy and runtime performance of 3D object tracking.
Accuracy boosts can be observed in the results, when CODAPT is partly distracted from the real object to track. The by the 2D object detections cropped area limits the 3D hypothesis area of the tracking and hence counteracts a drift – the tighter the detections, the better. The results exhibit a performance gain of up to 10% for the success, translational- and orientational-precision scores. Although YOLOv3 performs well on the proposed datasets (see Figure 6.13 and Appendix C), the 2D object detections are not perfect yet. Regarding the fast improvements of deep learning methods, 2D object detectors are expected to enhance their performance in the near future. Hence, a higher detection rate and tighter 2D object detections can be presumed, which are supposed to further improve the CODAPT performance.
Algorithms working in 2D are already exploited yet, to enhance the accuracy performance of 3D applications. The results of this work support the outcomes of (Osep et al., 2017) in the way, that 2D object detections can provide extended localization capabilities to enhance the precision of 3D object trackings. Furthermore, the results correlate with the ones of (Zarzar et al., 2019) and underline the statement, that 3D search area proposals from 2D object detections can improve precision and success performances of a 3D tracking application. 2D object detections are not utilized in 3D tracking applications only, but also in pure 2D tracking applications – this paradigm refers to as the already introduced tracking-by-detection. 2D tracking-by-detection methods are already established to tackle challenges like

target dynamics, intra/inter-class variation, measurement noise, realtime capability and enduring tracking losses caused by the recursive tracking paradigm (Khan et al., 2005; Y. Li et al., 2008; Breitenstein et al., 2010). The evaluation results of this thesis show that the proposed 3D tracking method suffers from partly high miss rates and could profit from a tracking-by-detection paradigm. However, tracking-by-detection approaches for general objects working in the 3D space are not well researched yet. Especially, it lacks of 3D object detectors that are as powerful and as general as 2D ones. Further, the open available 3D data basis is not extensive enough to keep pace with 2D object detectors like YOLO and its capability to detect 9000 object classes. Nevertheless, there are promising works of recent years like (C. R. Qi, L. Yi, et al., 2017; C. R. Qi, W. Liu, et al., 2018) that might be a good start point for future work.

In addition to the observed accuracy boosts, the results have clearly shown a runtime improvement of CODAPT, too. The by the 2D object detections cropped 3D area reduces the number of points to process. Global or semi-global methods, that recurrently check all points for certain conditions in each new frame or for each particle (like the PF hypothesis area crop that checks for each point whether it belongs to a convex hull or is inside or outside the convex hull area), benefit the most. The results show a mean runtime saving of up to 37% (from 30 s to 19 s per frame) for 200 applied particles and up to 51% (from 68 s to 33 s per frame) for 500 particles. The more particles, the higher the runtime saving ratio. The CODAPT overhead for data association does not fall into account. Although the runtime could be decreased noticeably, CODAPT is by far not real-time capable and therefore not suited for a navigation task, yet. CODAPT would not allow fast reactions to deal with safety-related problems like obstacles avoidance.

The components, which have to be performed per particle, have been proven to be the runtime bottle-neck. This holds especially for the convex hull crop. This already approximative method implements the quickhull algorithm (Clarkson and Shor, 1989) with a runtime complexity of $\mathcal{O}(n \log n)$. The runtime complexity class can barely be enhanced using deterministic methods (compare to sorting and searching methods in (Knuth, 1998)), but the overall runtime can be influenced. Starting points might be to apply a GPU implementation (A. Stein et al., 2012) or to investigate k-d tree based intersection approaches, that exploit the internal representation of subsampled point clouds and might provide different approximative methods (compare with (Radu Bogdan Rusu, 2010)). Alternatively, simpler methods like the box crop can be applied, which comes with a reduction in cropping accuracy, as the results indicate (for box crop about -14% mean orientational accuracy). Since runtime and accuracy can be threaten as similar important for safety-critical navigation tasks, the actual preference depends on the specific application. Another leverage point could be to accelerate the PF. Therefore, the framework could be parallelized on a GPU (Concha et al., 2018) or one part of the search space could be analyzed analytically before applying the PF (Doucet et al., 2000). The former approach could for example speed up the particle filter weighting by up to 15 times,

compared to a similar CPU as used for this work. The easiest way to save runtime might be to reduce the number of applied particles. However, this would result in a too sparse sampling, when comparing to other PF applications (about 480 particles in (Vatavu et al., 2015) or around 1000 in (Concha et al., 2018) for reasonable results).
The runtime results using 2D object detections fit to comparable works, though. The work of (Osep et al., 2017) and (Zarzar et al., 2019) could also register runtime improvements, using 2D object detections, as region proposal cues for the 3D trackings. On the other hand, (Leibe, Schindler, et al., 2008; Leibe, Cornelis, et al., 2007) noticed overall high runtimes for 3D tracking with 2D object detection, as well. Likewise, their proposed method is not real-time capable, although many vehicle applications rely on the strong prior of a reduced search space dimension (in the near of the street 2D plane). Hence, this work supports the impression that real-time capable, general 3D object tracking is still difficult to accomplish. A solution would be to still make assumptions, that can be made for specific application scenarios, to restrict the problem space (e.g. search objects along the 2D ground plane in vehicle applications).

7.2 2D Object Detection

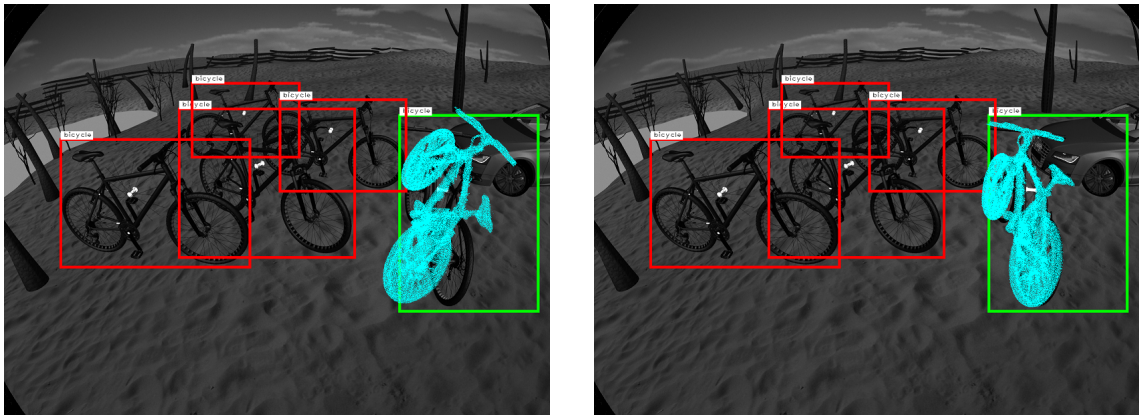
The third research question investigates, whether the 3D object tracker can improve the 2D object detector YOLO. Therefore, the accuracies of both object detection approaches are compared. If the 3D tracker leads to better results, it is assumed that it can improve the detections of YOLO. Further, in that case it can train (or refine) YOLO on automatically labeled objects, without worsen its overall accuracy performance. Respecting the results, YOLO mostly performs more accurate, more stable and more robust than the tracking methods. However, the tracking error, that can be attributed to the IPS projection error in the real-world data, is not quantifiable and the results of the trackers are biased by 3D-2D-projection effects, in general. Hence, a reliable statement, if the tracker methods can at least keep pace with YOLO's performance, is not possible. With the focus on high reliability for navigation tasks, the 3D tracker and CODAPT are not suitable to improve the performance of YOLO, yet.
Recent work also aims to improve the accuracy of 2D object detectors (semi-)automatically, using object trackers. For instance, the work (Teng et al., 2018) proposes a method to improve 2D object detections by 2D object tracking results and to fine-tune the detector at runtime. To induce reliability of the results, the user interactively chooses and labels the desired objects. Similar to CODAPT, this method also produces partly inaccurate results and misses. A different approach of (Papadopoulos et al., 2014) exploits a human eye tracker, to automatically segment and detect objects, which supports results of a 2D object detector – either online or offline. However, both works indicate that their performances do not reach the one

of established manual data labeling. Methods reducing the strong human supervision seem to be too unreliable yet, to improve 2D object detectors. This holds for an improved CODAPT method, as well.
A different direction of works investigates the improvement of 2D object detectors accuracies, by training them on simulated data completely or partly. Rozantsev et. al. propose an approach to synthesize images from simulated 3D object models. They state that simulated images yield significantly better performances than reusing augmented real-world images for training. Crucial for the results might not be the grade of realism "[...] in terms of image quality, but rather in terms of features used during the detector training" (Rozantsev et al., 2015). Unfortunately, modern DCNN 2D object detectors automatically learn complex features that are difficult to interpret for humans. Hence, image quality still might be an important factor and cannot be discarded.
Although simulated data might support detectors, the work (Georgakis et al., 2017) indicates a gap between simulated and real-world data. Surely, this gap strongly depends on the actual employed simulation environment, but it serves as a hint here, that the evaluation results of this thesis can be indeed explained because of this. Also, this supports the suggestion that YOLO performs better on real-world data, when learning on them.

7.3 CODAPT

The first research question asks for an approach to combine a 3D object pose tracker with a 2D object detector (CODAPT). The results have shown the successful application but also the limitations of CODAPT. CODAPT produces halfway precise results but also suffers from tracking losses and tracking distractions by other objects.
The overall performance of CODAPT is difficult to compare to other approaches, due to the self-provided datasets. On the one hand, these datasets could be created with regard to the needs of general navigation tasks, i.e. general object classes, different environment scenarios and free camera movements. Further, the data basis is compatible with the IPS framework by default. On the other hand, established datasets could allow a more extensive evaluation and would provide a common basis to compare CODAPT with different tracking approaches. Known open-available datasets providing 2D images and/or 3D point clouds are for example KITTI (Geiger, Lenz, and Urtasun, 2012), Ford Campus Vision and Lidar Dataset (Pandey et al., 2011) or the SLAM dataset by (Sturm et al., 2012). The former two represent a vast variety of traffic scenario datasets recorded by vehicles and the latter robot SLAM applications. The major reason against the usage of these datasets is the time limitations of this thesis. A wrapper for these datasets would have required a more in-depth analysis of the comprehensive IPS framework. Hence, this is considered as future work.

CODAPT can still be compared qualitatively to other object tracking frameworks, in terms of runtime, tracking loss situations and precision, as a rule-of-thumb estimate. The work of (Leal-Taixé et al., 2015) proposes a benchmark for multiple object tracking in 2D and 3D, also including non-rigid objects, which is considered as more challenging. In addition, the state-of-the-art results are continuously updated on the referenced website. These results indicate that 3D applications have a higher likelihood of keeping track of objects and achieve less tracking losses, but are less precise than 2D approaches yet. Reasons of this are not mentioned, but the 3D advantages are attributed to the capability of better handling occlusions and distractions by clutter, because of the heavily occluded benchmark scenes provided. These results support the choice of 3D approaches for object tracking, since these advantages correlate with more robustness. The less tracking precision counteracts another requirement of this work. Likewise, it motivates future work to further enhance 3D scene capturing and 3D tracking approaches, since 3D applications can make use of more information than 2D applications in general and should therefore at least perform equally precise. All the proposed methods (2D and 3D) are more precise and faster than CODAPT. They also lack of complete trackings and record similar high tracking losses as CODAPT. This is mainly attributed to the more challenging datasets, but the results indicate that not only CODAPT has these problems.
There are various possible ways to enhance the performance of CODAPT. The first general start point is parameter adjustment. Since the orientations of the example tracking masks in the Figures 6.14a – 6.14c are noticeable inaccurate, a further object tracking is performed with a lower particle orientation covariance (1.0 instead of 10.0 – compare with Appendix B.2). Figure 6.15 depicts exemplary results.

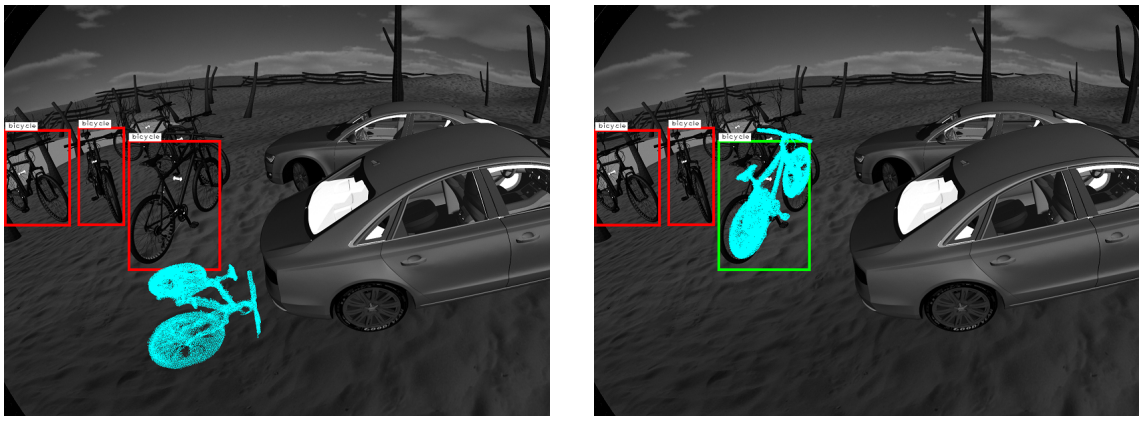


(a) Default parameters. (b) Adjusted parameters.

Figure 6.15: Exemplary tracking with adjusted orientation parameter. The colored markers are the same as in Figure 6.14.[1]

[1]Tracking scenario: Appendix A.1.1, without illu. changes, CODAPT Default-C.

This enhanced orientational precision can be observed during the whole tracking and it illustrates the potential improvement for other parameter adjustments, as well. A too coarse sampled tracking search space could also explain the similarity of the different parameter setup (*Fast, Default* and *Accurate*) results and the noticeable influence of randomness. However, a too fine-granular search space might bee too limited and might lead to more tracking losses. As a middle way, the parameters can be adjusted dynamically, for instance choose a higher search space resolution, if the probability of a tracking match is high (high CSHOT matching score) and lower the resolution otherwise.
Generally, CODAPT could benefit from more decisions made dynamically. The need of a beforehand created point cloud model with given starting positions makes the current CODAPT approach inflexible. This is also due to the limited time of this thesis. As an enhancement, CODAPT could create point cloud models dynamically, exploiting generative tracking approaches. They can also improve tracking robustness, by updating the object model iteratively continuously, to involve environment changes (e.g. dirt, light or weather conditions). Alternatively, a combination of generative object updating and prior knowledge in form of CAD models can be exploited, as proposed in (Wuest and Stricker, 2007). To apply CODAPT in general and unknown environments, such an extension is necessary.
Besides the parameter adjustment, the CSHOT descriptor matching could be changed. The PCL provides for example an iterative feature descriptor registration in its registration API, which was also considered as an implementation for CODAPT. This approach did not lead to more precise results or runtime improvements, though. Also, it is not suitable if multiple similar looking objects can be involved, since they might be not distinguishable only having the CSHOT descriptors. The PF is considered as more powerful for this task, because of its capability to incorporate proximity of tracking object between frames and its flexibility to adjust various parameters (resolution, number of particles, ...), to handle different scenarios.
Another factor to improve might be the chosen ISS feature points or the CSHOT descriptors. The used PCL tracking framework already provides features to perform tracking – it is able to determine the difference of point colors, surface normals and distances between two points, by default. The first implementation of CODAPT utilizes this features, but prior tests have shown a low expressive power of them. The Figure 6.15 depicts the major problem relying on these "simpler" features: the trackings get quickly distracted by the features of the surrounding. Colors and surface normals are not expressive enough to tackle this problem. Hence, they are only considered as the basis in more advanced feature descriptors like CSHOT. The introduced 3D feature comparisons in the works of (Guo, Bennamoun, et al., 2016) and (Hana et al., 2018) all rely on more complex features descriptors. Although the ISS/CSHOT combination is proposed as the best one in the benchmark, when it comes to best accuracy by low runtimes, there are more accurate ones that can be investigated, since the runtime components including ISS and CSHOT are not detected as a bottle-neck in the results of Section 6.2.2.



(a) Default PCL tracking features. (b) ISS/CSHOT features.

Figure 6.15: Exemplary comparison between tracking results, using the default tracking framework features and ISS/CSHOT features.[2]

A further crucial component, influencing accuracy and runtime performances, is the data association. In CODAPT, the data association is consciously kept simple, to save runtime. Even this simple data association has shown noticeable performance improvements for CODAPT and taking the runtime results into account, there is space to make use of more extensive approaches. MHT (Reid, 1979) or Joint Probabilistic Data Association (Fortmann et al., 1983) could for instance involve multiple concurrent object trackings and tracking hypotheses, i.e. not only the best particle, the most probable object detection class and the best fitting object detection can be analyzed, but all of them. On the other hand, processing more data means more runtime. There is still recent work that pay attention to runtime improvements (Hamid Rezatofighi et al., 2015) of data association methods.
The last considered type of improvement concerns the resolution of different components. To incorporate the discarded small object classes and objects farther away, these improvements are necessary. First, the spatial camera resolution can be increased. Alternatively, the stereo-camera baseline can be increased or a LIDAR can be employed. All three could lead to denser point clouds in the near- and far-range. Moreover, the camera's radiometric and spectral resolutions could be increased. Both could enhance the CSHOT descriptor resolution and could lead to more robust descriptors. Since the three mentioned resolution types are all considered as important and they can generally only be increased to the disadvantage of another of these resolutions, this topic is mainly considered as a matter of technical progress. Each enhanced resolution (without sacrificing another one) would lead to more features to process and hence to increased computation time, though. On the other hand, the increase of one resolution, when sacrificing another one, could advantageous for certain applications (e.g. higher radiometric resolution in areas with low color contrast, for more robustness).

[2]Tracking scenario: Appendix A.1.1, without illu. changes, CODAPT Default-C.

What remains is the question, whether the results of CODAPT can be generalized. The underlying idea is to employ the semantic detection and localization capabilities of the 2D object detector and make use of these information in a 3D tracking method. But this holds only for fully recursive tracking paradigms, i.e. trackings, that rely on the results of previous frames/frame-batches in a frame sequence (for offline working trackers in many approaches also the successive frames/frame-batches). Trackers that do rely on recursive tracking, handle each frame independently, i.e. perform the 2D object detection task already inherently (e.g. search a given object template in an image). In such a case, the result of an "external" 2D object detector could at least improve the detection confidence of the tracker (as an ensemble method). This holds also vice versa, when improving the 2D object detector with the results of the 3D tracking. For sure, the improvement of a CODAPT approach depends on the power of its single methods. An already precise and robust tracking would not benefit much from additional detections and the worse an object detection, the worse is its improvement factor for the tracking application. Considering a different point of view, the success depends also mainly on the data association part. If both isolated methods propose conflicting data, the data association have to decide. At each frame, a bad decision can lead to a tracking loss. So the CODAPT approach can also be harmful for the overall performance of both methods.

7.4 Limitations

This work and its results have potential limitations. On the one hand, the self-made datasets are too limited to make proper statements for general objects and environments. On the other hand, the limited access to measurement equipment prevented to gather ground truth data in the 3D space.
The exclusion of small objects in the evaluation halved the size of analyzable real-world data and discarded a third of the simulation data. Although the data is still valid, this lower sample size reduces the generality and the precision of the results. Moreover, mainly ideal objects and environments are represented in the data. The data do not cover challenging objects like soiled or damaged ones and different atmospheric conditions, such as rain or dust in the air.
The missing measurement equipment led to the workaround of using the starting pose of the rigid and non-movable objects in the local IPS coordinate system as 3D ground truth. Hence, the proposed evaluation method is adjusted to the IPS framework and has to be edited, to allow general hardware equipment and datasets. Actually, this workaround exploits the advantages of the already existing navigation capability of the IPS.
To overcome this limitations in future work, it is recommended to create the addressed wrapper to apply external datasets to the IPS framework. This would tackle the two limitations at once: At first, it makes data available to analyze more varying object classes and environment scenarios. Second of all, one can avoid the necessary measurement equipment and can make directly use of the ground truth data.

Conclusion

This work has investigated a combination of a visual 3D object tracker and 2D object detector for mutual improvement in the light of optical navigation tasks. The tracker was first implemented to conform state-of-the-art object features and then extended to make use of YOLO object detections projected in the 3D space. This approach, called CODAPT, was then compared to the results of the isolated tracker and to YOLO's object detections. The experiments were performed in custom real-world and simulation datasets.

The first research question considers the design, implementation and discussion of CODAPT. The tracking part mainly exploits ISS keypoints as well as CSHOT descriptors as extensive object representation. Both proved to be fast to calculate and invariant against different object and scene attributes. If the object class and the localization of an object detection fits to a current object position hypothesis, all points outside are discarded for further processing – this has been shown to be fast and reliable too. An extended particle filter was applied then to estimate the objects pose. The experiments indicated a noticeable influence of randomness and only minor improvements with higher number of particles and iterations. However, both were attributed to a parameter setup that can still be fine-tuned.

The second research question concerns a potential improvement of the 3D object tracker regarding accuracy and runtime, when applying YOLO's object detections. The tracking accuracy noticeably increases in the cases, where the tracking was partly distracted. The object detections avoid further distraction by restricting the search space. It did not reverse the multiple occurred object losses, though. The runtime also clearly reduces due to the limited area to process. Besides minor improvements relating to computations once per frame, the hypothesis crop operation executing once per particle profits the most. The higher the number of saved point comparisons and the higher the number of applied particles, the more the cropping method benefits.

The third research question deals with a potential accuracy improvement of YOLO's object detections, when applying the results of the 3D object tracking. The results first revealed superior performances of the 3D tracking approaches. However, a part of these performances were attributed to undesired 3D-2D projection effects. Despite, YOLO performed generally more robust and more precise. Due to the advantageous YOLO and the unreliable tracking results, the tracker is not considered to support the YOLO object detections.

All the evaluation results are limited in their expressiveness, by reason of limited datasets as well as the accuracy and precision of the IPS. Nevertheless, the evaluation results demonstrate a relevant benefit of the CODAPT approach for 3D object tracking – in particular, the results are applicable to various recursive tracking approaches. Although the proposed method does not fit the requirements of a navigation task yet, it has been shown to form a basis for several further improvements.

8.1 Contribution

This thesis provides primarily the following contributions:

- Evaluation of the stated research questions.
  This work proposes and evaluates a combination of visual 3D object tracking and 2D object detection for mutual improvement in accuracy and runtime performance.

- First implementation of a 3D object pose tracker for the IPS in the DLR.
  The implemented tracker is the first module for the IPS that can track objects in the 3D space.

- First extension of the particle filter framework from the point cloud library with feature point descriptors.
  To the best of my knowledge, this work is the first combination of the particle filter framework from the point cloud library, that uses feature point descriptor comparison for particle weighting.

8.2 Outlook

There are several starting points for future work arising from this thesis which should be pursued.
Promising extensions for CODAPT are mentioned to improve its desired characteristics for optical navigation tasks. To enhance its flexibility, generative model update and the use of object detections as a starting point for tracking can be implemented. Both would cancel the need for a pre-build object model. Further, a dynamic parameter update is suggested to account for different sized search areas in the particle filtering. CODAPT's robustness can be increased by implementing more extensive data association (i.e. allow multiple object hypotheses simultaneously) or an extension for a fully tracking-by-detection paradigm, to avoid tracking losses caused by the recursive tracking approach. Another important ability to improve is the runtime, by employing parallel GPU implementations of the particle filter or a faster point cloud intersection for k-d trees. The overall power for deployment in general applications can be increased by implementing multiple object tracking or evaluating a parameter refinement. To perform more extensive evaluations for CODAPT, the in the limitations stated IPS dataset wrapper is recommended.
Further research directions are the creation of more extensive 3D object datasets, to train general 3D object detectors based on deep neural networks. Having this, tracking-by-detection methods fully working in the 3D space can be investigated.
I recommend to first enhance the flexibility. In my opinion, the manual creation of object models in advance might be the highest inhibition threshold, to employ CODAPT in the field.

Bibliography

Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen (Dec. 2006). "Face Description with Local Binary Patterns: Application to Face Recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.12, pp. 2037–2041. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.244.

Arulampalam, M. Sanjeev et al. (2002). "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". In: *IEEE Transactions on signal processing* 50.2, pp. 174–188.

Asvadi, Alireza, Paulo Peixoto, and Urbano Nunes (2015). "Detection and tracking of moving objects using 2.5 d motion grids". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, pp. 788–793.

Aytar, Y. and A. Zisserman (2011). "Tabula Rasa: Model Transfer for Object Category Detection". In: *IEEE International Conference on Computer Vision*.

Azim, Asma and Olivier Aycard (2014). "Layer-based supervised classification of moving objects in outdoor dynamic environment using 3D laser scanner". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. IEEE, pp. 1408–1414.

Basgall, Paul L., Fred A. Kruse, and Richard C. Olsen (2014). "Comparison of lidar and stereo photogrammetric point clouds for change detection". In: *Laser Radar Technology and Applications XIX; and Atmospheric Propagation XI*. Vol. 9080. International Society for Optics and Photonics, 90800R.

Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool (2006). "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Ed. by Ales Leonardis, Horst Bischof, and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 404–417. ISBN: 978-3-540-33833-8.

Berkmann, Jens and Terry Caelli (1994). "Computation of surface geometry and segmentation using covariance techniques". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16.11, pp. 1114–1116.

Besl, Paul J. and Neil D. McKay (1992). "Method for registration of 3-D shapes". In: *Sensor Fusion IV: Control Paradigms and Data Structures*. Vol. 1611. International Society for Optics and Photonics, pp. 586–607.

Börner, Anko et al. (2017). "IPS – a vision aided navigation system". In: *Advanced Optical Technologies* 6.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Breitenstein, Michael D. et al. (2010). "Online multiperson tracking-by-detection from a single, uncalibrated camera". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9, pp. 1820–1833.

Brgfx (2019). *Set nature backgrounds, username: brgfx*. https://www.freepik.com (visited on 29.08.2019).

Britz, Denny (2015). *Understanding Convolutional Neural Networks for NLP*. http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/ (visited on 29.08.2019).

Broggi, Alberto et al. (2013). "A full-3D voxel-based dynamic obstacle detection for urban scenario using stereo vision". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, pp. 71–76.

Brown, Duane C. (1971). "Close-range camera calibration". In: *Photogramm. Eng* 37.8, pp. 855–866.

Burns, Don (1998–2019). *OpenSceneGraph*. `http://www.openscenegraph.org/`.

Bursuc, Andrei, Giorgos Tolias, and Hervé Jégou (2015). "Kernel local descriptors with implicit rotation matching". In: *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*. ACM, pp. 595–598.

Choi, Changhyun and Henrik I. Christensen (2013). "RGB-D object tracking: A particle filter approach on GPU". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1084–1091.

CIE, Commission Internationale de l'Éclairage (1977). "CIE Recommendations on Uniform Color Spaces, Color-Difference Equations, and Metric Color Terms". In: *Color Research & Application* 2.1, pp. 5–6. DOI: `10.1002/j.1520-6378.1977.tb00102.x`.

Clarkson, Kenneth L. and Peter W. Shor (1989). "Applications of random sampling in computational geometry, II". In: *Discrete & Computational Geometry* 4.5, pp. 387–421.

Comaniciu, Dorin, Visvanathan Ramesh, and Peter Meer (2003). "Kernel-based object tracking". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5, pp. 564–575.

Concha, David et al. (2018). "Performance evaluation of a 3D multi-view-based particle filter for visual object tracking using GPUs and multicore CPUs". In: *Journal of Real-Time Image Processing* 15.2, pp. 309–327.

Dai, Jifeng et al. (2016). "R-fcn: Object detection via region-based fully convolutional networks". In: *Advances in neural information processing systems*, pp. 379–387.

Dalal, Navneet and Bill Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. Washington, DC, USA: IEEE Computer Society, pp. 886–893. ISBN: 0-7695-2372-2. DOI: `10.1109/CVPR.2005.177`.

Deng, Jia et al. (2009). "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 248–255.

Dong, Jingming and Stefano Soatto (2015). "Domain-size pooling in local descriptors: DSP-SIFT". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5097–5106.

Doucet, Arnaud et al. (2000). "Rao-Blackwellised particle filtering for dynamic Bayesian networks". In: *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., pp. 176–183.

Durrant-Whyte, Hugh and Tim Bailey (2006). "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2, pp. 99–110.

Everingham, Mark et al. (2010). "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2, pp. 303–338.

Fischler, Martin A. and Robert C. Bolles (1981). "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6, pp. 381–395. ISSN: 00010782.

Foley, James D. et al. (1990). *Computer Graphics: Principles and Practice (2Nd Ed.)* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0-201-12110-7.

Fortmann, Thomas, Yaakov Bar-Shalom, and Molly Scheffe (1983). "Sonar tracking of multiple targets using joint probabilistic data association". In: *IEEE journal of Oceanic Engineering* 8.3, pp. 173–184.

Fox, Dieter (2002). "KLD-sampling: Adaptive particle filters". In: *Advances in neural information processing systems*, pp. 713–720.

Frossard, Davi and Raquel Urtasun (2018). "End-to-end Learning of Multi-sensor 3D Tracking by Detection". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 635–642.

Fu, Cheng-Yang et al. (2017). "Dssd: Deconvolutional single shot detector". In: *arXiv preprint arXiv:1701.06659*.

Fukushima, Kunihiko (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4, pp. 193–202.

Geiger, Andreas, Martin Lauer, et al. (2013). "3d traffic scene understanding from movable platforms". In: *IEEE transactions on pattern analysis and machine intelligence* 36.5, pp. 1012–1025.

Geiger, Andreas, Philip Lenz, Christoph Stiller, et al. (2013). "Vision meets robotics: The KITTI dataset". In: *The International Journal of Robotics Research* 32.11, pp. 1231–1237.

Geiger, Andreas, Philip Lenz, and Raquel Urtasun (2012). "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Georgakis, Georgios et al. (2017). "Synthesizing training data for object detection in indoor scenes". In: *arXiv preprint arXiv:1702.07836*.

Girao, Pedro et al. (2016). "3D Object Tracking in Driving Environment: a short review and a benchmark dataset". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, pp. 7–12.

Girshick, Ross (2015). "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448.

Girshick, Ross et al. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587.

Glasner, Daniel et al. (2011). "Aware object detection and pose estimation". In: *2011 International Conference on Computer Vision*. IEEE, pp. 1275–1282.

Godec, Martin et al. (2010). "On-line random naive bayes for tracking". In: *2010 20th International Conference on Pattern Recognition*. IEEE, pp. 3545–3548.

Grießbach, Denis et al. (2012). "IPS–a system for real-time navigation and 3D modeling". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 39, B5.

Guo, Yulan, Mohammed Bennamoun, et al. (2016). "A comprehensive performance evaluation of 3D local feature descriptors". In: *International Journal of Computer Vision* 116.1, pp. 66–89.

Guo, Yulan, Ferdous Ahmed Sohel, et al. (2013). "TriSI: A Distinctive Local Surface Descriptor for 3D Modeling and Object Recognition." In: *GRAPP/IVAPP*, pp. 86–93.

Guo, Yulan, Ferdous Sohel, et al. (2015). "A novel local surface feature for 3D object recognition under clutter and occlusion". In: *Information Sciences* 293, pp. 196–213.

Hamid Rezatofighi, Seyed et al. (2015). "Joint probabilistic data association revisited". In: *Proceedings of the IEEE international conference on computer vision*, pp. 3047–3055.

Han, Jungong et al. (2013). "Enhanced computer vision with microsoft kinect sensor: A review". In: *IEEE transactions on cybernetics* 43.5, pp. 1318–1334.

Hana, Xian-Feng et al. (2018). "A comprehensive review of 3D point cloud descriptors". In: *arXiv preprint arXiv:1802.02297*.

Harris, C.G. and J.M. Pike (1988). "3D positional integration from image sequences". In: *Image and Vision Computing* 6.2, pp. 87–90.

Hartley, Richard and Andrew Zisserman (2003). *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press. ISBN: 0521540518.

Harwin, Steve and Arko Lucieer (2012). "Assessing the accuracy of georeferenced point clouds produced via multi-view stereopsis from unmanned aerial vehicle (UAV) imagery". In: *Remote Sensing* 4.6, pp. 1573–1599.

He, Kaiming et al. (2017). "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969.

Hirschmüller, Heiko (2008). "Stereo processing by semiglobal matching and mutual information". In: *IEEE Transactions on pattern analysis and machine intelligence* 30.2, pp. 328–341.

Hirschmüller, Heiko, Maximilian Buder, and Ines Ernst (2012). "Memory efficient semi-global matching". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 3, pp. 371–376.

Huang, Haibin et al. (2018). "Learning local shape descriptors from part correspondences with multiview convolutional networks". In: *ACM Transactions on Graphics (TOG)* 37.1, p. 6.

Hubel, David H. and Torsten N. Wiesel (1968). "Receptive fields and functional architecture of monkey striate cortex". In: *The Journal of physiology* 195.1, pp. 215–243.

Irmisch, Patrick (2017). "Camera-based distance estimation for autonomous vehicles". MA thesis. Technical University Berlin. Faculty of Electrical Engineering, Computer Science. Department of Computer Vision, and Remote Sensing.

Irmisch, Patrick et al. (2019). "Simulation Framework for a visual-inertial navigation system". In: *IEEE International Conference on Image Processing (ICIP)*.

Jepson, Allan D., David J. Fleet, and Thomas F. El-Maraghi (2003). "Robust online appearance models for visual tracking". In: *IEEE transactions on pattern analysis and machine intelligence* 25.10, pp. 1296–1311.

Johnson, Andrew E. and Martial Hebert (1999). "Using spin images for efficient object recognition in cluttered 3D scenes". In: *IEEE Transactions on pattern analysis and machine intelligence* 21.5, pp. 433–449.

Khan, Zia, Tucker Balch, and Frank Dellaert (2005). "MCMC-based particle filtering for tracking a variable number of interacting targets". In: *IEEE transactions on pattern analysis and machine intelligence* 27.11, pp. 1805–1819.

Kitagawa, Genshiro (1987). "Non-gaussian state—space modeling of nonstationary time series". In: *Journal of the American statistical association* 82.400, pp. 1032–1041.

Klasing, Klaas et al. (2009). "Comparison of surface normal estimation methods for range sensing applications". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE, pp. 3206–3211.

Knopp, Jan et al. (2010). "Hough transform and 3D SURF for robust three dimensional classification". In: *European Conference on Computer Vision*. Springer, pp. 589–602.

Knuth, Donald E. (1998). *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc. ISBN: 0-201-89685-0.

Krainin, Michael et al. (2011). "Manipulator and object tracking for in-hand 3D object modeling". In: *The International Journal of Robotics Research* 30.11, pp. 1311–1327.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Lampert, C.H., M.B. Blaschko, and T. Hofmann (June 2008). "Beyond Sliding Windows: Object Localization by Efficient Subwindow Search". In: *CVPR 2008*. Best paper award. Max-Planck-Gesellschaft. Los Alamitos, CA, USA: IEEE Computer Society, pp. 1–8.

Leal-Taixé, Laura et al. (2015). "Motchallenge 2015: Towards a benchmark for multi-target tracking". In: *arXiv preprint arXiv:1504.01942*.

Lei, Yun, Xiaoqing Ding, and Shengjin Wang (2008). "Visual tracker using sequential bayesian learning: Discriminative, generative, and hybrid". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.6, pp. 1578–1591.

Leibe, Bastian, Nico Cornelis, et al. (2007). "Dynamic 3d scene analysis from a moving vehicle". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8.

Leibe, Bastian, Konrad Schindler, et al. (2008). "Coupled object detection and tracking from static cameras and moving vehicles". In: *IEEE transactions on pattern analysis and machine intelligence* 30.10, pp. 1683–1698.

Lepetit, Vincent, Pascal Fua, et al. (2005). "Monocular model-based 3d tracking of rigid objects: A survey". In: *Foundations and Trends in Computer Graphics and Vision* 1.1, pp. 1–89.

Li, Peixia et al. (2018). "Deep visual tracking: Review and experimental comparison". In: *Pattern Recognition* 76, pp. 323–338.

Li, Yuan et al. (2008). "Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.10, pp. 1728–1740.

Lin, Tsung-Yi, Priya Goyal, et al. (2017). "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Lin, Tsung-Yi, Michael Maire, et al. (2014). "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer, pp. 740–755.

Linsen, Lars (2001). *Point cloud representation*. Univ., Fak. für Informatik, Bibliothek.

Liu, Li et al. (2018). "Deep learning for generic object detection: A survey". In: *arXiv preprint arXiv:1809.02165*.

Liu, Wei et al. (2016). "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer, pp. 21–37.

Longuet-Higgins, Christopher (1981). "A computer algorithm for reconstructing a scene from two projections". In: *Nature* 293.5828, p. 133.

Lowe, David G. (1999). "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*. ICCV '99. Washington, DC, USA: IEEE Computer Society, pp. 1150–1157. ISBN: 0-7695-0164-8.

Mariottini, G.L., Domenico Prattichizzo, and Giuseppe Oriolo (Jan. 2004). "Epipole-based visual servoing for nonholonomic mobile robots". In: vol. 1, 497–503 Vol.1. ISBN: 0-7803-8232-3. DOI: 10.1109/ROBOT.2004.1307198.

Mian, Ajmal, Mohammed Bennamoun, and Robyn Owens (2010). "On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes". In: *International Journal of Computer Vision* 89.2-3, pp. 348–361.

Milan, Anton et al. (2017). "Online multi-target tracking using recurrent neural networks". In: *Thirty-First AAAI Conference on Artificial Intelligence*.

Miyasaka, Takeo, Yoshihiro Ohama, and Yoshiki Ninomiya (2009). "Ego-motion estimation and moving object tracking using multi-layer lidar". In: *2009 IEEE intelligent vehicles symposium*. IEEE, pp. 151–156.

Moulon, Pierre, Pascal Monasse, Renaud Marlet, et al. (2012). *OpenMVG. An Open Multiple View Geometry library.* `https://github.com/openMVG/openMVG/blob/develop/docs/sphinx/rst/openMVG/cameras/cameras.rst` (visited on 29.08.2019).

Mouragnon, Etienne et al. (2006). "Real time localization and 3d reconstruction". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. IEEE, pp. 363–370.

Nistér, David (2004). "An efficient solution to the five-point relative pose problem". In: *IEEE transactions on pattern analysis and machine intelligence* 26.6, pp. 0756–777.

Nister, David, Oleg Naroditsky, and James Bergen (2004). "Visual odometry". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. Ieee, pp. I–I.

Osep, Aljoša et al. (2017). "Combined image-and world-space tracking in traffic scenes". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1988–1995.

Ošep, Aljoša et al. (2016). "Multi-scale object candidates for generic object tracking in street scenes". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3180–3187.

Ozuysal, Mustafa et al. (2010). "Fast keypoint recognition using random ferns". In: *IEEE transactions on pattern analysis and machine intelligence* 32.3, pp. 448–461.

Pandey, Gaurav, James R. McBride, and Ryan M. Eustice (2011). "Ford campus vision and lidar data set". In: *The International Journal of Robotics Research* 30.13, pp. 1543–1552.

Papadopoulos, Dim P. et al. (2014). "Training object class detectors from eye tracking data". In: *European conference on computer vision*. Springer, pp. 361–376.

Pauly, Mark, Markus Gross, and Leif P Kobbelt (2002). "Efficient simplification of point-sampled surfaces". In: *Proceedings of the conference on Visualization'02*. IEEE Computer Society, pp. 163–170.

Pauwels, Karl and Danica Kragic (2015). "Simtrack: A simulation-based framework for scalable real-time object pose detection and tracking". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1300–1307.

Pearson, Karl (1901). "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11, pp. 559–572.

Qi, Charles R., Wei Liu, et al. (2018). "Frustum pointnets for 3d object detection from rgb-d data". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 918–927.

Qi, Charles R., Li Yi, et al. (2017). "Pointnet++: Deep hierarchical feature learning on point sets in a metric space". In: *Advances in neural information processing systems*, pp. 5099–5108.

Qi, Xianbiao et al. (2014). "Pairwise rotation invariant co-occurrence local binary pattern". In: *IEEE transactions on pattern analysis and machine intelligence* 36.11, pp. 2199–2213.

Redmon, Joseph (2013–2016). *Darknet: Open Source Neural Networks in C*. `http://pjreddie.com/darknet/` (visited on 29.08.2019).

Redmon, Joseph, Santosh Divvala, et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271.

— (2018). "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767*.

Reid, Donald (1979). "An algorithm for tracking multiple targets". In: *IEEE transactions on Automatic Control* 24.6, pp. 843–854.

Ren, Shaoqing et al. (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Rosten, Edward and Tom Drummond (2006). "Machine Learning for High-speed Corner Detection". In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*. ECCV'06. Graz, Austria: Springer-Verlag, pp. 430–443. ISBN: 3-540-33832-2, 978-3-540-33832-1.

Rozantsev, Artem, Vincent Lepetit, and Pascal Fua (2015). "On rendering synthetic images for training an object detector". In: *Computer Vision and Image Understanding* 137, pp. 24–37.

Russakovsky, Olga et al. (2015). "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3, pp. 211–252.

Rusu, R. B. and S. Cousins (May 2011). "3D is here: Point Cloud Library (PCL)". In: *2011 IEEE International Conference on Robotics and Automation*, pp. 1–4. DOI: 10.1109/ICRA.2011.5980567.

Rusu, Radu Bogdan (2010). "Semantic 3d object maps for everyday manipulation in human living environments". In: *KI-Künstliche Intelligenz* 24.4, pp. 345–348.

Rusu, Radu Bogdan, Nico Blodow, et al. (2008). "Aligning point cloud views using persistent feature histograms". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 3384–3391.

Rusu, Radu Bogdan, Zoltan Csaba Marton, et al. (2008). "Learning informative point classes for the acquisition of object model maps". In: *2008 10th International Conference on Control, Automation, Robotics and Vision*. IEEE, pp. 643–650.

Salti, Samuele, Federico Tombari, and Luigi Di Stefano (2014). "SHOT: Unique signatures of histograms for surface and texture description". In: *Computer Vision and Image Understanding* 125, pp. 251–264.

Sande, K.E.A. van de et al. (2011). "Segmentation As Selective Search for Object Recognition". In: *IEEE International Conference on Computer Vision*.

Scaramuzza, Davide (2015). *Towards Robust and Safe Autonomous Drones*. `https://de.slideshare.net/SERENEWorkshop/towards-robust-and-safe-autonomous-drones` (visited on 29.08.2019).

Scaramuzza, Davide and Friedrich Fraundorfer (2011). "Visual odometry [tutorial]". In: *IEEE robotics & automation magazine* 18.4, pp. 80–92.

Scharstein, D. and R. Szeliski (2019). *Middlebury Stereo Evaluation - Version 3*. `http://vision.middlebury.edu/stereo/eval3/` (visited on 29.08.2019).

Sermanet, Pierre et al. (2013). "Overfeat: Integrated recognition, localization and detection using convolutional networks". In: *http://arxiv.org/abs/1312.6229*.

Shaifee, Mohammad Javad et al. (2017). "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video". In: *Journal of Computational Vision and Imaging Systems* 3.1.

Shen, Chunhua, Junae Kim, and Hanzi Wang (2010). "Generalized kernel-based visual tracking". In: *IEEE Transactions on Circuits and Systems for Video Technology* 20.1, pp. 119–130.

Simo-Serra, Edgar et al. (2015). "Discriminative learning of deep convolutional feature point descriptors". In: *Proceedings of the IEEE International Conference on Computer Vision*, pp. 118–126.

Stein, Ayal, Eran Geva, and Jihad El-Sana (2012). "CudaHull: Fast parallel 3D convex hull on the GPU". In: *Computers & Graphics* 36.4, pp. 265–271.

Stein, Fridtjof and Gérard Medioni (1992). "Structural indexing: Efficient 3-D object recognition". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 2, pp. 125–145.

Sturm, Jürgen et al. (2012). "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 573–580.

Sun, Jian, Maks Ovsjanikov, and Leonidas Guibas (2009). "A concise and provably informative multi-scale signature based on heat diffusion". In: *Computer graphics forum*. Vol. 28. 5. Wiley Online Library, pp. 1383–1392.

Szeliski, Richard (2010). *Computer Vision: Algorithms and Applications*. 1st. Berlin, Heidelberg: Springer-Verlag. ISBN: 1848829345, 9781848829343.

Teng, Ervin, Rui Huang, and Bob Iannucci (2018). "ClickBAIT-v2: Training an Object Detector in Real-Time". In: *arXiv preprint arXiv:1803.10358*.

Teuliere, Céline, Eric Marchand, and Laurent Eck (2015). "3-D model-based tracking for UAV indoor localization". In: *IEEE Transactions on cybernetics* 45.5, pp. 869–879.

Tombari, Federico, Samuele Salti, and Luigi Di Stefano (2010). "Unique signatures of histograms for local surface description". In: *European conference on computer vision*. Springer, pp. 356–369.

— (2011). "A combined texture-shape descriptor for enhanced 3D feature matching". In: *2011 18th IEEE international conference on image processing*. IEEE, pp. 809–812.

— (2013). "Performance evaluation of 3D keypoint detectors". In: *International Journal of Computer Vision* 102.1-3, pp. 198–220.

Tuzel, Oncel, Fatih Porikli, and Peter Meer (2006). "Region Covariance: A Fast Descriptor for Detection And Classification". In: *In Proc. 9th European Conf. on Computer Vision*, pp. 589–600.

Tyagi, Ambrish et al. (2007). "Kernel-based 3d tracking". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8.

Tzutalin (2015). *LabelImg*. `https://github.com/tzutalin/labelImg` (visited on 29.08.2019).

Unnikrishnan, Ranjith and Martial Hebert (2008). "Multi-scale interest regions from unorganized point clouds". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, pp. 1–8.

Vatavu, Andrei, Radu Danescu, and Sergiu Nedevschi (2015). "Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters". In: *IEEE Transactions on Intelligent Transportation Systems* 16.1, pp. 498–511.

Viola, P. and M. Jones (2001). "Rapid object detection using a boosted cascade of simple features". In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Vol. 1, I-511-I–518 vol.1. DOI: 10.1109/CVPR.2001.990517.

Wang, Qing et al. (2012). "Object tracking via partial least squares analysis". In: *IEEE Transactions on Image Processing* 21.10, pp. 4454–4465.

Watt, Alan (1993). *3D Computer Graphics*. 2nd. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201631865.

Wu, Y., J. Lim, and M. Yang (2013). "Online Object Tracking: A Benchmark". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2411–2418. DOI: 10.1109/CVPR.2013.312.

Wuest, Harald and Didier Stricker (2007). "Tracking of industrial objects by using cad models". In: *JVRB-Journal of Virtual Reality and Broadcasting* 4.1.

Yang, Fan, Huchuan Lu, and Yen-Wei Chen (2010). "Bag of features tracking". In: *2010 20th International Conference on Pattern Recognition*. IEEE, pp. 153–156.

Yang, Hanxuan et al. (2011). "Recent advances and trends in visual tracking: A review". In: *Neurocomputing* 74.18, pp. 3823–3831.

Yi, Sang-ri and Junho Song (2018). "Particle Filter Based Monitoring and Prediction of Spatiotemporal Corrosion Using Successive Measurements of Structural Responses". In: *Sensors* 18.11, p. 3909.

Yilmaz, Alper, Omar Javed, and Mubarak Shah (2006). "Object tracking: A survey". In: *Acm computing surveys (CSUR)* 38.4, p. 13.

Yoon, Kwangjin et al. (2019). "Data Association for Multi-Object Tracking via Deep Neural Networks". In: *Sensors* 19.3, p. 559.

Yousif, Khalid, Alireza Bab-Hadiashar, and Reza Hoseinnezhad (2014). "Real-time RGB-D registration and mapping in texture-less environments using ranked order statistics". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2654–2660.

Zarzar, Jesus, Silvio Giancola, and Bernard Ghanem (2019). "Efficient Tracking Proposals using 2D-3D Siamese Networks on LIDAR". In: *arXiv preprint arXiv:1903.10168*.

Zeisl, Bernhard et al. (2010). "On-line semi-supervised multiple-instance boosting". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1879–1879.

Zeng, Andy et al. (2017). "3dmatch: Learning local geometric descriptors from rgb-d reconstructions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1802–1811.

Zhang, Xin et al. (2013). "Object class detection: A survey". In: *ACM Computing Surveys (CSUR)* 46.1, p. 10.

Zhong, Yu (2009). "Intrinsic shape signatures: A shape descriptor for 3d object recognition". In: *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, pp. 689–696.

Appendix

The appendix provides further material to complete the evaluation part of this work. First, the simulated and recorded datasets are specified. Afterwards, the applied configurations for the evaluation are listed, together with the used hardware. Furthermore, more extensive evaluation results are presented. Finally, the used external software frameworks are provided.

A - Datasets

This appendix lists dataset (Table A.1) and object class (Table A.2) attributes that are chosen for this work. Moreover, schematic overviews of the used real and simulated datasets are provided. The overviews are not to scale but approximated by the rule of thumb. The Table A.3 contains the legend for the used symbolism.

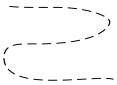
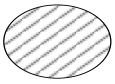
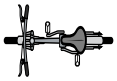
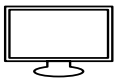
Table A.1: List of the dataset attributes.

Attribute	**Description**
Background Clutter	The background in the viewing range of the tracked object has similar colors or textures.
Illumination Variation	The illumination of the tracked object is changed, which leads to a pixel intensity change of at least 20 values.
Occlusion	The tracked object is partially or fully occluded.
Out-of-View	Tracking object parts leave the viewing range.
Rotation	The camera is rotated while recording the objects.

Table A.2: Object property specification.

Property	**Definition**	**Low (+)**	**Medium (++)**	**Large (+++)**
Size	Object size in dimension with the largest object extend.	$\leq$ 1m	otherwise	$\geq$ 3m
Color	Averaged object pixel intensity.	$\leq$ 85	otherwise	$\geq$ 170
Textureness	The absence of homogeneous colored areas (in % of hom. areas). Hom. areas are considered to be $\geq 2dm^2$ large.	$\geq$ 50%	otherwise	$\leq$ 20%
Curviness	The averaged curviness of the object surface – estimated value with regard to the amount of straight areas and corners.	-	-	-

Table A.3: Icon legend for the schematic dataset overviews.

Icon	Description	Size	Color	Texture	Curviness
 Start/End	Origin of the local coordinate system. Start and end point of the IPS session, also considered as a checkpoint.	-	-	-	-
 Checkpoint	Parking point of the IPS.	-	-	-	-
	Objects marked with a red dot belong to the objects chosen for tracking.	-	-	-	-
	Trajectory of the IPS session.	-	-	-	-
	Shadow in the environment. The recorded illumination change from a light area to shadow area for a pixel is ≤ -20.	-	-	-	-
	Objects that lead to a obstructed view between camera and tracking object. From left to right: tree, house, table.	-	-	-	-
	Tracking object car.	+++	+++	+	+++
	Tracking object motorbike.	++	+	++	+++
	Tracking object bicycle.	++	+	+++	+++
	Tracking object laptop.	+	++	+	+
	Tracking object monitor.	+	+	+	+
	Tracking object bottle.	+	+++	+	+++
	Tracking object office chair.	++	+	+	++
	Tracking object wood chair.	++	++	+++	+

A.1 - Simulated

A.1.1 - Desert

The desert setup represents the simulated wasteland dataset mapped in the Figures A.1.1.1. The Figure A.1.1.2 shows exemplary images. The scene contains sand textures on a wavily floor and desert typical, sparse disposed objects like cacti, dry trees and rocks. The background consist of a dark sky and clouds. The IPS sessions lasts around 760 seconds and contains 345.2 walked meters.

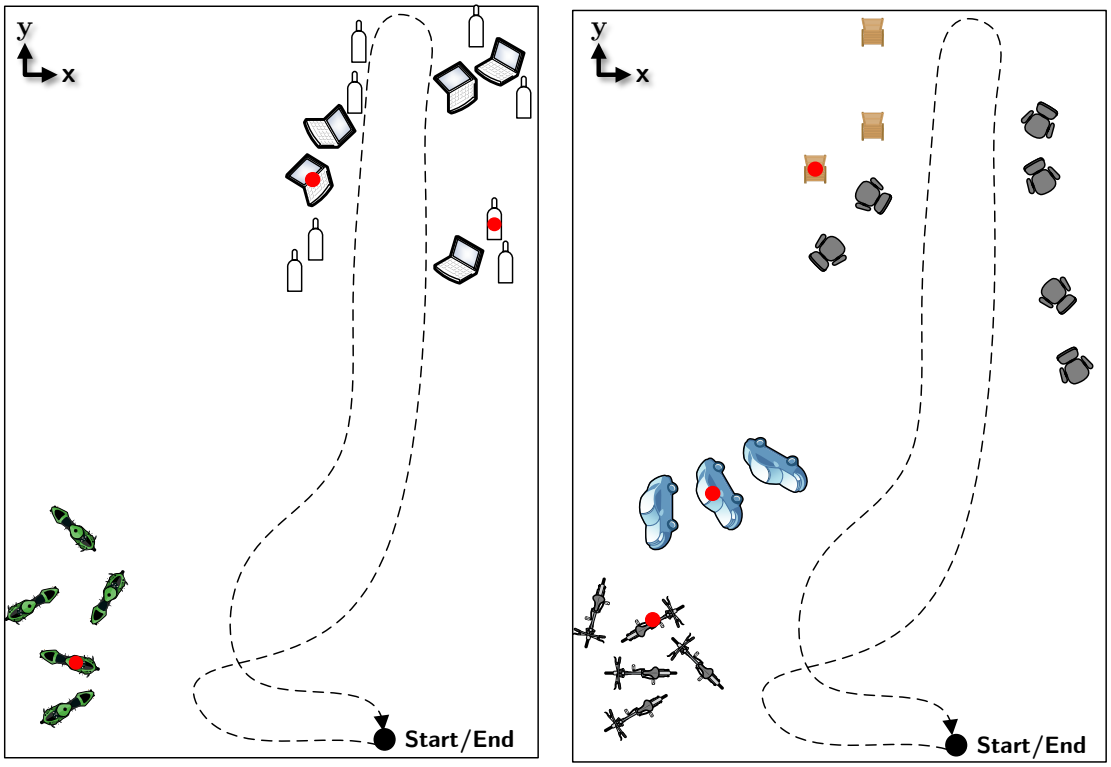
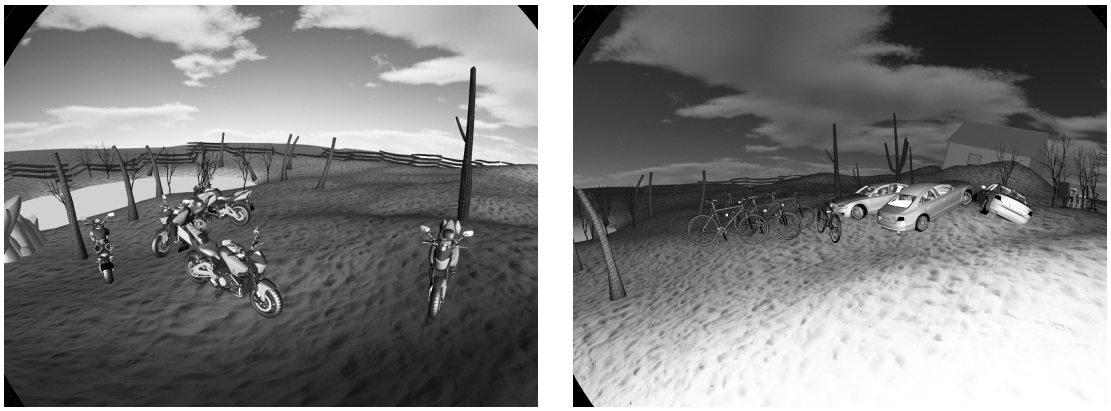


Figure A.1.1.1: Simulated desert dataset maps.



(a) Without illumination changes.

(b) With illumination changes.

Figure A.1.1.2: Exemplary left camera images from the desert dataset (increased brightness for better visualization).

A.1.2 - Hexenhaeuser

This setup called "Hexenhaeuser" (German for witch houses) represents the outdoor urban area dataset and is mapped in the Figures A.1.2.1. The Figure A.1.2.2 shows exemplary images. It represents a village in a wood, whereby lodges are located in the main area and the background is textured as vegetation. The IPS session has around 130 walked meters and lasts 360 seconds. The scene is provided by the DLR.

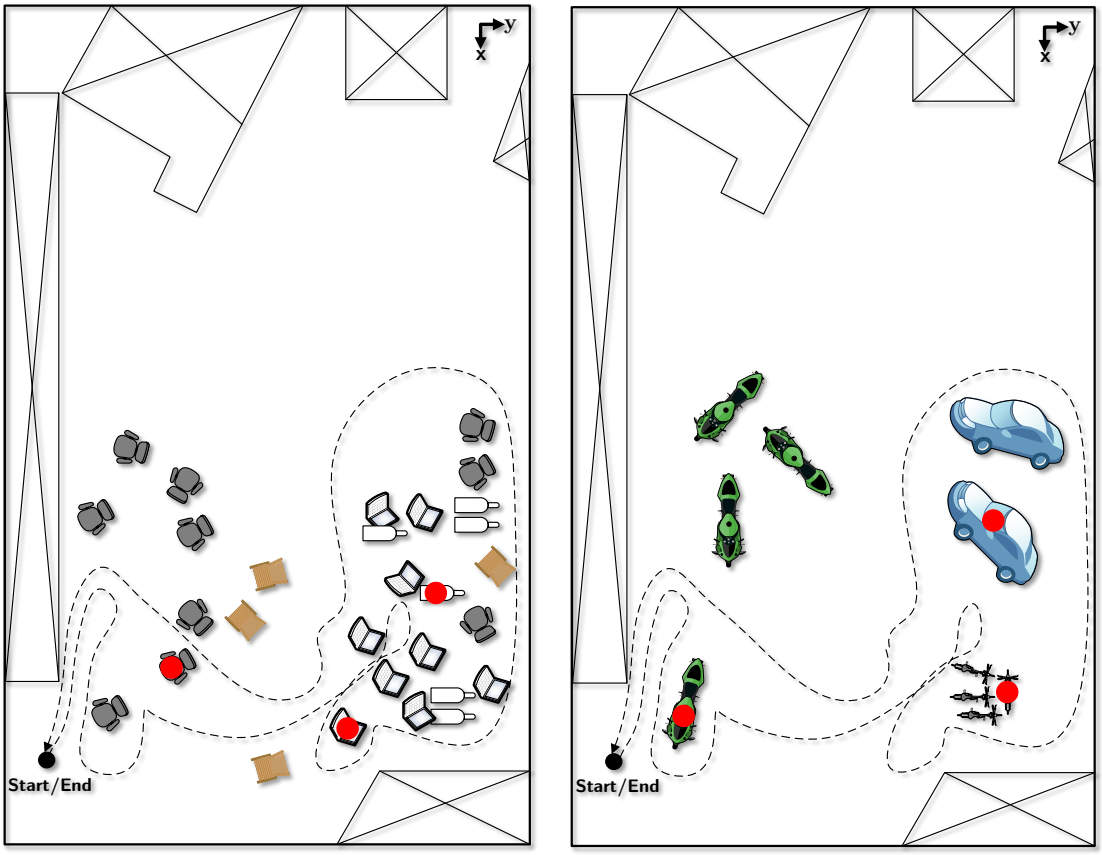
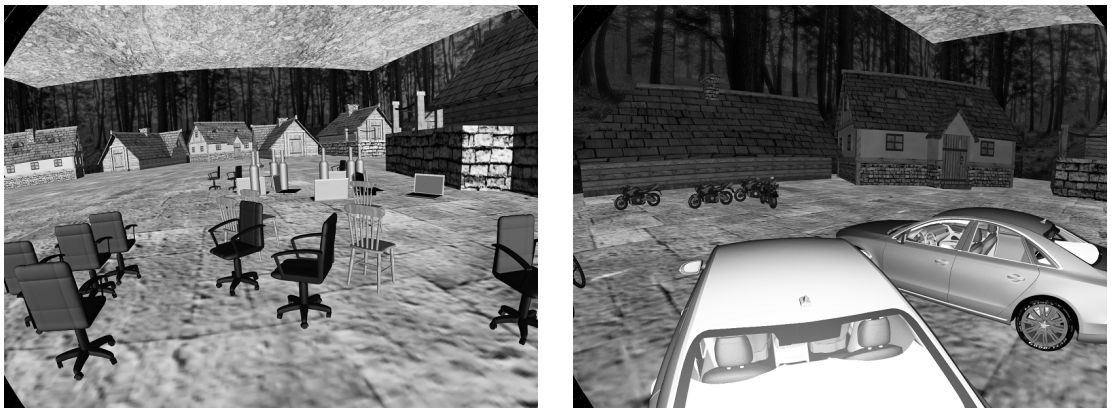


Figure A.1.2.1: Simulated Hexenhaeuser dataset maps.



(a) Without illumination changes.

(b) With illumination changes.

Figure A.1.2.2: Exemplary left camera images from the Hexenhaeuser dataset (increased brightness for better visualization).

A.1.3 - Office

This office dataset mapped in Figure A.1.3.1 represents the simulated indoor urban scenario. The Figure A.1.3.2 shows exemplary images. The dataset consists of a corridor decorated by posters, a carped on the floor, white walls, overhead lights and mainly even, homogenous textured surfaces – actually it is a replication of a tract of the DLR. The IPS session lasts around 380 seconds and contains 55.2 walked meters. The office scene is provided by the DLR.

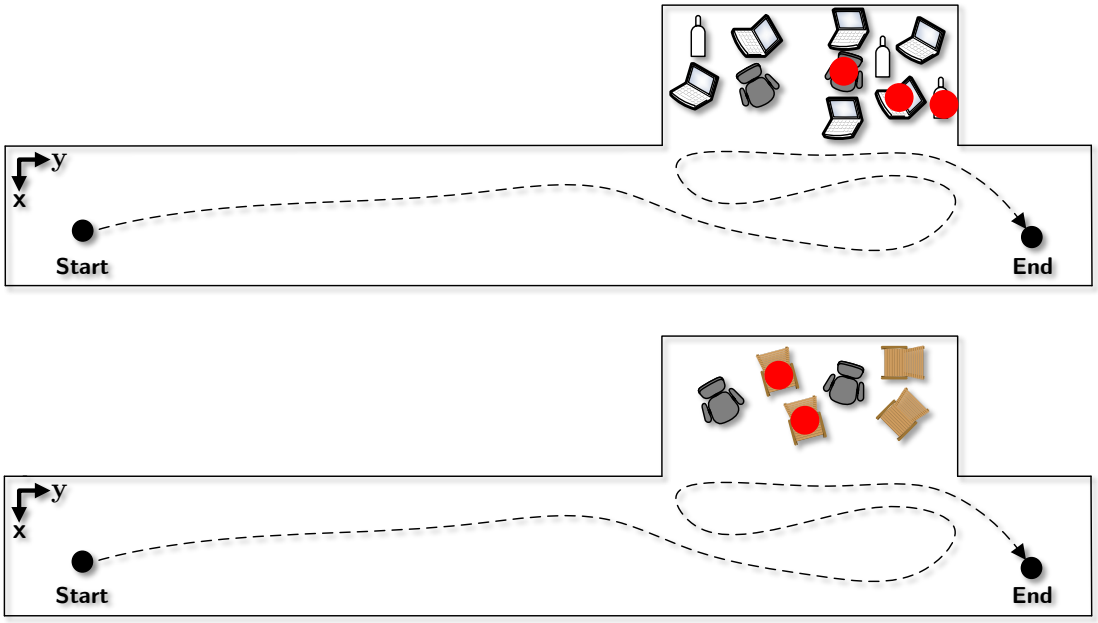
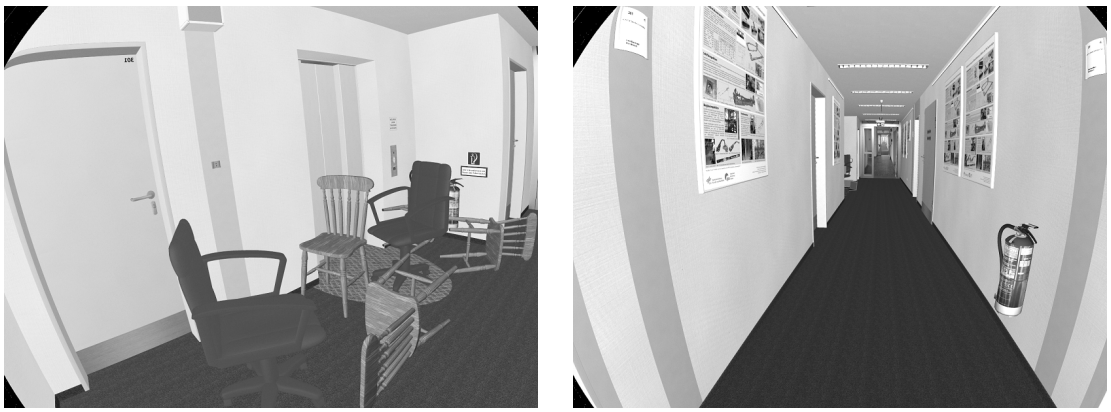


Figure A.1.3.1: Simulated office dataset maps.



(a) Without illumination changes.

(b) With illumination changes.

Figure A.1.3.2: Exemplary left camera images from the simulated office dataset (increased brightness for better visualization).

A.2 - Real World

A.2.1 - Outdoor

This dataset (Figure A.2.1.1) provides a real-world outdoor urban area scene with high textured grass, bushes and trees. Additionally, a parking area for small vehicles is covered. The trajectory is smoothed, the IPS session lasts about 253 seconds and contains 89.6 walked meters. The Figure A.2.1.2 shows an exemplary image.

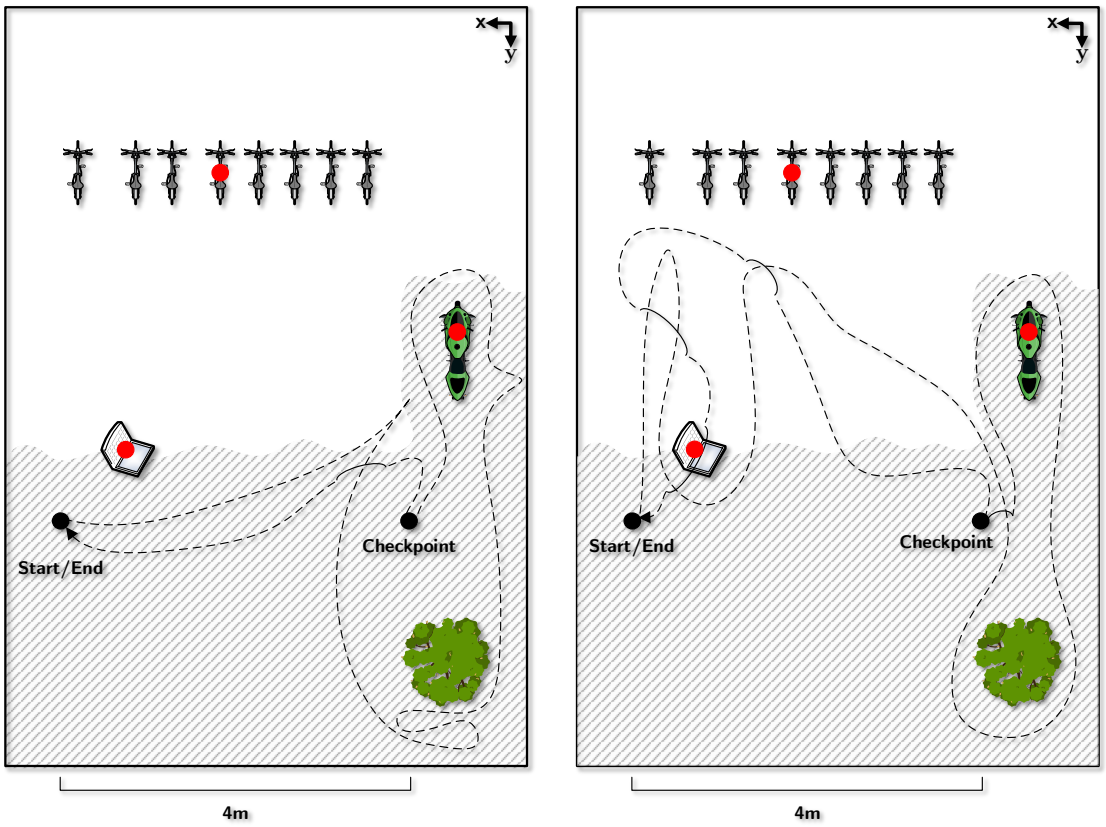


Figure A.2.1.1: Real outdoor dataset maps.

Figure A.2.1.2: Exemplary left camera image from the backdoor dataset (increased brightness for better visualization).

A.2.2 - Parking Lot

The parking lot dataset mapped in Figure A.2.2.1 represents a wasteland area with crushed stones on the floor, providing irregular clutter and parked cars. A shadow area creates illumination changes. The trajectory is smoothed, the IPS session lasts about 275 seconds and contains 109.3 walked meters. The Figure A.2.2.2 shows an exemplary image.

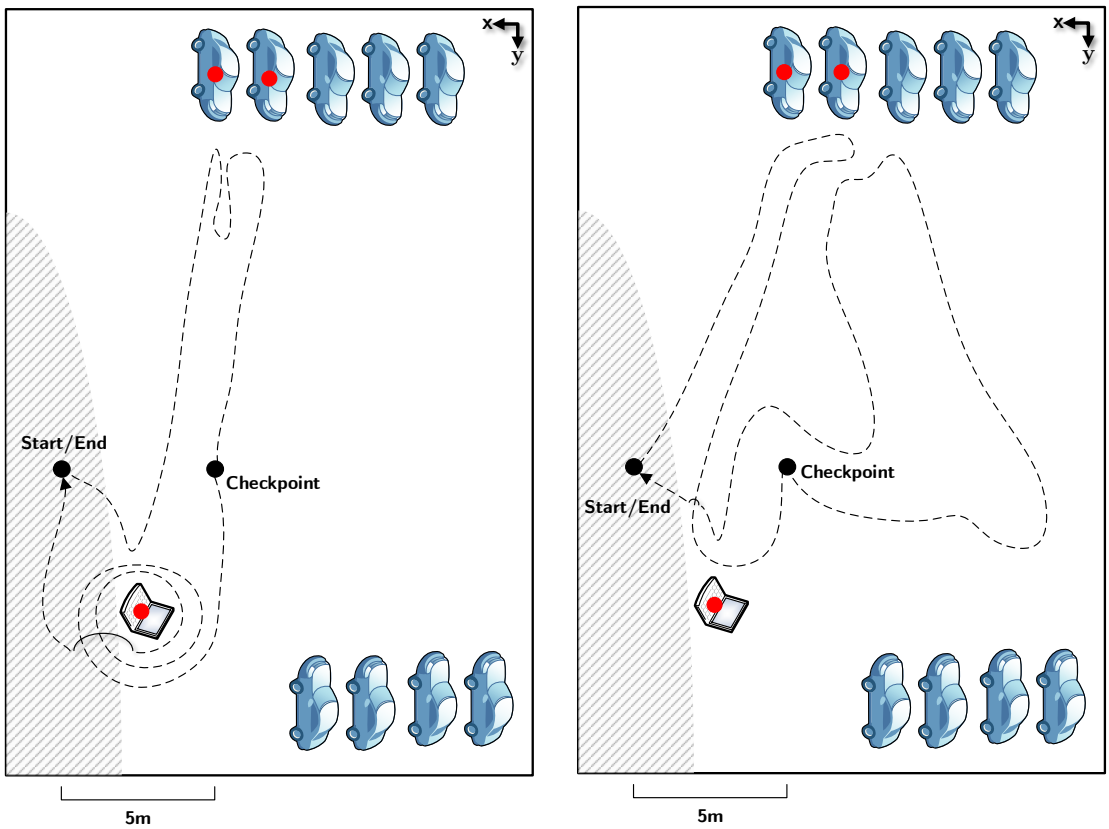


Figure A.2.2.1: Real parking lot dataset maps.

Figure A.2.2.2: Exemplary left camera image from the parking lot dataset (increased brightness for better visualization).

A.2.3 - Office

The real-world office dataset (Figure A.2.3.1) creates a similar situation as in the related simulation dataset. In contrast, in this dataset the illumination is changed repeatedly during the IPS session, by switching the overhead lights on and off. The trajectory is smoothed, the IPS session lasts about 311 seconds and contains 79.2 walked meters. The Figure A.2.3.2 shows an exemplary image.

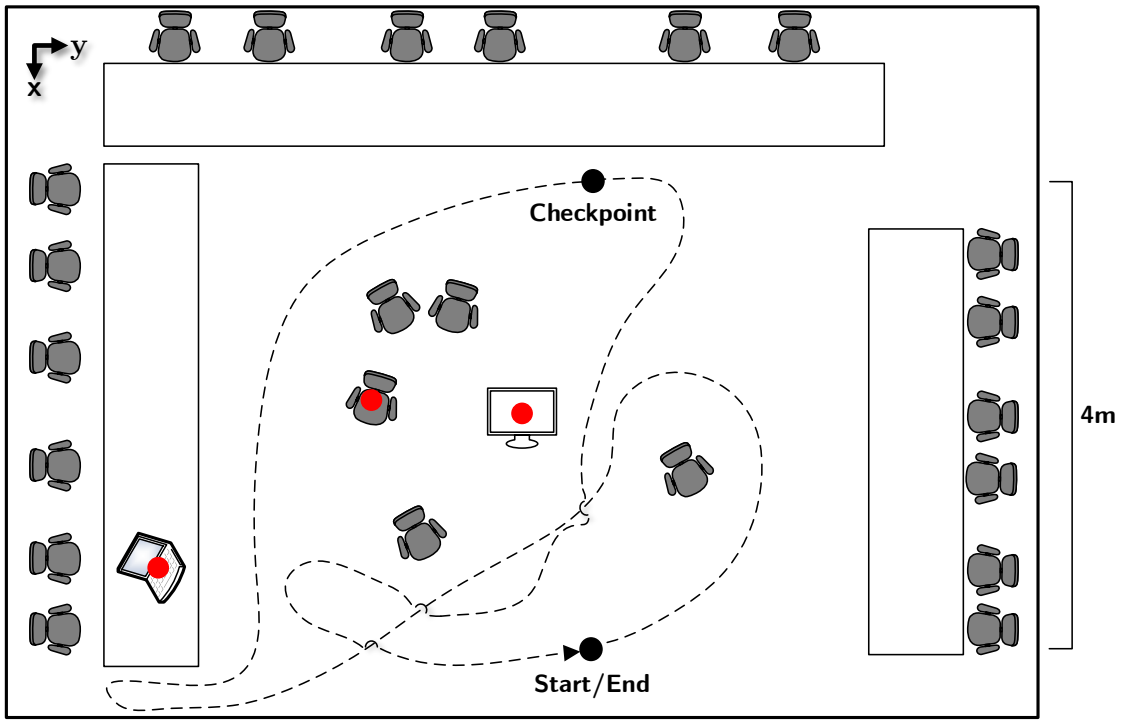


Figure A.2.3.1: Real office dataset map.

Figure A.2.3.2: Exemplary left camera image from the parking lot dataset (increased brightness for better visualization).

A.3 - Trajectory Smoothing

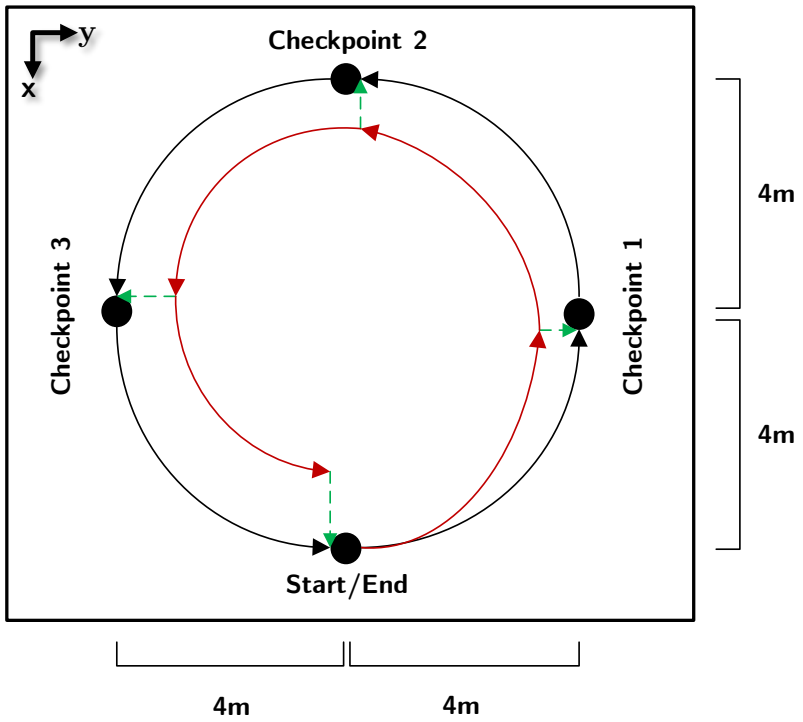


Figure A.3.1: Illustration of the trajectory smoothing approach. The black line depicts the ground truth trajectory, the red one the real, error-prone trajectory and the dashed green line the trajectory smoothing at the checkpoints.

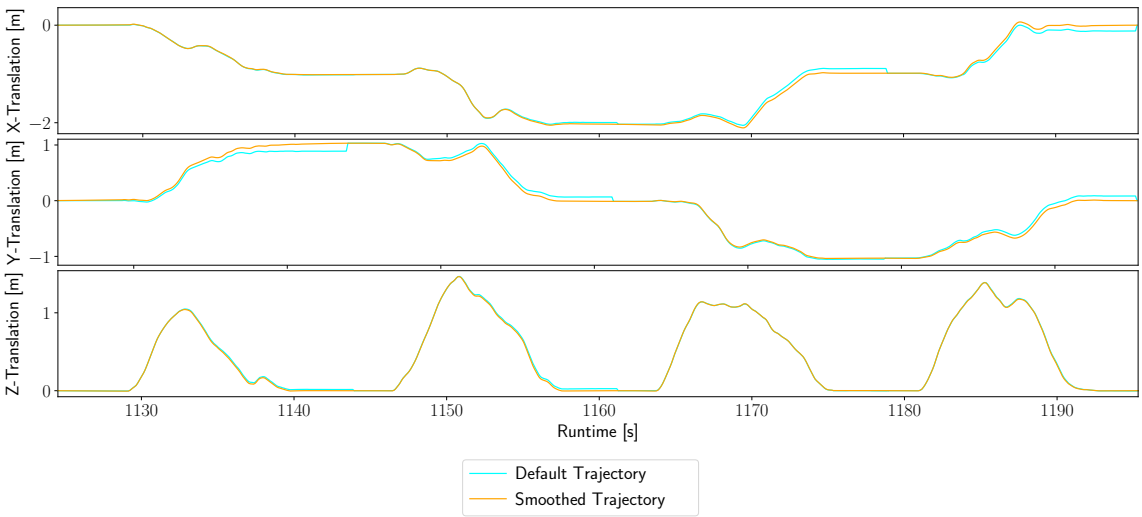


Figure A.3.2: Illustration of an exemplary trajectory smoothing with two checkpoints. The plots illustrate the IPS translation over time for the x, y and z directions, respectively.

The trajectory smoothing approach tackles the translational error of the IPS, when estimating a trajectory during an IPS session. The procedure exploits sized checkpoints, where the IPS is parked shortly. To not influence the orientation during the procedure, the IPS orientation has to be the same at each checkpoint. This is ensured here by using a square tool. Beforehand, the distances between the checkpoints and the local coordinates, respectively, have to be planned and determined. Having the local coordinates, the target-position of the IPS while parking is known. The respective difference between the target-position and the actual position of the IPS results in the translational closed-loop error between two checkpoints. In the trajectory smoothing, this error is reduced towards zero. As a result, the actual start and end point of the local trajectory between two checkpoints are similar to the ground truth ones and this trajectory is moved closer to the ground truth trajectory (considering only the translation values). The Figure A.3.1 illustrates the process – the distances between the checkpoints are exemplary and can be chosen arbitrary. The Figure A.3.2 shows an example application of the trajectory smoothing method with two checkpoints. The first checkpoint is applied at 1163 s and the second one at 1198 s. For the narrowed IPS sessions of this work, a significant difference between two and four checkpoints could not be determined – therefore, all trajectory smoothings are performed with two checkpoints. The trajectory smoothing approach is created and provided by the DLR.

B - Configurations

B.1 - IPS Configuration

The following tables specify the used IPS setups for all real world and simulated experiments. The Table B.1.1 describes the imaging specification for all simulated and real cameras, Table B.1.2 the radiometric calibrations of both cameras, Table B.1.3 interior geometric calibrations of the respective cameras, Table B.1.4 the stereo-camera geometric calibrations and Table B.1.5 the geometric calibrations between the left camera and the IMU. The last Tables B.1.6 and B.1.7 consist the parameters for the SGM algorithm and the point cloud creation, respectively. All listed parameters are provided by the DLR.

Table B.1.1: Camera imagining specification. Parameters from left to right: frame rate, spatial resolution, radiometric resolution and the number of channels.

Frame Rate $[Hz]$	**Spat. Res.** $[(px)^2]$	**Rad. Res.** $[Bit]$	**#(Channels)**
10	1360x1024	8	1

Table B.1.2: Radiometric camera calibrations. CCD-Noise-DN name the noise for the digital numbers of a CCD element.

	CCD-Noise-DN	**Gain**
$\boldsymbol{C_{real}}$	0.05	60.0
$\boldsymbol{C'_{real}}$	0.05	60.0
$\boldsymbol{C_{sim}}$	0.2658	59.1944
$\boldsymbol{C'_{sim}}$	0.2658	59.1944

Table B.1.3: Geometric interior camera calibrations. $k1, k2, k3$ denote the radial parameters and $p1, p2$ the tangential parameters from the lens distortion model of (Brown, 1971). σ describes the calibration uncertainty. All other parameters are named as in Figure 3.1.

	u(c) $[px]$	**v(c)** $[px]$	**f** $[px]$	**k1**	**k2**	**k3**	**p1**	**p2**
$\boldsymbol{C_{real}}$	710.62	545.52	777.14	-0.26822	0.13302	-0.03806	0.0	0.0
$\boldsymbol{C'_{real}}$	680.74	541.97	773.73	-0.26148	0.12073	-0.03079	0.0	0.0
$\boldsymbol{\sigma_{real}}$	0.3	0.3	0.2	0.0	0.0	0.0	0.0	0.0
$\boldsymbol{C_{sim}}$	709.86	548.06	774.50	-0.25591	0.11185	-0.02573	0.0	0.0
$\boldsymbol{C'_{sim}}$	680.49	542.70	771.48	-0.25031	0.10323	-0.02195	0.0	0.0
$\boldsymbol{\sigma_{sim}}$	0.3	0.3	0.2	0.0	0.0	0.0	0.0	0.0

Table B.1.4: Stereo camera calibrations. $E_{CC'}$ defines the transformation between both cameras with the translation parameters x, y, z and the orientation parameters $roll, pitch, yaw$. σ represents the calibration uncertainty.

	x $[m]$	**y** $[m]$	**z** $[m]$	**roll** $[°]$	**pitch** $[°]$	**yaw** $[°]$
$\boldsymbol{E_{CC'_{real}}}$	0.0023	0.022	0.0325	-1.56675	$2.8012e^{-3}$	1.58055
$\boldsymbol{\sigma_{real}}$	0.001	0.001	0.001	$3.11472e^{-4}$	$5.8293e^{-4}$	$3.02407e^{-4}$
$\boldsymbol{E_{CC'_{sim}}}$	-0.20068	$-2.65952e^{-4}$	$7.96739e^{-4}$	$5.19831e^{-3}$	$7.68439e^{-3}$	$5.53522e^{-3}$
$\boldsymbol{\sigma_{sim}}$	0.001	0.001	0.001	$3.11472e^{-4}$	$5.82932e^{-4}$	$3.02407e^{-4}$

Table B.1.5: Geometric calibrations of the IMU. E_{CI} defines the transformation between the left camera and the IMU with the translation parameters x, y, z and the orientation parameters $roll, pitch, yaw$. σ represents the calibration uncertainty.

	x $[m]$	**y** $[m]$	**z** $[m]$	**roll** $[°]$	**pitch** $[°]$	**yaw** $[°]$
$\boldsymbol{E_{CI_{real}}}$	-0.20087	-3.71598e^{-4}	5.41001e^{-4}	0.01084	6.794e^{-3}	4.477e^{-3}
$\boldsymbol{\sigma_{real}}$	8.02784e^{-5}	7.22689e^{-5}	8.28483e^{-5}	5.72702e^{-5}	6.40584e^{-5}	7.2014e^{-5}
$\boldsymbol{E_{CI_{sim}}}$	0.0023	-0.022	0.0325	-1.56675	2.80121e^{-3}	1.58055
$\boldsymbol{\sigma_{sim}}$	8.02784e^{-5}	7.22689e^{-5}	8.28483e^{-5}	5.72702e^{-5}	6.40584e^{-5}	7.20140e^{-5}

Table B.1.6: Parameters for the SGM algorithm from left to right: the actual SGM algorithm, the used cost table within the algorithm, the strategy to interpolate subpixels, the radiometric resolution and the maximum allowed disparity value of the resulting disparity map and the applied post processing filter method with its kernel size. Details for the algorithm and the cost matrix can be found in (Hirschmüller et al., 2012).

Alg.	**Cost Tab.**	**Interpolation**	**Post Filter**	**Rad. Res.** $[Bit]$	**Max. Disp.** $[px]$
eSGM	Census7x5s	PARABOLA	MEDIAN 3x3	8	256

Table B.1.7: Parameters for the point cloud creation algorithm from left to right: the size of a bin for point cloud voxelization, the maximum allowed spatial point cloud depth and the minimum necessary number of points to create a voxel.

Leaf Size $[m]$	**Max. Depth** $[m]$	**Min. Points per Voxel**
0.01	10.0	2

Table B.1.8: YOLO configuration. From left to right: The used YOLO version, the minimum allowed object detection probability, the applied processor to run YOLO and the input dimension of images for YOLO. Not mentioned parameters are not changed and taken as provided in (Redmon, 2013–2016).

Version	**Detection Threshold**	**Device**	**Spat. Res.**
v3	0.7	GPU	480x480

B.2 - CODAPT Configuration

Table B.2.1: CODAPT configuration parameters.

Parameter	Value	Description
Particle Translation Covariance	0.000225	Covariance to initialize the translation values for a particle.
Particle Orientation Covariance	10.0	Covariance to initialize the orientation values for a particle.
Particle Noise Covariance	0.00001	Covariance for the noise in the PF sampling process.
Particle Bin Size	0.01	Size for the cubic particle bins.
Delta	0.99	Delta bound value for the Kullback-Leibler distance calculation.
Epsilon	0.2	Epsilon bound value for the Kullback-Leibler distance calculation.
Particle Resampling Likelihood	0.0	Likelihood to completely resample all particles.
Normal Estimation Search Radius	0.03	Search radius to estimate the surface normal of a point.
Data Association Intersection Threshold	0.5	Search radius to estimate the surface normal of a point.
Non-Maximum Suppression Threshold Factor	6.0	Multiple of a point cloud resolution to take for the ISS keypoint non-maximum suppression.

B.3 - PC Hardware Configuration

Table B.3.1: Used PC hardware configuration for the evaluation part of this work. The evaluation runs on the SSD, datasets and results are stored on the HDD.

CPU	RAM	GPU	OS	Harddrive
Intel Core i7 6700 4 Cores 8 Threads @3400 MHz	32 GB DDR4 @1333 MHz	Nvidia GeForce GTX 1060 6 GB	Windows 7 Enterprise Build 7601	HDD 2 TB (Toshiba DT01ACA200) SSD 250 GB (Samsung PM871a)

C - Additional Evaluation Results

C.1 - Associated Object Detections

Table C.1.1: Associated Object Detections of the simulation datasets. The first column depicts the overall object detection associations with the tracking, the second column the amount of object detections (where object associations are expected) using the ground truth and the third column the association rate using both values. The abbreviation "Illum" in parentheses denotes the dataset with illumination change.

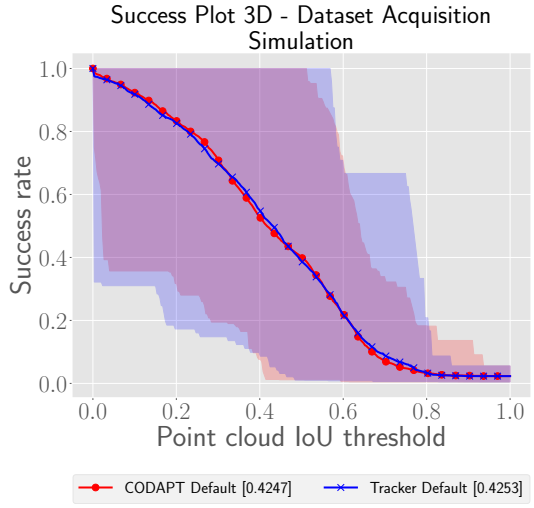
Scenario	Object	**Overall Associations**	**GT Object Detections**	**Rate**
Desert1	Bottle	18	474	**0.04**
	Laptop	98	408	**0.24**
	Motorbike	388	1584	**0.24**
Desert1 (Illu)	Bottle	66	474	**0.14**
	Laptop	110	408	**0.27**
	Motorbike	382	1584	**0.24**
Desert2	Bicycle	1308	2178	**0.60**
	Car	-	-	**-**
	Chair	86	456	**0.19**
Desert2 (Illu)	Bicycle	1283	2178	**0.59**
	Car	-	-	**-**
	Chair	98	456	**0.21**
Office1	Chair1	700	720	**0.97**
	Chair2	161	726	**0.22**
Office1 (Illu)	Chair1	694	726	**0.96**
	Chair2	124	726	**0.17**
Office2	Bottle	383	624	**0.61**
	Chair	556	744	**0.75**
	Laptop	48	616	**0.08**
Office2 (Illu)	Bottle	427	624	**0.68**
	Chair	630	744	**0.85**
	Laptop	106	604	**0.18**
Hexenh. 1	Bottle	178	1548	**0.11**
	Chair	223	828	**0.27**
	Laptop	313	1140	**0.27**
Hexenh. 1 (Illu)	Bottle	366	1548	**0.24**
	Chair	237	852	**0.28**
	Laptop	285	1140	**0.25**
Hexenh. 2	Bicycle	602	1374	**0.44**
	Car	-	-	**-**
	Motorbike	477	942	**0.51**

Hexenh. 2 (Illu)	Bicycle	589	1374	**0.43**
	Car	-	-	**-**
	Motorbike	441	942	**0.47**

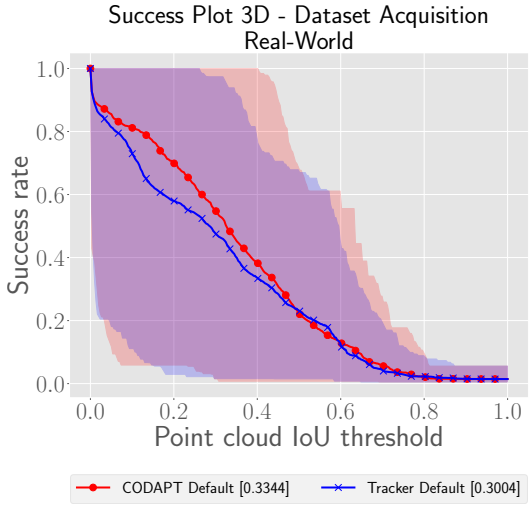
Table C.1.2: Associated object detections of the real dataset. Analogous to Table C.1.1. Rates above 1.0 mean that the tracking results were erroneously matched to multiple object detections in some frames.

Scenario	Object	**Overall Associations**	**GT Object Detections**	**Rate**
Office	Chair	674	1404	**0.48**
	Laptop	0	1296	**0.00**
	Monitor	0	1350	**0.00**
Office (Illu)	Chair	580	768	**0.76**
	Laptop	237	480	**0.49**
	Monitor	0	1416	**0.00**
Parking Lot 1	Laptop	162	150	**1.08**
Parking Lot 2	Laptop	158	150	**1.05**
Backdoor 1	Bicycle	256	408	**0.63**
	Laptop	134	708	**0.19**
	Motorbike	667	1122	**0.59**
Backdoor 2	Laptop	113	324	**0.35**
	Motorbike	1085	1380	**0.79**

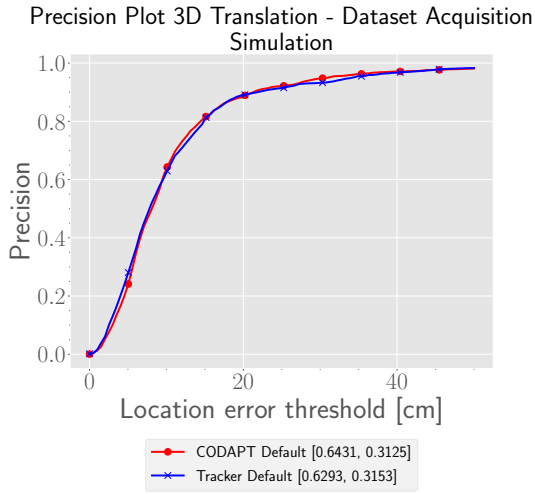
C.2 - 3D Object Pose Tracking Accuracy



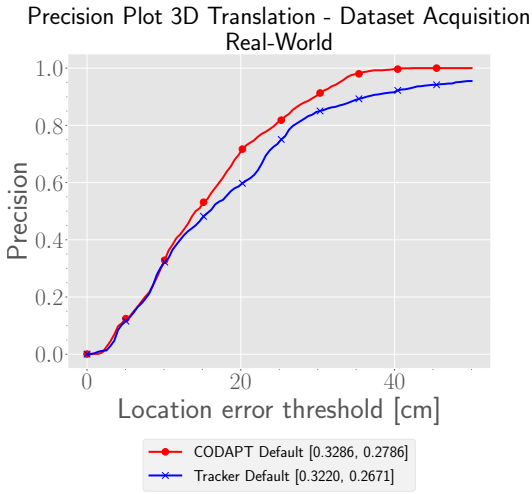
(a)



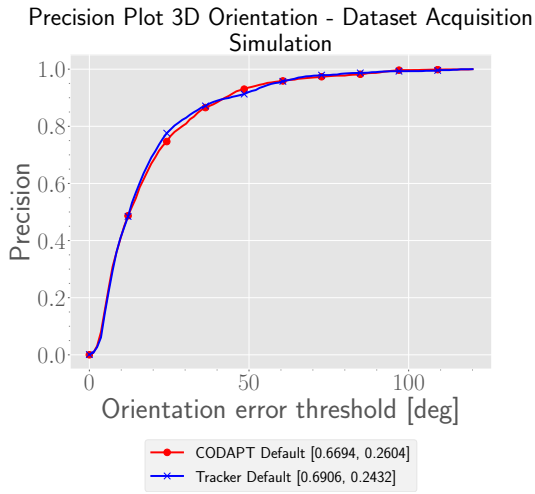
(b)



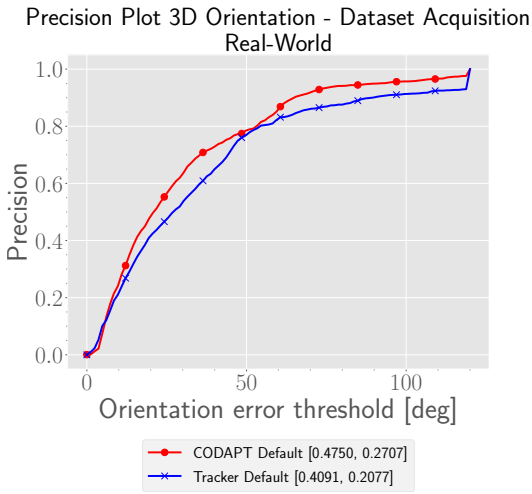
(c)



(d)



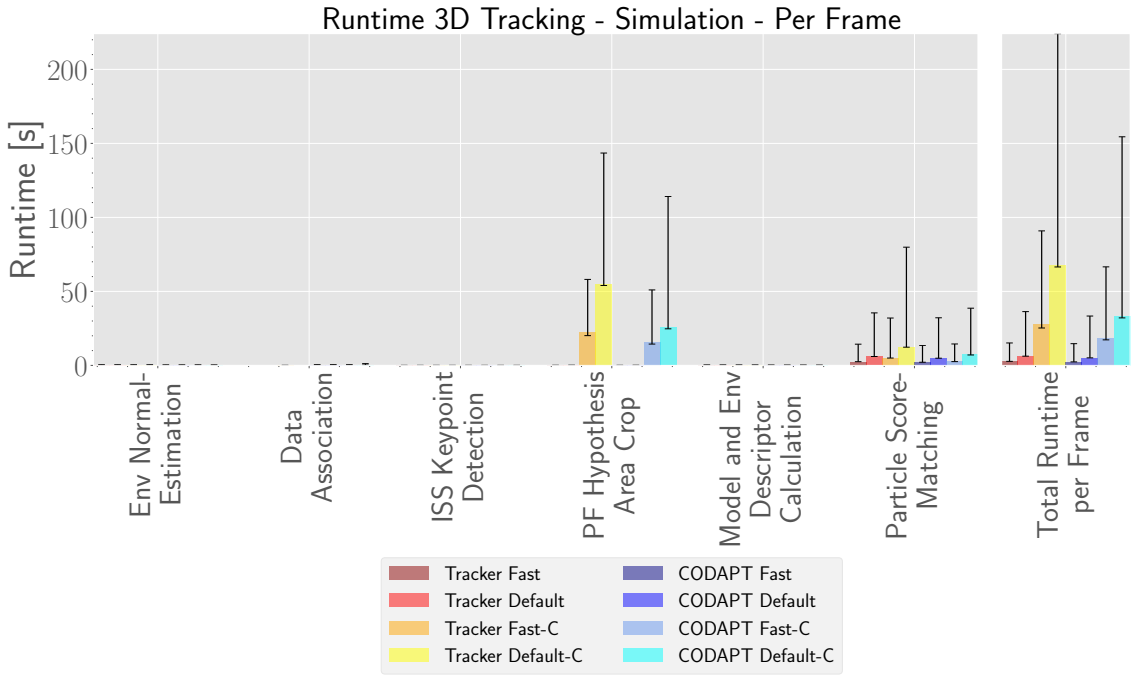
(e)



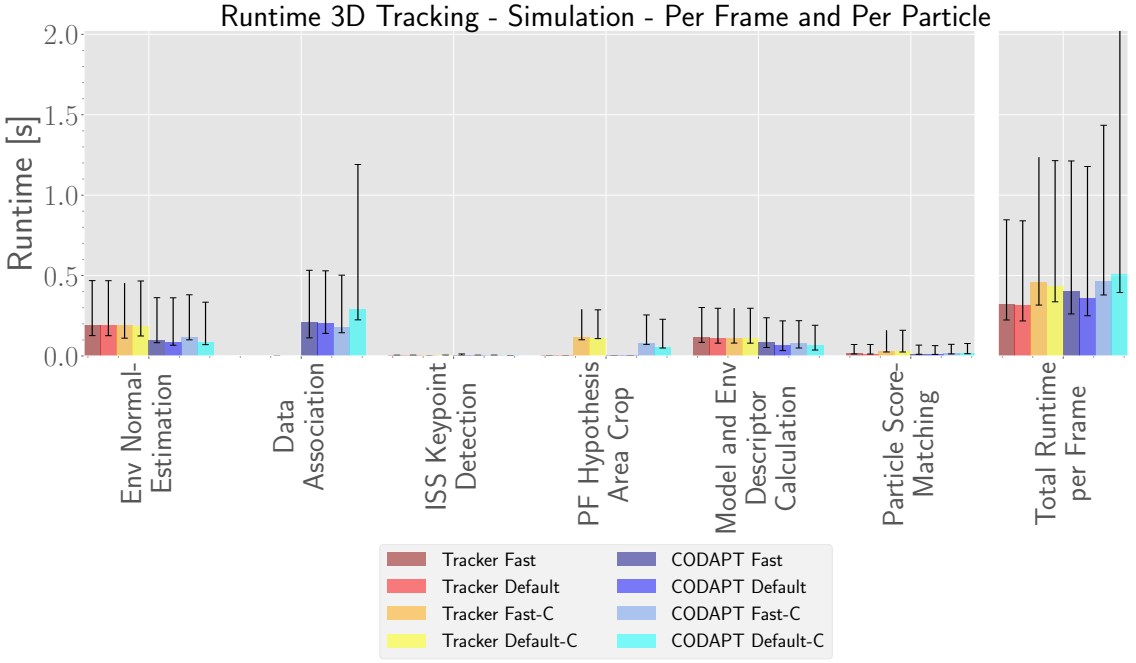
(f)

Figure C.2.1: Full-dataset evaluation results for the dataset acquisition methods.

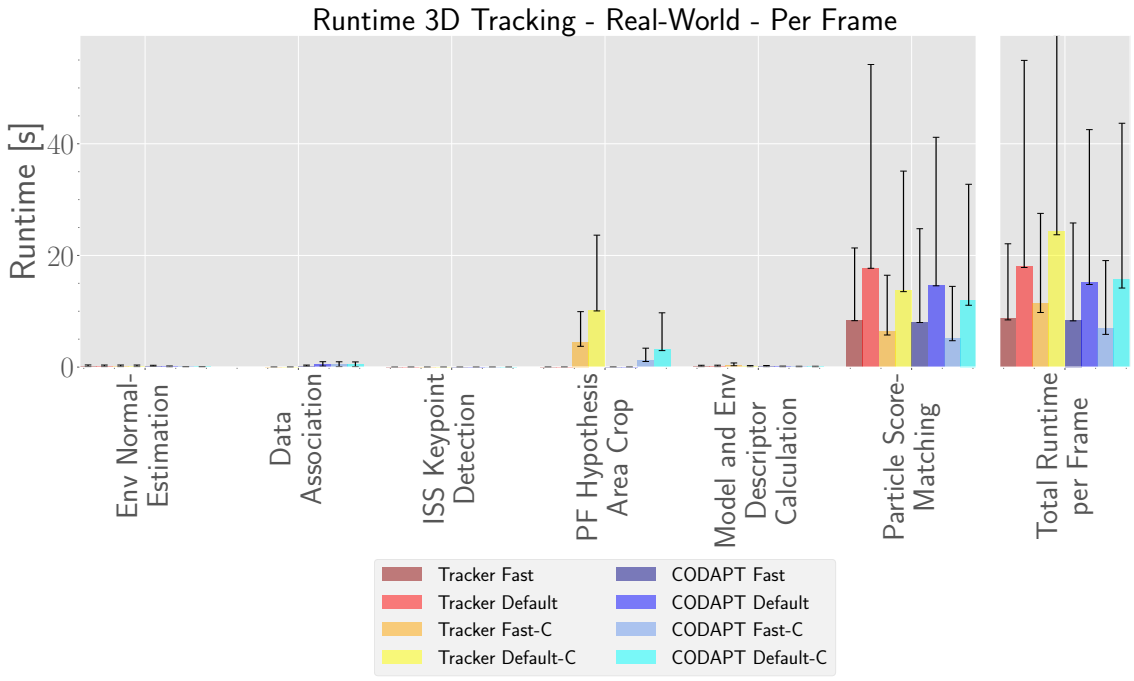
C.3 - 3D Object Pose Tracking Runtime



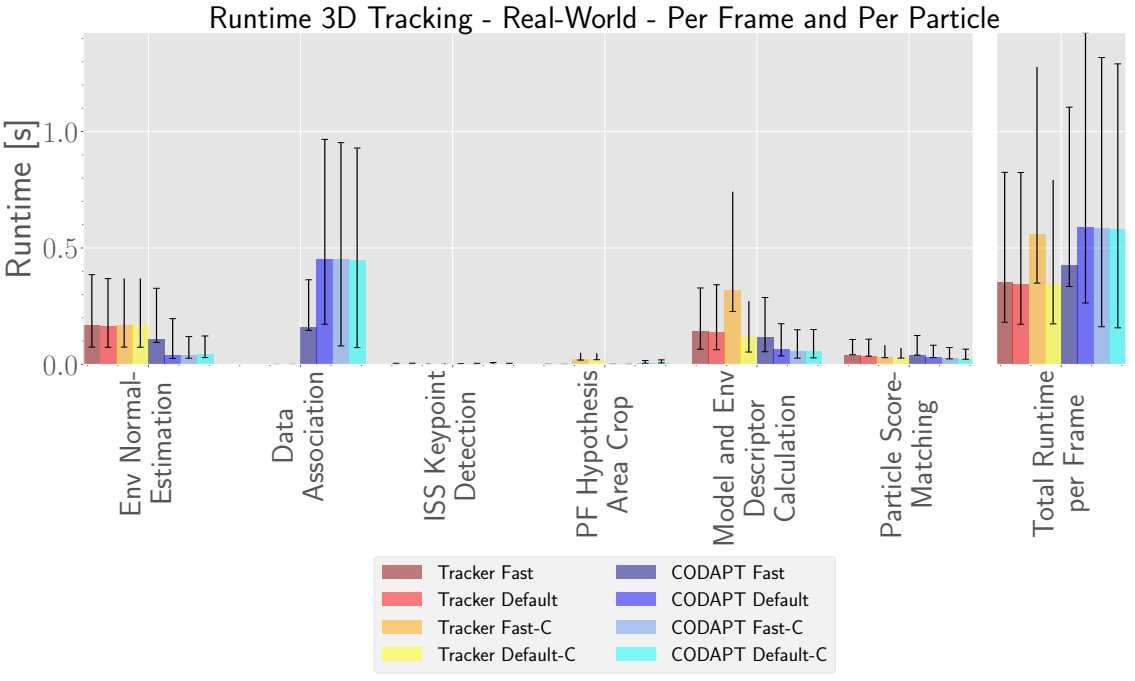
(a)



(b)



(c)



(d)

Figure C.3.1: Additional runtime evaluation results.

C.4 - 2D Object Detection Accuracy

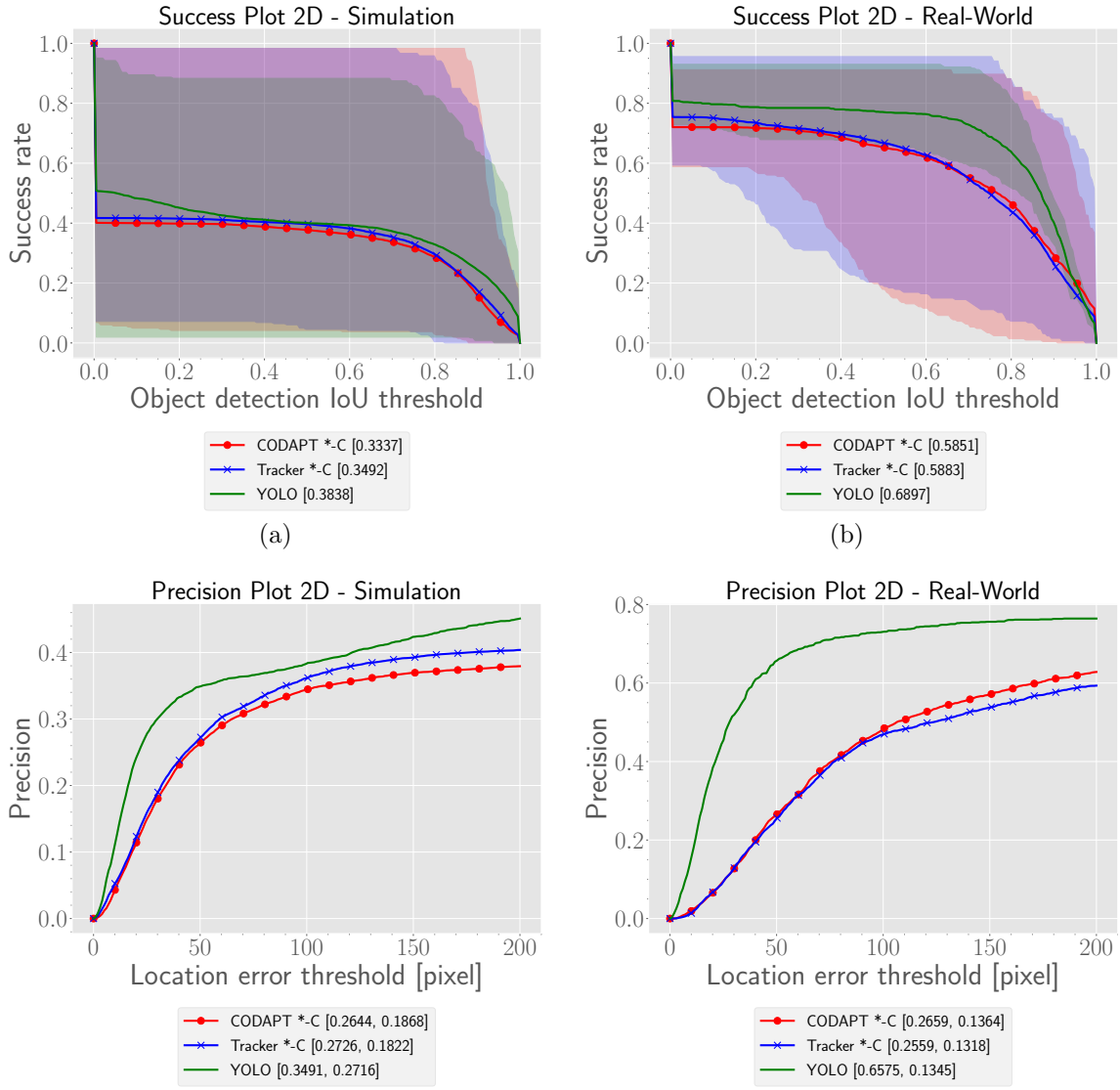


Figure C.4.1: Additional 2D object detection accuracy results.

D - Used External Software Frameworks

The proposed 3D object tracker exploits the external software frameworks listed in Table D.1.

Table D.1: External software frameworks used by the proposed 3D object tracker.

Name	**Note**
OSLib	DLR intern C++ software library with basic structures and algorithms for image processing. Last accessed: 29.08.2019.
OSVisionLib	DLR intern C++ software library with computer vision algorithms and interfaces to apply external libraries. Last accessed: 29.08.2019.
OSPythonLib	DLR intern Python software library with machine learning algorithms and evaluation scripts. Last accessed: 29.08.2019.
PCL	Point Cloud Library 1.8 (R. B. Rusu and Cousins, 2011). Large scale, open-source project for 2D/3D image and point cloud processing. Last accessed: 29.08.2019.
OpenSceneGraph	OpenSceneGraph 3.4.0 (Burns, 1998–2019) is an open-source 3D graphics toolkit for visual simulation, computer games, virtual reality, scientific visualization and modeling. It bases on C++ and OpenGL. Last accessed: 29.08.2019.