Semantic SLAM for Dynamic Environments

Nadia Garson Wangberg
Jan Tommy Gravdahl, Trygve Utstuemo

Department of Engineering Cybernetics — NTNU 2022

Abstract

There was a strong expectation ten years ago for the adaptation of mobile robots in commercial and industrial applications. Yet, many would say that the number of deployed mobile robots in these settings is disappointing. We believe a significant cause is the difficulty in designing robust SLAM systems capable of high performance in real-world environments.

Today, many visual SLAM algorithms perform well in ideal cases but fail to generalize to diverse real-world scenarios. There have been significant innovations within the field of SLAM, but long-term operations remain difficult. Robust performance in dynamic environments is particularly challenging, and these settings are common in both commercial and industrial applications. This thesis explores the challenges visual inertial SLAM faces in dynamic scenes and methods to improve their performance in these settings.

The main contribution of this thesis is the proposal of Semantic-Kimera-VIO, a modified version of Kimera-VIO. Semantic-Kimera-VIO is an open-source stereo inertial SLAM system designed for dynamic scenes. Our system uses semantic segmentation images to classify and discard image feature points from dynamic objects. We demonstrate that our version outperforms the original Kimera-VIO in terms of Relative Position Error (RPE) on the highly dynamic VIODE datasets.

Sammendrag

For ti år siden var det høye forventninger til bruk av mobile roboter i kommersiell og industriell virksomhet. Imidertidig, vil mange si at antallet mobile roboter i bruk i virksomhet er skuffende. Vi tror en av hovedårsakene er vanskeligheten i å designe robuste SLAM-sytemer med høy ytelse i virkelige omgivelser.

I dag fungerer mange visuelle SLAM-algoritmer godt i ideelle omgivelser, men sliter med å generalisere til mangfoldige virkelige omgivelsene. Det har vært sterk innovasjon innenfor SLAM feltet, men langsiktig kjøring av SLAM er fortsatt en utfordring. Robust ytelse i dynamiske omgivelser er spesielt utfordrende, og slike omgivelser er svært vanlig i både komersiell og industriell virksomhet. Denne oppgaven undersøker hvorfor dynamiske omgivelser er utfordrende for visuell inertial SLAM og hvilke metoder som kan brukes for å forbedre ytelsen.

Hovedbidraget til denne oppgaven er Semantic-Kimera-VIO, en modifisert versjon av Kimera-VIO. Semantic-Kimera-VIO er en stereo inertiell SLAM algoritme med åpen kildekode, designet for dynamiske omgivelser. Vår versjon bruker semantiske segmenteringsbilder for å klassifisere og forkaste visuelle feature punkter fra dynamiske objekter. Vi demonstrerer at vår versjon har en lavere Relativ Posisjon Error (RPE) enn Kimera-VIO på det svært dynamiske VIODE datasettet.

Preface

The research presented in this thesis was done at the Department of Engineering Cybernetics at the *Norwegian University of Science and Technology* (NTNU). Jan Tommy Gravdahl from NTNU and Trygve Utstumo from Cognite supervised the thesis. This project was done in collaboration with Cognite, who provided the author with a laptop and motivated the use of a modern SLAM system suitable for industrial environments.

This thesis is mainly original work, with parts of the background theory and introduction restated and expanded from the author's specialization project. The specialization project is unpublished, titled "Robustness of Off-the-shelf Visual Tracking Cameras for Wheeled Robotics", and conducted in the autumn of 2022. Unless otherwise stated, all figures and illustrations have been created by the author.

The code for the implemented system is hosted at github.com/nadiawangberg/masters-thesis, and the code for the evaluation of the system is hosted at github.com/nadiawangberg/eval.

Acknowledgements

I want to thank my professor Jan Tommy Gravdahl for giving excellent tips on research and thesis writing. Thanks to Trygve Utstumo and Cognite for giving me the freedom to choose a topic I was interested in and lending me the necessary equipment to complete this project. Thanks to Trym Haavardsholm for a fascinating course on SLAM, tips on choosing a thesis topic, and feedback on the final report. Thanks to Tyler for giving especially in-depth feedback on my writing. I would also like to thank my mom and grandma for always supporting me, and give a special thanks to Ludvig for everything.

Trondheim, June 2022
Nadia Garson Wangberg

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Master thesis assignment

Name of the candidate: Nadia Wangberg
Subject: Engineering Cybernetics
Title:

Semantic SLAM for Dynamic Environments

Theoretical assignment:

1. **Describe previous work on VSLAM,** with an additional focus on performance in dynamic scenes. Summarize the most important recent works on improving the robustness of VSLAM in dynamic scenes.
2. **Describe the essential components of VSLAM**
3. **Describe the challenge of tracking in dynamic scenes -** Define dynamic scenes and why dynamic scenes are challenging for many VSLAM algorithms. Describe how previous VSLAM algorithms handle dynamic objects. Describe why performing well in dynamic scenes is important for industrial and commercial applications.
4. **Describe the evaluation of SLAM** - What is the difference between accuracy and robustness? How should robustness be evaluated?

Practical assignment:

1. **Choose an open-source VSLAM algorithm** suitable for commercial and industrial use, with performance in dynamic scenes in mind.
2. **Modify the chosen open-Source VSLAM algorithm** to improve performance in dynamic environments.
3. **Evaluate performance on highly dynamic datasets** - Test on several datasets, including highly dynamic datasets and static datasets. Ensure that the datasets display a variety of environments. Evaluate both based on visual inspection and using metrics for accuracy and robustness.

External advisor: Trygve Utstumo, Cognite AS
Project assigned: January 10th, 2021
To be handed in by: June 6th, 2021

Trondheim, January 28, 2021

Jan Tommy Gravdahl
Professor, supervisor

Contents

1 Introduction

Simultaneous localization and mapping (SLAM) is the process of estimating the position and orientation of an agent while simultaneously mapping an unknown environment. SLAM is commonly used in the fields of virtual and augmented reality, self-driving cars, 3D reconstruction, and autonomous navigation in *Global Navigation Satellite System* (GNSS) denied environments.

In this chapter we present the motivations, contributions and structure of the thesis in section 1.1, section 1.2 and section 1.3 respectively. Then in section 1.4, we present previous work on *Visual SLAM* (VSLAM), with an additional focus on contributions to improve robustness in dynamic environments.

1.1 Motivation

Cadena et al. (2016) argues SLAM is entering the "Robust-Perception age" characterized by an increased focus on robust performance in diverse environments. Cadena et al. (2016) define *robustness* as the ability to track in a broad range of environments for extended periods. Environments with moving objects, also known as dynamic scenes, are mentioned as particularly challenging (Cadena et al., 2016). These settings pose a significant challenge for industrial and commercial applications, which commonly have moving people, vehicles, or machines. Achieving robust performance in these settings is key to bringing autonomous robots out of the laboratory and into the industry.

Traditionally when developing SLAM algorithms, the world is assumed static. This assumption is known as the *static world assumption*, which greatly simplifies the SLAM problem. These SLAM systems perform well in ideal settings but tend to fail in dynamic environments, where the static world assumption no longer holds (Cadena et al., 2016).

In short, the motivation of this thesis is to improve the performance of VSLAM in dynamic scenes. As current methods are not designed with dynamic environments in mind, tracking failures are common. Currently, mobile robots cannot reliably navigate in real-world scenarios with a high density of dynamic objects. We believe that improving tracking in dynamic scenes is necessary to bring mobile robots into the industry and commercial applications. Our goal in this project is to explore new methods to filter out feature points produced by dynamic objects, ultimately improving mobile robots' navigation in non-ideal real-world scenes.

1.2 Contributions

In essence, the main contributions of this thesis are to:

- Compare open-source VSLAM algorithms, and choose a system suitable for dynamic industrial environments.

- Modify the frontend of the modern open-source *Stereo Inertial SLAM* algorithm Kimera, to filter out dynamic outliers using semantic segmentation.
- Improve Kimeras visualization, allowing for a more straightforward visual inspection of the results.
- Evaluate robustness using metrics beyond the traditional *Absolute Trajectory Error* (ATE) and *Relative Position Error* (RPE) metrics.
- Experimentally show that the addition of semantic outlier removal in Kimera increased accuracy in terms of RPE on the *Visual-Inertial Odometry in Dynamic Environments* (VIODE) dataset.
- Experimentally show that Kimeras geometric verification is sufficient in moderately dynamic environments, such as on the uHumans2 dataset.

Experimentally show that the addition of semantic outlier removal to Kimera increased accuracy,

1.3 Thesis Structure

In this thesis, we improve the accuracy of the modern stereo inertial SLAM algorithm Kimera in dynamic environments. In chapter 1 we present the motivation and contributions of this thesis. Additionally, we present previous work on VSLAM, with an additional focus on systems designed for dynamic scenes. Chapter 2 covers a brief overview of the SLAM problem. Chapter 3 further describes the VSLAM frontend and its specific difficulties in dynamic environments. We present methods for evaluating SLAM and VO in chapter 4. Chapter 5 describes semantic segmentation on image data and how we modified Kimera to utilize semantic data to increase performance in dynamic scenes. We show and discuss our results in chapter 6. Finally, in chapter 7 we present a conclusion and further works.

1.4 Previous Work

This section briefly presents previous work on VSLAM, largely in chronological order. We also highlight how the field has developed toward integrating several sensors to increase performance. Different strategies for designing VSLAM systems are presented in section 1.4.1, monocular SLAM in section 1.4.2, stereo SLAM in section 1.4.3, and visual inertial SLAM in section 1.4.4. Finally, in section 1.4.5, VSLAM solutions specifically designed to tackle dynamic scenes are presented.

1.4.1 Visual SLAM

Visual SLAM (VSLAM) simultaneously estimates the camera motion, also known as *ego-motion*, and a map of the environment using camera data. VSLAM algorithms often depend on bright constant light conditions and commonly require more computation power than LiDAR-based SLAM. An advantage of cameras is their cheap price point, low weight, high frequency, and long-range (Debeunne and Vivet, 2020). This section discusses a few different methods of designing VSLAM systems.

VSLAM or Visual Odometry

VSLAM differs from *Visual Odometry* (VO) as VSLAM performs loop closure. Loop closure corrects for accumulated drift by recognizing a previously visited location and updating the map and location accordingly. A figure of the SLAM map before and after loop closure is shown in Figure 1.1. Place recognition and loop closure is further explained in section 2.2.

Direct or Indirect

Visual measurement processing is often categorized into two different paradigms - direct and indirect methods. Indirect methods, such as those used in ORB-SLAM (Mur-Artal et al., 2015) are feature-based and minimize geometric error. Direct methods work directly on image intensity values and minimize photometric error. Direct methods, such as LSD-SLAM (Engel et al., 2014) and DSO (Engel et al., 2016), often provide denser maps and are less computationally heavy but lack robustness to photometric and geometric distortions. On the other hand, they are known to be robust to low texture, motion blur, de-focus, and high-frequency textures such as carpet or asphalt (Cadena et al., 2016), (Lovegrove et al., 2011).

Filtering or Optimization

Initially, SLAM was solved by filtering-based approaches, such as the *Extended Kalman Filter* (EKF) or the *Particle Filter* (PF) (Bailey et al., 2006), (Sim et al., 2005). EKF-based SLAM implementations suffered from linearization errors and limitations of the map size (Bailey et al., 2006). For filtering-based SLAM, the current state X_t can be estimated from the posterior distribution given by the filtering equation:

$$P(X_t|Z_1...Z_t), \tag{1.1}$$

where $Z_1...Z_t$ are all the previous measurements up the present measurement (Russell and Norvig, 2010).

Today most SLAM methods are smoothing-based, meaning they estimate the full pose trajectory and map based on the full set of measurements. Modern methods are also optimization and keyframe-based, which show impressive results (Campos et al., 2020), (Qin et al., 2017), (Engel et al., 2014), (Wang et al., 2017). Unlike filtering, smoothing-based methods re-estimates past states given all the measurements up to the present. In smoothing-based SLAM, a past state X_k is estimated from the posterior distribution given by:

$$P(X_k|Z_1...Z_t), \tag{1.2}$$

where $Z_1...Z_t$ are all the measurements up to the present measurement, and $k \subseteq [0, t)$ (Russell and Norvig, 2010).

1.4.2 Visual Monocular SLAM

Monocular SLAM solves the SLAM problem using a single camera as its sensor. A disadvantage of monocular SLAM is its cumbersome *Structure from Motion* (SfM) initialization process (Mur-Artal and Tardos, 2017). Another main disadvantage is that depth is unobservable, making the metric scale of both the map and the estimated trajectory unknown.

The unobservable metric scale causes scale drift, which can be accounted for by global bundle adjustment, which optimizes for scale.

One of the most influential keyframe-based indirect monocular SLAM algorithms is *Parallel Tracking and Mapping* (PTAM) (Klein and Murray, 2007). PTAM is often mentioned as crucial preliminary work for several modern SLAM methods such as ORB-SLAM (Mur-Artal et al., 2015) and SVO (Forster et al., 2014). It was unique for the time with its key-frame selection, feature matching, point triangulation, and relocalization.

ORB-SLAM

ORB-SLAM (Mur-Artal et al., 2015) is a notable indirect optimization-based monocular SLAM system. Like PTAM, ORB-SLAM is keyframe-based but improves upon many of its components. It includes better place recognition, loop closure, robustness to occlusions, and allows mapping in larger areas (Mur-Artal et al., 2015). Although improved, its monocular sensor setup causes initialization difficulties and an unobservable metric scale.

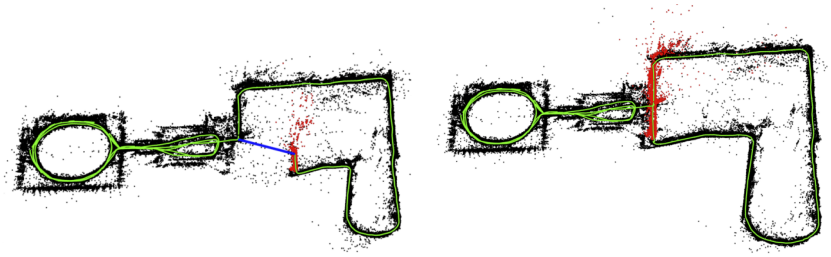

Figure 1.1: The monocular ORB-SLAM map before loop closure (left) and after loop closure (right). Courtesy of *IEEE Transactions on Robotics* (see Mur-Artal et al., 2015, Fig. 5.).

The unobservable metric scale can cause notable scale drift in the ORB-SLAM map, shown in the left-most image in Figure 1.1. ORB-SLAM performs 7DoF scale drift aware loop closure, which corrects for the scale drift in many cases (Mur-Artal et al., 2015), shown in the right-most image in Figure 1.1. Nevertheless, ORB-SLAM suffers from initialization difficulties, and in the cases where scale-drift cannot be corrected, the pose estimate is often unusable (Mur-Artal and Tardos, 2017).

Improving Robustness of Monocular SLAM

Several solutions have been proposed to increase robustness and decrease failures of monocular SLAM. Both ORB-SLAM and LSD-SLAM are scale-drift aware, detecting and correcting scale-drift errors during loop closure. Deep-learning methods have been proposed to estimate depth from a single frame in an attempt to make scale observable for monocular data (Zhan et al., 2021), (Godard et al., 2018), (Yang et al., 2018). Scale can also be estimated by assuming a constant known height between the ground plane and the camera. This requires ground plane detection and a constant height between the camera and plane (Fanani et al., 2017). Simultaneous Planning Localization and Mapping (SPLAM) methods have been proposed for avoiding motions that could cause monocular SLAM failures. Prasad et al. (2016) proposes a SPLAM algorithm that uses reinforcement learning to generate trajectories to optimize accuracy and robustness.

1.4.3 Stereo SLAM

Researchers have increasingly utilized multiple sensors to handle the issues monocular SLAM faces. Issues such as difficult initialization and scale estimation can both be solved by using a stereo camera setup. Stereo allows the system to extract depth, and hence also the metric scale from a single frame pair (Mur-Artal and Tardos, 2017), (Wang et al., 2017).

Stereo cameras can only accurately estimate depth with a sufficiently large baseline. In ORB-SLAM2, depth is extracted directly from stereo or RGB-D if the depth of the observed points is lower than 40 times the baseline (Mur-Artal and Tardos, 2017). For distances longer than 40 times the baseline, ORB-SLAM2 would not extract depth directly and would essentially be equivalent to a monocular system. For a system with a 6cm baseline, ORB-SLAM2 would not extract depth directly for distances larger than 2.4m. In other words, failures common for monocular systems could occur in open environments if the stereo baseline is too short. This motivates the introduction of additional sensors to improve performance.

1.4.4 Visual Inertial SLAM

Another method that increases robustness and makes metric scale observable is the fusion of data from an *Inertial Measurement Unit* (IMU) (Qin et al., 2017), (Mourikis and Roumeliotis, 2007). Such systems are known as *Visual Inertial Odometry* (VIO) and *Visual Inertial SLAM* (VISLAM). The IMU is a proprioceptive sensor that measures acceleration and angular rates at a high frequency. Unlike cameras which are exteroceptive sensors, IMUs are not affected by most external environmental factors. This includes being unaffected by varying light conditions, motion blur during fast motion, featureless environments, and dynamic scenes. The addition of inertial data could therefore increase robustness in dynamic scenes (Koji et al., 2021), (Rosinol et al., 2020).

Figure 1.2: Intel Realsense T265i stereo-inertial SLAM camera (Realsense, 2021)

Some of the most prominent SLAM solutions today utilize both stereo and IMU data (Campos et al., 2020), (Qin and Shen, 2018), (Grunnet-Jepsen et al.). Figure 1.2 shows the Intel Realsense T265i tracking camera, a stereo camera with integrated IMU. This camera is a hardware-software combination that runs VISLAM onboard (Grunnet-Jepsen et al.).

1.4.5 SLAM Designed for Dynamic Environments

While the systems discussed above focus on the VSLAM problem as a whole, this section focus on the specific contributions to an improved ego-motion estimation in dynamic

scenes. Saputra et al. (2018) presented a survey on VSLAM and SfM in dynamic environments, which suggested two approaches for solving the problem; *Robust Visual SLAM* and *Joint Motion Segmentation and Reconstruction*. Robust Visual SLAM outputs a map of static points; the dynamic feature points are detected and discarded. Joint Motion Segmentation and Reconstructions additionally track dynamic objects. This method outputs a dynamic map in addition to a static map. The main focus of this section is Robust Visual SLAM.

ORB-SLAM in Dynamic Scenes

ORB-SLAM (Mur-Artal et al., 2015) claims to be somewhat robust to dynamic scenes, outperforming PTAM and LSD-SLAM. Figure 1.3 shows relocalization in a dynamic scene for ORB-SLAM, where the blue lines visualizes the matched features between the images.

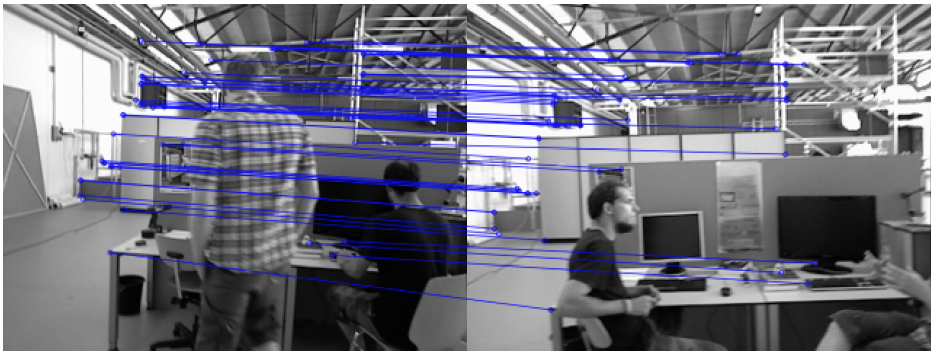

Figure 1.3: Relocalization of ORB-SLAM in a dynamic environment. Courtesy of *IEEE Transactions on Robotics* (see Mur-Artal et al., 2015, Fig. 8.).

One of ORB-SLAMs main contributions is its survival of the fittest strategy applied to keyframe and map point selection. This strategy frequently adds map points and keyframes, boosting tracking robustness in challenging settings such as fast movements. ORB-SLAM later removes redundant keyframes and map points, allowing the map to be bounded in size when viewing the same area. This strategy may be one reason why ORB-SLAM is more robust in dynamic scenes than other monocular methods from its time, such as PTAM and LSD-SLAM.

DS-SLAM

DS-SLAM Yu et al. (2018) builds on ORB-SLAM2 and improves robustness in dynamic scenes. DS-SLAM combines semantic information with a geometric moving consistency check to remove dynamic outliers. This method is not reviewed in Saputra et al. (2018) but could be classified as Robust Visual SLAM as dynamic points are dismissed and considered outliers.

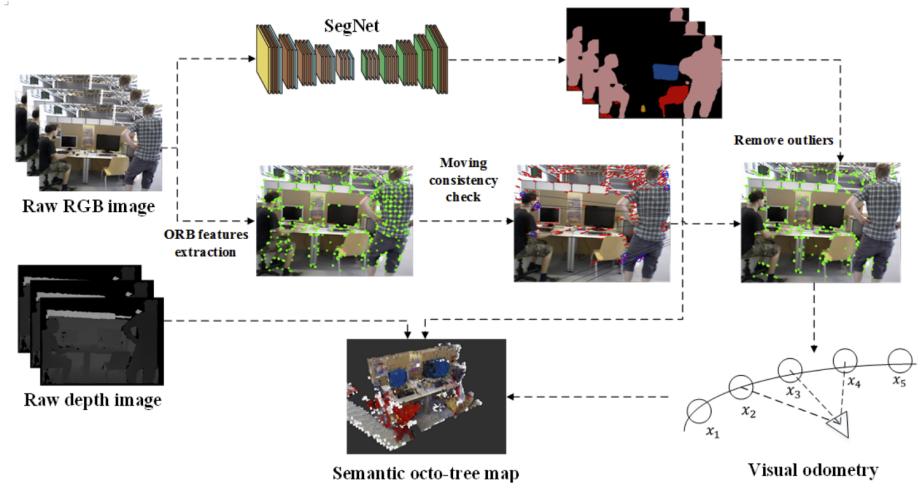

Figure 1.4: Pipeline of DS-SLAM. Courtesey of *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (see Yu et al., 2018, Figure. 1.)

DS-SLAM's geometric moving consistency check utilizes epipolar geometry to detect moving objects. SegNet is used for semantic segmentation, and feature points located on people are classified as likely outliers. The semantic and geometric checks are combined, filtering out points detected as dynamic by both. DS-SLAM runs in real-time, and the results show an increased performance over ORB-SLAM2 on the TUM RGB-D dataset, indicating improved performance in dynamic scenes.

SOF-SLAM

Semantic Optical Flow SLAM (SOF-SLAM) Cui and Ma (2019) shares many similarities to DS-SLAM. Like in DS-SLAM, ORB-SLAM2 is modified to disregard dynamic feature points, SegNet is used for segmentation, and the TUM-RGB-D dataset is used for testing. SOF-SLAM is also not reviewed in Saputra et al. (2018) but could be classified as Robust Visual SLAM based on the same reasoning as with DS-SLAM. Unlike DS-SLAM, SOF-SLAM couples semantic and geometrical information in a tightly coupled manner. This is done by using semantics to aid the calculation of the epipolar geometry instead of performing a loose voting process between the semantics and the epipolar geometry. SOF-SLAM performs well and outperforms DS-SLAM. Several frames could have been used to detect dynamic features to improve SOF-SLAM, and a probabilistic framework could have been implemented to estimate the likelihood of a feature point being dynamic.

Dynamic SLAM

As opposed to the systems mentioned above, Dynamic SLAM Henein et al. (2020) includes dynamic objects in the map instead of filtering them out. This is done by introducing dynamic object factors in the factor graph. As dynamic features are no longer considered outliers but instead included in the estimation, this algorithm could be classified as Joint Motion Segmentation and Reconstruction by Saputra et al. (2018). A downside of Dynamic SLAM is that it is not suitable for long-term applications, as the graph of dynamic objects grows with time.

VINS-Mask

The main contribution of the VIODE (Koji et al., 2021) is the simulated visual-inertial dataset for benchmarking the robustness of VISLAM and VIO in dynamic scenes, further described in section 5.4.2. In addition to the dataset, VIODE also proposes VINS-Mask, a method to improve the robustness of VISLAM in dynamic scenes. VINS-Mask is a modified VINS-Mono that incorporates semantic segmentation to filter out dynamic outliers in the front end. The method is similar to DS-SLAM and SOF-SLAM, although VINS-Mask does not combine semantics with geometric verification. VINS-Mask also differs from DS-SLAM and SOF-SLAM as it is a VISLAM algorithm that incorporates IMU.

Kimera and Robustness in Dynamic Scenes

Kimera is a modern open-source VISLAM library with future research in mind (Rosinol et al., 2020). Due to combing both stereo and inertial data, Kimera accurately estimates depth and is somewhat robust to visually challenging data. Kimera consists of four modules, Kimera-VIO, Kimera-Mesher, Kimera-Semantics, and Kimera-RPGO. (Rosinol et al., 2021). The code is modular and runs in real-time on a CPU, making it easy to contribute to and run on an average laptop. Kimera differentiates itself from other modern SLAM algorithms as it performs both dense mesh reconstruction and 3D semantic labeling. In addition to its unique features, Kimera compares well with other modern VINS algorithms, such as VINS-mono on the EuRoC MAV Dataset.

Kimera uses modern techniques for outlier rejection of erroneous loop closures in its pose graph optimizer. Erroneous loop closures are detected using mono and stereo geometric verification. Additionally, a modified *Incremental Consistency Measurement Set Maximization* (PCM) designed for single robots are used to improve outlier loop closure rejection.

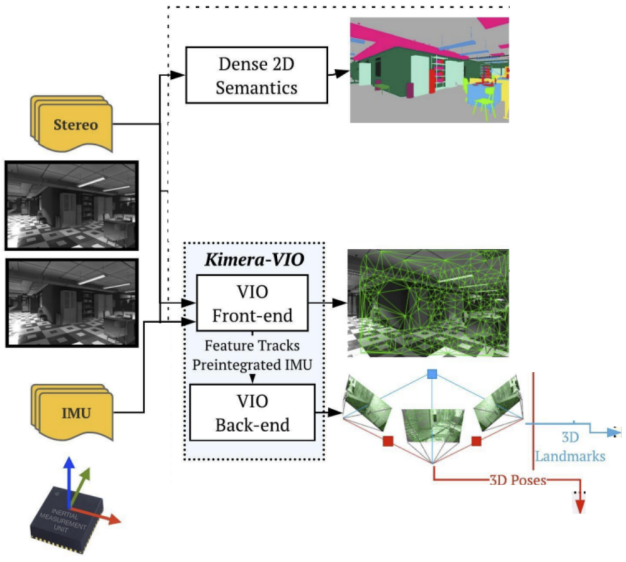

Figure 1.5: Simplified module diagram of Kimera-VIO. This figure is a modified version of Fig. 2. in Rosinol et al. (2020). This module diagram does not include the Kimera-Mesher, Kimera-Semantics, Kimera-RPGO, nor modules introduced in Rosinol et al. (2021).

In Rosinol et al. (2021), new Kimera modules were introduced, which were built on top of the original open-source Kimera. Kimera-DSG constructs a 3D *Dynamic Scene Graph* (DSG). The DSG contains a hierarchical representation of buildings, rooms, and objects, useful for high-level understanding and decision-making. However, this module does not run in real-time and does therefore not contribute to the localization task in the SLAM problem.

Another important contribution of Rosinol et al. (2021) is the introduction of Kimera-PGMO. Kimera-PGMO builds on Kimera-RPGO, which was introduced with the original Kimera in Rosinol et al. (2020). Unlike Kimera-RPGO, which only optimizes the pose graph, Kimera-PGMO simultaneously optimizes the mesh. The new modules introduced in Rosinol et al. (2021) are not open-source. Therefore, they were difficult to get access to, and we have not further investigated whether they contribute to improved robustness in dynamic scenes.

2 The Components of VO and VSLAM

VSLAM uses camera data to map an unknown environment while simultaneously estimating the cameras motion within the environment (Irani et al., 1994). SLAM is often necessary for applications in GNSS denied environments, such as virtual and augmented reality and navigation of autonomous indoor robotics. Even in outdoor applications, SLAM is often useful as GNSS runs at a relatively low frequency and lacks accuracy near large buildings. The output of the VISLAM algorithm Kimera is shown in Figure 2.1.

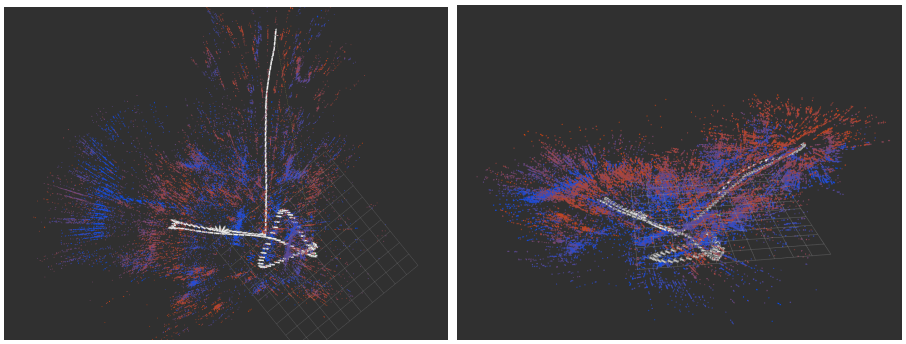

Figure 2.1: The sparse map and pose trajectory of Kimera-VIO on the static VIODE parking-lot dataset.

This chapter presents an overview of the components of optimization-based VSLAM and VO. As presented in section 1.4, SLAM can be formulated both as a filtering- or optimization-based problem. However, this chapter will only focus on optimization-based SLAM, as these methods are most relevant for modern SLAM systems (Cadena et al., 2016).

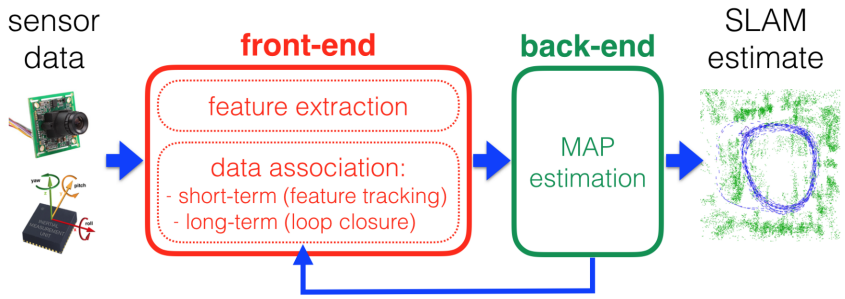

Figure 2.2: Pipeline of a typical optimization based SLAM system. Courtesy of *IEEE Transactions on Robotics* (see Cadena et al., 2016, Fig. 2)

Figure 2.2 shows the typical architecture of an optimization-based SLAM problem. SLAM is commonly divided into two main components: the frontend and backend. The frontend extracts data from the sensors and performs data association between the measurements and the map. Typical sensor data for SLAM is presented in section 2.1, while

the frontend is presented in section 2.2. The backend uses the data associations from the frontend to estimate the ego-motion and map (Cadena et al., 2016). The states estimated by SLAM is described in section 2.4, and the SLAM backend in section 2.3.

2.1 Sensor Data

The input of the SLAM frontend is raw sensor data, as seen in Figure 2.2. The sensor data can be from primary sensors and additionally from complementary sensors. Common primary sensors are cameras and LiDARs, while IMUs and wheel odometers can be used as complementary sensors. As discussed in section 1.4 modern algorithms utilize multiple sensors to increase robustness. This thesis focuses on stereo inertial SLAM, which uses data from stereo camera and IMU.

The input of the SLAM frontend is raw sensor data, as seen in Figure 2.2. The sensor data can be from primary sensors and complementary sensors. Common primary sensors are cameras and LiDARs, while IMUs and wheel odometers can be used as complementary sensors. As discussed in section 1.4 modern algorithms utilize multiple sensors to increase robustness. This thesis focuses on stereo inertial SLAM, which uses stereo camera and IMU data.

2.2 Frontend

The frontend of SLAM systems commonly consists of feature extraction, short-term data association, and long-term data association, as seen in the red box in Figure 2.2. SLAM and VO share many of the same components, but long-term data association is not performed in VO. As mentioned in section 1.4.2, VSLAM and VO are divided into direct and indirect methods. This section will only focus on the frontend of indirect methods, also known as feature-based methods.

Feature Extraction

Feature extraction in Visual SLAM extracts points of interest from raw images. Feature extraction is commonly performed for each camera frame by feature extractors such as ORB, SIFT, or SURF (Mur-Artal et al., 2015), (Qin et al., 2017), (Engel et al., 2014). Figure 2.3 shows feature extraction for Kimera-VIO.

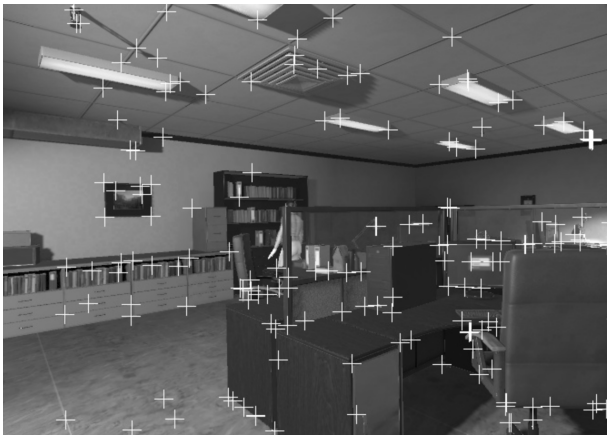

Figure 2.3: Feature detection in Kimera-VIO (Rosinol et al., 2020) on the uHumans2 office dataset.

The extractors typically both detect and describe points of interest. Binary descriptors such as ORB are typically faster to compute but less accurate than Histogram of Gradient (HOG) descriptors such as SIFT (Mur-Artal et al., 2015).

Short-term Data Association

In terms of visual feature-based SLAM, short-term data association is the process of matching observed 2D features with existing 3D map features (Zhou et al., 2016). The matches are also known as *2D-3D matches*. Feature points are considered a match if their descriptors have sufficiently high similarity. Short-term data associations are fed to the backend at a high frequency; they are shown as v_i factors in the factor graph in Figure 2.5, and are further described in section 2.3.

Long-term Data Association

Long-term data association is the process of associating new measurements to old landmarks (Cadena et al., 2016). This is also known as loop closure detection or place recognition. Place recognition is time-consuming as the entire map needs to be considered for a potential match. Consequently, long-term data association is usually run at a lower frequency than short-term. Long-term data association makes SLAM globally consistent and corrects for accumulated drift. The main distinction between VO and SLAM, is that long-term data association is not performed in VO algorithms.

Loop closures are commonly detected by the *Bag of Words* (BoW) approach, such as in ORB-SLAM (Mur-Artal et al., 2015), VINS-Mono (Qin et al., 2017) and Kimera (Rosinol et al., 2020). The loop closures detected by the frontend are fed to the backend as loop closure constraints. These are shown in the factor-graph in Figure 2.5 as c_i factors. The SLAM backend is further described in section 2.3. The typical resulting point-cloud map after a loop closure was previously shown for ORB-SLAM in Figure 1.1.

IMU Preintegration

IMU can be used as a complementary sensor in VSLAM and VO to make VISLAM and VIO (Campos et al., 2020), (Qin et al., 2017). IMU measurements are typically high frequency and can often be well over 100Hz (Qin et al., 2017). It can be challenging to

manage high-frequency data, and processing it incorrectly can often lead to high computation times. In order to reduce the computational complexity of visual-inertial systems, the IMU measurements can be preintegrated (Forster et al., 2015).

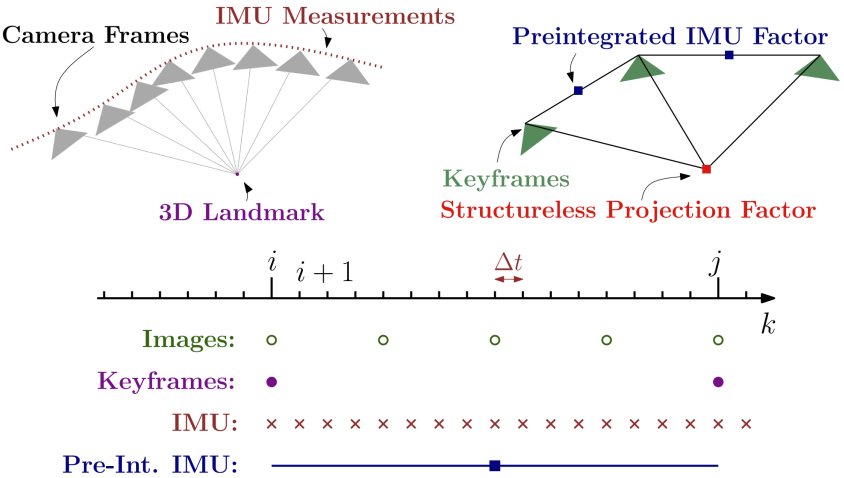

Figure 2.4: Several IMU measurements summarized into a single preintegrated IMU factor. Courtesy of *Robotics: Science and Systems (RSS) conference* (see Forster et al., 2015, Fig. 4 and Fig. 5).

IMU preintegration is the process of integrating IMU measurements between selected keyframes into one pose factor. This reduces the number of IMU factors in the factor graph and, therefore also, the computational complexity. The IMU-factors are typically relative motion constraints, shown as u_i in the factor-graph in Figure 2.5. A visualization of preintegrated IMU factors can be seen in Figure 2.4.

Integration of IMU data is known to accumulate error, leading to drift in the position estimate. As the time between keyframes is usually only a few seconds at max, drift is minimal, and IMU preintegration shows good results (Forster et al., 2015), (Qin et al., 2017), (Campos et al., 2020).

2.3 Backend

While the SLAM frontend associates raw sensor data, the backend uses the abstracted data from the frontend to estimate and output the ego-motion and an environment map (Cadena et al., 2016). The SLAM backend is commonly formulated as a *Maximum a Posteriori* (MAP) estimation problem given by

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}|Z) = \tag{2.1a}$$

$$\underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}) \prod_{k=1}^{m} p(z_k|\mathcal{X}_k) = \tag{2.1b}$$

$$\underset{\mathcal{X}}{\operatorname{argmin}} \sum_{k=0}^{m} \|h_k(\mathcal{X}_k) - z_k\|^2_{\Omega_k} \tag{2.1c}$$

, where $\mathcal{X}$ is the SLAM state, consisting of both the map and the entire trajectory of the camera ego-motion. $\mathcal{X}^*$ is the MAP estimate of the SLAM state. Z is all of the

m measurements , where z_k is a specific measurement at time step k with measurement noise Ω_k. In the case of indirect visual SLAM, z_k would be a detected image feature as visualized in Figure 2.3. $h_k(\mathcal{X}_k)$ is the measurement prediction function, which is often the re-projection function in feature-based SLAM, which projects 3D map point onto the 2D image frame.

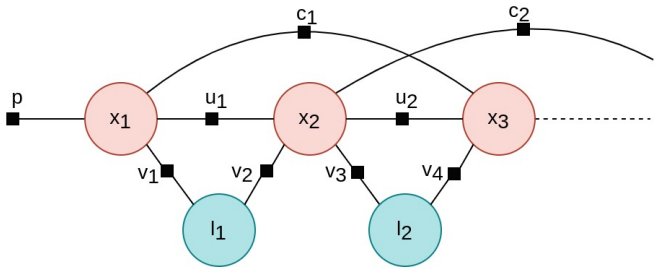

Figure 2.5: The SLAM backend formalized by factor graphs. This figure is recreated and modified version of Fig. 3 in Cadena et al. (2016).

The formalism of factor graphs is often used for the SLAM backend; a figure of this is shown in Figure 2.5. A factor graph consists of nodes (states) and factors (constraints). The factor constraints are output from the frontend, while the states are to be estimated by the backend. Non-linear optimizers such as Levenberg-Marquardt can be used to solve the optimization problem (Mur-Artal et al., 2015).

2.4 SLAM Estimate

The SLAM backend outputs the SLAM estimate, as seen in Figure 2.2. The estimated states are the map of the environment and the ego-motion estimate relative to a fixed world coordinate system. This output is commonly defined as

$$\mathcal{X}^* = \begin{bmatrix} \mathbf{T}_{1:n} \\ \mathbf{x}^w \end{bmatrix}, \tag{2.2}$$

where $\mathbf{T}_{1:n}$ is the ego-motion trajectory consisting of n camera poses $T_i \in SE(3)$ given in the world frame. $\mathbf{x}^w$ is the entire SLAM pointcloud map consisting of several map points $x_i^w \in \mathbb{R}^3$ in the world coordinate frame. The special euclidean group $SE(3)$ describes the camera pose, and consists of position and orientation. $SE(3)$ is defined as

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \;\middle|\; \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\}. \tag{2.3}$$

The special orthogonal group $SO(3)$ is used to describe camera orientation. $SO(3)$ is defined as

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} \;\middle|\; \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \det \mathbf{R} = 1 \right\}. \tag{2.4}$$

3 Tracking in Dynamic Scenes

Dynamic scenes are a significant challenge for current VSLAM systems, which makes it difficult for mobile robots to navigate in non-ideal conditions (Cadena et al., 2016). Section section 3.1 gives an intuition on why dynamic scenes are challenging and discusses the static world assumption common in most SLAM systems today. Section 3.2 covers how RANSAC is used in SLAM to reduce the effect of dynamic objects.

3.1 The Challenge of Dynamic Scenes

This section covers why dynamic scenes are challenging for most SLAM systems today. Section 3.1.1 presents the static world assumption, and section 3.1.2 presents the occurrence of tracking failures when this assumption no longer holds.

3.1.1 The Static World Assumption

The real world is dynamic; examples range from crowded cities to industrial sites with moving vehicles and machines. Most SLAM algorithms today rely on the static world assumption, resulting in severe challenges in dynamic environments (Cadena et al., 2016).

The fundamental equations in VSLAM relies on the environment being static. For feature-based visual SLAM, the re-projection error is minimized in the backend, as previously explained in section 2.3. The re-projection error relies on the projection function, which projects 3D map points to 2D image features. The projection function is also known as the measurement prediction function. This function is no longer able to correctly predict measurements in dynamic scenes.

The feature-based measurement prediction function is defined as

$$h_{ik}(T_{c_i}^w, x_k^w) = \pi_n(inv(T_{c_i}^w) \cdot x_k^w), \tag{3.1}$$

where π_n is the projection function, further explained in Szeliski (2010), $T_{c_i}^w$ is the camera pose in world frame and x_k^w is a 3D map point in world frame.

The projection function in Equation 3.1 does not hold in dynamic scenes. When the 3D map points x_k^w are dynamic, the projection function π_n no longer predicts the 2D feature point measurements z_k. This can cause an erroneous estimation of the SLAM map and camera pose when solving the backend optimization problem in Equation 2.1, which relies on an accurate projection function to predict the measurement z_k.

3.1.2 An Intuition on Ego-motion Estimation in Dynamic Scenes

An intuition of how dynamic scenes affects tracking can be obtained by studying how humans estimate ego-motion in dynamic scenes. Imagine you are estimating your own motion from within a train using visual information alone, as shown in Figure 3.1. When looking out the window at a static scenery, one can utilize the static world assumption and estimate ego-motion without ambiguities. Based on visual information alone, one

can deduce that the train in Figure 3.1 is moving towards the right based on the relative motion of the trees out the window.

Figure 3.1: Consecutive images from left to right, captured from within a subway with a static outside environment. The image sequence shows the subway moving towards the right.

When the world can no longer be assumed static, the problem of estimating ego-motion becomes more difficult. In Figure 3.2 only the relative motion between the two trains can be estimated using visual tracking. The outside train moves left relative to our train. However, if the world can be dynamic, there is no way of knowing whether our train or the outside train is moving. We can observe relative motion, but we cannot know if it corresponds to our own ego-motion or the motion of the train outside the window. This example displays the difficulty in estimating ego-motion when the static world assumption no longer holds.

Figure 3.2: Consecutive images from left to right, captured from within a subway with a dynamic outside environment. From this image sequence alone, it is impossible to deduct the absolute motion of the subways.

3.2 Outlier Rejection using RANSAC

Filtering away outliers using RANSAC is a standard method of improving the robustness of VSLAM in dynamic scenes. This chapter will cover the basics of the RANSAC algorithm in section 3.2.1, section 3.2.2 highlights the use of RANSAC in SLAM, while section 3.2.3 presents the limitations of RANSAC in SLAM in dynamic scenes.

3.2.1 The RANSAC Algorithm

Random sample consensus (RANSAC) is a method of estimating the parameters of a mathematical model from a set of data points that contains outliers. RANSAC is usually used to estimate an inlier set, which is later used by a more accurate estimation method such as *Linear Least Squares* (LSS) or *Levenberg Marquardt* (LM). In the case of optimization-based SLAM, Levenberg Marquardt is commonly used (Szeliski, 2010).

Algorithm 1: RANSAC

for *i = 1 to L* **do**
1: Sample n random data points from M
2: Estimate the parameter vector $\vec{\alpha}$ based on sampled data points
3: Find how many data items (of M) fit the model with parameter vector $\vec{\alpha}$ within a given user tolerance. Call this S.
if *S is big enough* **then**
1: Accept fit, $S_{inlier} = S$, and exit with success
2: **return** S_{inlier}
1: Algorithm exit with fail
2: **return**

The pseudo-code in algorithm 1 describes the RANSAC algorithm. The pseudo-code was inspired by (Wan Bejuri et al., 2015) and (Szeliski, 2010). M is the set of data points, L is the number of RANSAC iterations or models to be tested, S_{inlier} is the inlier set, and finally, $\vec{\alpha} = (\alpha_i ... \alpha_n)$ is the parameters of the mathematical model $f(x; \vec{\alpha})$ to be fit.

3.2.2 Robust Ego-motion Estimation with RANSAC

RANSAC is used in many modern SLAM algorithms to filter out outliers (Mur-Artal et al., 2015), (Mur-Artal and Tardos, 2017), (Campos et al., 2020)(Qin et al., 2017), (Qin and Shen, 2018), (Rosinol et al., 2020). RANSAC is used in the initialization process and the relocalization process in both VINS-Mono (Qin et al., 2017) and ORB-SLAM (Mur-Artal et al., 2015). In Kimera (Rosinol et al., 2020) RANSAC is used to perform geometric verification at every keyframe. By using RANSAC, the erroneous data associations are detected and filtered out, which improves the robustness and accuracy of the tracking.

In SLAM, the mathematical model to be fit is shown in Equation 2.1 and Equation 3.1, which assumes that the world is static as explained in section 3.1.1. When feature points do not confine with the static world assumption, RANSAC can classify the feature points as outliers and discard them. In most cases, this improves the performance of SLAM in dynamic scenes.

Geometric Verification in Kimera

Kimera uses RANSAC to filter away outliers at every keyframe by geometric verification. However, the mathematical model used in RANSAC differs depending on the sensor setup. In the monocular case, the *5-point algorithm* estimates the relative camera motion between two calibrated image frames, using only five point-correspondences (Nister, 2004). By using additional information from stereo or inertial data, the number of required point correspondences can be reduced. In Kimera the *5-point RANSAC* (Nister, 2004) is used for the monocular setup, *3-point* (Horn, 1987) for stereo, *2-point* (Kneip et al., 2011) for

monocular-inertial, and finally, the *1-point RANSAC* for stereo-inertial. Both the 2-point and 1-point methods use the relative orientation information from inertial measurements to aid estimation. This increases the robustness in visually challenging environments such as dynamic scenes (Kneip et al., 2011), (Rosinol et al., 2021).

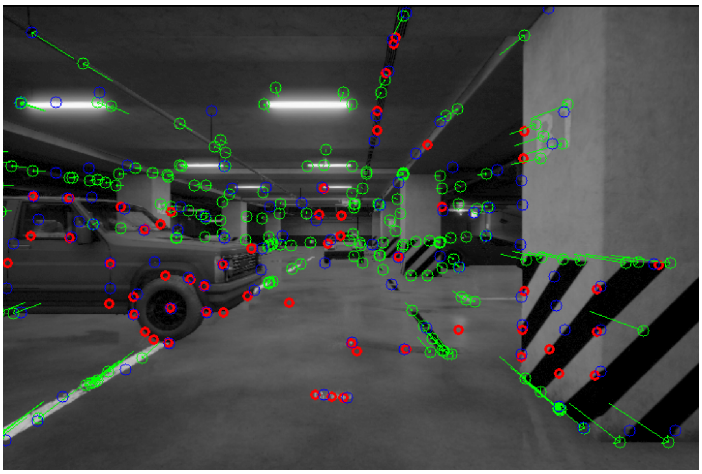

Figure 3.3: Dynamic features are successfully classified as outliers by the 1-Point RANSAC in Kimera-VIO, on the highly dynamic VIODE parking-lot-3 dataset. Tracked feature points are shown in green, and outliers discarded by RANSAC are shown in red.

Figure 3.3 shows the feature points detected by Kimera-VIO in an environment with a moving vehicle. The red points are classified as outliers by the geometric verification and are discarded in further calculations. This shows how RANSAC is used to increase the robustness of VISLAM in dynamic scenes.

3.2.3 The Limitations of RANSAC

Although RANSAC may improve the accuracy of SLAM in dynamic scenes, this is not always the case. RANSAC filters away outliers from dynamic objects if the number of outliers is in the minority. However, severe tracking failures may occur in environments where the majority of the feature points are from dynamic objects (Cui and Ma, 2019).

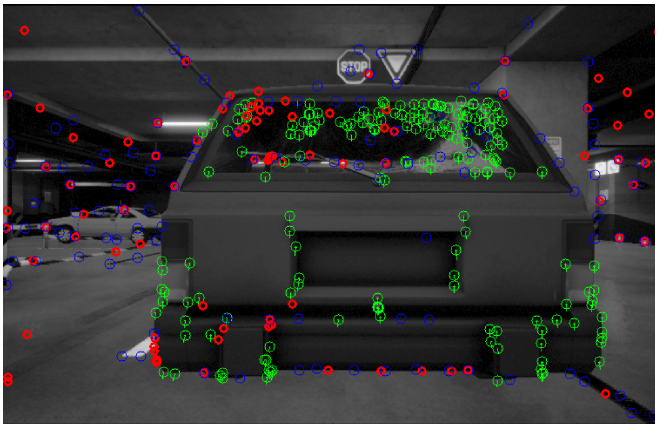

Figure 3.4: Dynamic features are incorrectly classified as inliers by the 1-Point RANSAC in Kimera-VIO, on the highly dynamic VIODE parking-lot-3 dataset. Tracked feature points are shown in green, and outliers discarded by RANSAC are shown in red.

Figure 3.4 shows a highly dynamic scene with a moving vehicle. This figure shows features points detected by Kimera-VIO, where most of the detected features are located on the moving vehicle. We observe that most features on the dynamic vehicle are classified by RANASC as inliers in green, while the feature points in red on the static background are classified as outliers. This misclassification of inliers and outliers may result in severe tracking failures.

4 Evaluating VSLAM

SLAM and VO systems are evaluated based on the quality of the map and estimated ego-motion. Usually, the estimated pose trajectory is evaluated against a ground truth pose trajectory. In order to get an objective evaluation, error metrics such as the Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) are used. These metrics are good at measuring local and global accuracy and consistency, but fall short when evaluating robustness in scenarios with frequent tracking failures. This chapter describes traditional methods of evaluating accuracy in section 4.1, and discusses the need for new metrics to evaluate robustness in section 4.2. Finally, in section 4.3 the differences between simulated and real-world datasets for benchmarking SLAM and VO are presented.

4.1 Evaluating Accuracy

The accuracy of SLAM can be measured by the quality of the estimated ego-motion and map. Generally, only the ego-motion estimate is evaluated, either by visual inspection or by the ATE and RPE metrics. The ego-motion trajectory, also known as pose trajectory, is a trajectory of rigid-body $SE(3)$ poses, as previously explained in section 2.4. When evaluating the SLAM pose, the estimated ego-motion trajectory is compared to a ground truth trajectory. Ground truth ego-motion can easily be produced in simulators and is often estimated by optical tracking or GNSS for real-world datasets, as further described in section 4.3. It is possible to evaluate the quality of the map, as has been done in Rosinol et al. (2021). This is less common but useful if the SLAM map is to be used as part of a more extensive system. In this thesis, only the quality of the ego-motion is considered.

Root Mean Square Error

The *Root Mean Square Error* (RMSE) is a standard method for evaluating the quality of a set of n estimates $\hat{x}_i$ compared to observed values x_i, where $i \in [1, n]$. The RMSE equation is defined as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{x}_i - x_i)^2}. \tag{4.1}$$

4.1.1 Relative Pose Error

The Relative Pose Error (RPE) evaluates local accuracy (Sturm et al., 2012b). This is especially useful when evaluating visual odometry systems, as it can be used to evaluate drift. The drift measured is the relative drift between two time steps instead of a global accumulated drift. The relative pose error RPE_i at time step i is given by

$$RPE_i = (T_i^{-1}T_{i+\Delta})^{-1}(\hat{T}_i^{-1}\hat{T}_{i+\Delta}), \tag{4.2}$$

where the estimated pose and ground truth pose at time step i is given by $\hat{T}_i \in SE(3)$ and $T_i \in SE(3)$, respectively, and Δ represents a fixed time interval. The RMSE RPE is described by

$$RMSE(RPE_{1:m}, \Delta) = \sqrt{\frac{1}{m}\sum_{i=1}^{m} \|trans(RPE_i)\|^2}, \tag{4.3}$$

where $m = n - \Delta$, and n is the number of time steps. $trans(RPE_i)$ is the translational component of the relative pose error RPE_i. When evaluating SLAM, averaging over all possible time intervals Δ is recommended (Sturm et al., 2012b), as shown in

$$RMSE(RPE_{1:n}) = \frac{1}{n}\sum_{\Delta=1}^{n} RMSE(RPE_{1:n}\Delta). \tag{4.4}$$

The RMSE RPE helps give an overview of average drift over the entire data sequence, while RPE_i evaluates the drift at a specific time.

4.1.2 Absolute Trajectory Error

The Absolute Trajectory Error (ATE) evaluates the global consistency of the estimate and describes statistics of the entire trajectory (Sturm et al., 2012b). This error metric represents accumulated error or accumulated drift in a SLAM system. Loop closures in SLAM correct for accumulated drift and makes the SLAM estimate globally consistent, as explained in section 2.2. Thus, ATE is a suitable error metric for evaluating full SLAM solutions, as it takes into account how well the loop closures eliminate accumulated drift. The ATE_i at time step i is defined as

$$ATE_i = T_i^{-1} S \hat{T}_i, \tag{4.5}$$

and the RMSE ATE is defined as

$$RMSE(ATE_{1:n}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|trans(ATE_i)\|^2}, \tag{4.6}$$

where n is the number of time steps, $\hat{T}_i \in SE(3)$ is the estimated trajectory at time step i, and $T_i \in SE(3)$ is the ground truth trajectory at time step i. $S \in SE(3)$ is the rigid-body transform between the estimated and ground truth coordinate systems. The $trans(ATE_i)$ is the translational component of the absolute trajectory error ATE_i.

Alignment

As the ground truth and the estimated trajectories may be described in different coordinate systems, the trajectories must be aligned to estimate ATE. The estimated pose $\hat{T}_i$ needs to be transformed to the ground truth coordinate system by the static transform S, as seen in Equation 4.5. The transform S can be estimated by Least-Squares as described in the method of Horn (Berthold K. P. Horn, 1988) or by the Umeyama algorithm (Umeyama, 1991). The Umeyama algorithm can correct for both translation, rotation, and scale. The $SE(3)$ Umeyama algorithm corrects for translation and rotation, while the $Sim(3)$ Umeyama algorithm additionally corrects for scale. The $Sim(3)$ alignment is usually required for monocular SLAM algorithms where the scale is not estimated as

explained in section 1.4.2. As the scale is estimated in both stereo SLAM and VISLAM, the $SE(3)$ Umeyama is usually sufficient.

4.2 Evaluating Robustness

In recent years robustness has increasingly been in focus when designing SLAM algorithms (Cadena et al., 2016), (Barnes et al., 2018), (Yang and Ramanan, 2021). Robustness is defined as the ability to track in a broad range of environments for extended periods Cadena et al. (2016), which includes tracking in dynamic environments.

Traditional error metrics such as the RMSE ATE, and RMSE RPE do not give a good representation of the performance in challenging datasets (Wang et al., 2020). This is because the metrics can only be calculated on successful parts of the SLAM trajectory. This is seen from the ATE_i and RPE_i equations in Equation 4.2 and Equation 4.5. Both equations are undefined when the estimated pose $\hat{T}_i$ from SLAM is not output.

Less robust SLAM algorithms lose track more often in challenging sequences, and consequently, the ATE_i and RPE_i scores cannot be calculated. Wang et al. argues that as a result, less robust SLAM algorithms could get a higher RMSE ATE and RMSE RPE, as they "give up" difficult sequences. In challenging scenarios, such as dynamic scenes, the RMSE ATE and RMSE RPE may give severely misleading results. This motivates the introduction of a new metric for evaluating robustness.

4.3 Simulated Datasets

Recording a dataset is often necessary to efficiently benchmark the performance of SLAM systems. This is because setting up SLAM to run in real-time on a sensor platform is time-consuming and often infeasible in the early stages of development. Datasets for SLAM can either be recorded in the real world or in simulated environments. This section will describe the advantages and disadvantages of simulated data.

Ground truth ego-motion is commonly estimated by GNSS or optical tracking for real-world datasets. GNSS suffer from inaccuracies and are unavailable indoors. Optical tracking systems use several cameras capturing reflected light from small spherical markers attached to the object to be tracked. Optical tracking systems are limited to small and constrained indoor settings and is expensive and time-consuming to set up. Small indoor datasets commonly rely on optical tracking as in MuSe dataset (Arora et al., 2019) and the OpenLoris dataset (Shi et al., 2020), while larger outdoor datasets commonly depend on GNSS as in the Kitti dataset (Geiger et al., 2012). Due to the nature of simulated datasets, the ground truth is known. The known ground truth ego-motion trajectory from simulated data is a significant advantage over real-world data.

Sensor temporal and spatial synchronization is tedious but necessary when creating real-world datasets. VIO and VISLAM systems are multi-sensor systems requiring visual and inertial sensor data. Inertial and visual data from the IMU and camera are received with a time offset on real-world hardware. There is also a spatial transformation between the inertial and visual sensors. In order to fuse IMU and camera data, both temporal and spatial calibration is required to estimate the time offset and the spatial transformation (Furgale et al., 2013). This synchronization can be complex and cumbersome. There is

no time offset for simulated datasets, and the spatial transformation is known, meaning the temporal and spatial synchronization is only required for real-world datasets.

Sensor artifacts are usually not present in simulated data. Simulators commonly produce perfect camera data at the expense of realism. Real-world cameras suffer from artifacts in some environments, including the rolling shutter effect, motion blur, and over- and underexposure. These effects are commonly not present in simulated environments, which makes simulators less realistic, leading to a false sense of confidence when developing in simulators alone.

Visually realistic simulated environments are challenging to create and necessary to evaluate SLAM accurately on simulated data. A common problem with simulated datasets is the so-called "sim-to-real" gap, which is the difference in the algorithm's performance when transferred from the simulated environment to the real world. A way to overcome this is to create large and diverse simulated datasets that are photo-realistic (Wang et al., 2020). Today simulators are often created in the Unity (Unity Technologies, 2021) or the Unreal (Epic Games, 2022) game engines to simulate real-world environments realistically (Koji et al., 2021) (Rosinol et al., 2021).

The repeatability of simulators is an advantage when evaluating SLAM in specific conditions. An example is the simulated VIODE (Koji et al., 2021) and uHumans2 (Rosinol et al., 2021) datasets, containing dynamic environments. These datasets have static validation datasets, identical to the dynamic datasets, except they have no moving vehicles or people. When simulating, it is possible to perfectly recreate experiments, varying only one condition at a time. This repeatability is difficult in real-world environments, as it is hard to vary only one condition between recordings. Therefore, it is difficult to use real-world datasets to test SLAM in particular conditions, such as dynamic scenes.

Perfect semantic segmentation labels can be acquired from simulators, as was done in both uHumans2 (Rosinol et al., 2021) and VIODE (Koji et al., 2021) datasets. It is important to note that a semantic segmentation algorithm such as SegNet is needed for real-world applications, as further discussed in section 5.5.1. Real-world semantic segmentation methods do not acquire the same accuracy as the simulated ground truth labels. This may lead drop in performance when transferring semantic SLAM algorithms developed in a simulator to the real world.

5 Method and Implementation

This chapter presents Semantic-Kimera-VIO, a modified Kimera-VIO designed for dynamic environments. Firstly in section 5.1 we present the process of choosing a suitable SLAM system to modify. Then in section 5.2 we present the hardware used and the process of setting up the SLAM system. In section 5.3, we discuss how our system was evaluated, and in section 5.4 which datasets were used. Finally, in section 5.5, we present the method of using semantic segmentation to perform outlier rejection in Semantic-Kimera-VIO.

5.1 Choosing a SLAM system

Several prominent VSLAM systems were considered for this project. The systems we considered most strongly are compared in Table 5.1. Firstly, the system had to be open-source, as the code needed to be available to the author in order to be modified. ORB-SLAM2 is an open-source algorithm and was initially considered due to its reputation and popularity. As described in chapter 3, ORB-SLAM2 had been used in several previous projects on improving VSLAM in dynamic scenes. Although suitable, ORB-SLAM2 was not cutting-edge and did not support IMU. ORB-SLAM3 (Campos et al., 2020), VINS-Mono (Qin et al., 2017) and VINS-Fusion (Qin et al., 2019) were considered due to being modern visual inertial systems. VINS-Mono has been used in previous work on displaying VIO difficulties in dynamic scenes (Koji et al., 2021) and therefore seemed fitting. VINS-Fusion supports both IMU and stereo but was not chosen as the code was difficult to work with.

Table 5.1: A comparison between the VSLAM systems considered for this project.

	Year	ROS	Open-source	Stereo/RGB-D	IMU	Permissive Licence
ORB-SLAM	2015	No	Yes	No	No	No
ORB-SLAM2	2016	Yes	Yes	Yes	No	No
VINS-Mono	2017	Yes	Yes	No	Yes	No
VINS-Fusion	2019	Yes	Yes	Yes	Yes	No
ORB-SLAM3	2020	Yes	Yes	Yes	Yes	No
Kimera-VIO	2020	Yes	Yes	Yes	Yes	Yes
Kimera-2021	2021	Yes	No	Yes	Yes	N/A

Kimera-VIO (Rosinol et al., 2020) was eventually chosen for this project, as it is a modern stereo-inertial system that is modular and easy to work with. There are significant advantages to stereo over monocular, as explained in section 1.4.3. Utilizing inertial data is also highly beneficial, as discussed in section 1.4.4. Kimera-VIO is implemented in ROS and does not require a GPU to run. These factors make it easy to contribute to and run the algorithm on an average laptop. Kimera also differentiates itself from other algorithms by providing additional features as presented in section 1.4.5.

In this project, a SLAM system suitable for industrial applications had to be chosen. Therefore, one of the considerations was the license the software was published under. A disadvantage of many of the considered systems in Table 5.1 was the GPLv3 license, which the ORB-SLAM systems, VINS-Mono, and VINS-Fusion were distributed under. The GPLv3 license is less favorable for commercial applications than permissive licenses, such as the BSD license, which Kimera-VIO uses. This is because modifications to software licensed under GPLv3 have to be published openly, which is not always beneficial for companies.

5.2 Setup and Visualization

This section presents the process of setting up Kimera-VIO for this project. This includes describing the hardware used, choosing Kimera parameters, and visualizing Kimera-VIO's output in RViz for evaluation and debugging.

5.2.1 Hardware

The laptop used for this project had specifications given in Table 5.2. The entire project was run on this laptop in real-time using the ROS framework.

Table 5.2: Laptop specifications

Model	Thinkpad X1 Carbon
Memory	15,4GiB
Processor	Intel® Core™ i7-8550U CPU @ 1.80GHz × 8
Graphics	Mesa Intel® UHD Graphics 620 (KBL GT2)
OS Name	Ubuntu 20.04.3 LTS
OS type	64-bit
ROS version	ROS Noetic Ninjemys

5.2.2 Choosing Parameters for Kimera-VIO

Kimera-VIO has a broad range of parameters that can quickly be modified. The parameters were mainly unchanged, except for the IMU preintegration method. In Kimera, the IMU preintegration parameter was initially set to `ImuFactor` but was changed to `CombinedImuFactor` for this project.

As presented in section 2.2, IMU preintegration is the process of summarizing several IMU measurements into one IMU factor to reduce computational costs. In this project, GTSAM `CombinedImuFactor` was chosen, mainly as the accuracy and robustness of the tracking severely improved. There are several differences between the `ImuFactor` and the `CombinedImuFactor`. Firstly, the `CombinedImuFactor` involves both the previous and current time step biases, imposing the biases to be slowly varying. Another difference is that the preintegration covariance includes the noise in the bias estimate. The correlation between the uncertainty of the bias and the preintegrated measurement is also preserved. By changing the IMU preintegration method to `CombinedImuFactor` for Kimera-VIO, the accuracy drastically improved both in dynamic and static sequences.

5.2.3 Robot Operating System

Robot Operating System (ROS) is an open-source middleware for robotics applications (Quigley et al., 2009). ROS was chosen for this project due to being suitable for robotics and real-time tasks. Kimera-VIO supports ROS out of the box, and all the datasets considered for this project was stored in the rosbag format.

In ROS, processes are run in parallel, communicating using the message-passing model. Processes are called nodes, and they publish messages and subscribe to messages on topics. ROS allows for quick development times due to its vast library of packages, debugging tools, and visualization software. Additionally, ROS is highly modular, allowing components to be easily switched out. In this project, the visualization tool RViz and the debugging tool rqt_graph have been used.

5.2.4 Visualization in RViz

RViz was used for visualization of this project. RViz is a 3D visualizer in ROS, allowing visualization of ROS messages and tf-transforms. The RViz visualization in this project was built on the original Kimera visualization, with a few additions.

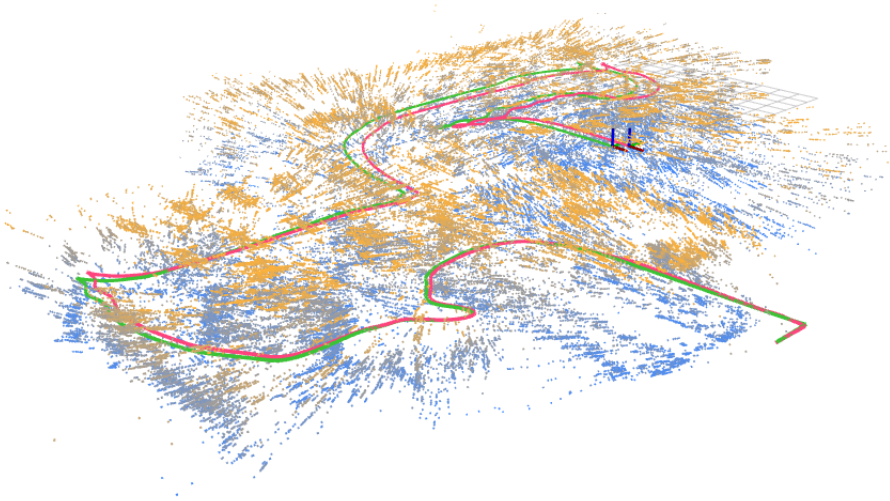

Figure 5.1: RViz visualization of Semantic-Kimera-VIO run on the uHumans2 office 0h dataset. The visualization shows the Semantic-Kimera-VIO ego-motion trajectory (pink), ground truth trajectory (green), and a sparse map (orange / blue).

Figure 5.1 shows the RViz visualization of the Kimera trajectory in pink, the ground truth trajectory in green, and the entire Kimera pointcloud. The Kimera pose trajectory and ground truth pose trajectory were visualized using `TFTrajectory` from the `jsk_rviz_plugins` package (of Tokyo JSK laboratory, 2022). This visualization makes it easy to compare the ground truth trajectory and the Kimera pose trajectory by visual inspection. The point size and decay time of the pointcloud were changed so the entire pointcloud would be easily visible.

Figure 5.2: RViz visualization of Semantic-Kimera-VIO run on the uHumans2 office 0h dataset. The visualization shows the semantic segmentation image overlayed with the debug feature tracking image using the `jsk_rviz_plugins`
.

Figure 5.2 shows Kimera-VIO's tracked visual features overlayed with the semantic segmentation image using `OverlayImage` from the `jsk_rviz_plugins` package (of Tokyo JSK laboratory, 2022). The tracked inlier points are shown in green, new feature tracks in blue, and outliers in red. This visualization can be used to observe whether feature points located on dynamic objects are correctly classified as outliers by Kimera-VIO.

5.2.5 Running Kimera as VIO Instead of VISLAM

As described in section 1.4.5 Kimera consists of several modules. Only the Kimera-VIO module was modified and ran in this project, meaning our system does not perform loop closure or place recognition. Therefore, although our system is a component of a complete VISLAM system, it runs as a pure VIO algorithm in this project. This was done to limit the project's scope and allow for a more straightforward analysis of the results. Our modifications to Kimera-VIO, described in Figure 5.7, were limited to changes to the short-term data association. This allowed for running Kimera-VIO as an isolated system, which made the analysis simpler as loop closures could not affect the results.

5.3 Evaluation

This section describes the specifics of evaluating the robustness and accuracy of Semantic-Kimera-VIO in dynamic environments. We describe how accuracy was evaluated using the Python library *Evo* and how robustness was evaluated by a new metric introduced in this thesis. We then discuss how to account for randomness in each SLAM execution and why ATE and rotational RPE was omitted in the analysis.

5.3.1 Evaluation using Evo

The Python library Evo (Grupp, 2017) was used to evaluate the accuracy of Semantic-Kimera-VIO and Kimera-VIO in ROS. Evo supports standard SLAM error metrics and works well with the rosbag dataset format. This library was used to perform coordinate transform alignment and calculate ATE and RPE, these metrics are previously described in chapter 4. Evo was also used to create the ATE, RPE, and trajectory plots in chapter 6.

5.3.2 Alignment

The Evo implementation of the Umeyama algorithm was used for aligning the ground truth and estimated ego-motion trajectories. Alignment is necessary for visual inspection and calculating the ATE as explained in section 5.3.

The $SE(3)$ Umeyama algorithm was performed, correcting for rotation and translation. Only the beginning of the trajectories was aligned, making it easy to observe drift. The scale was not corrected for as Semantic-Kimera-VIO is a stereo-inertial algorithm which estimates scale. By only aligning translation and rotation, scale errors could be observed from visual inspection.

5.3.3 Evaluating Robustness with Success Rate

In Wang et al. (2020) robustness was evaluated by *Success Rate* (SR). SR was defined as the ratio of non-lost sequences to the number of total sequences. According to Wang et al. this metric better captures the performance of SLAM in challenging environments with frequent tracking failures. A downside of this definition is that it is not clearly defined what a non-lost sequence is. In this project, we define SR as

$$SR := \frac{duration(valid\_trajectory)}{duration(entire\_trajectory)} \in [0, 1] \tag{5.1a}$$

$$valid\_trajectory := (init\_success)\&(\neg lost\_track) \tag{5.1b}$$

This definition somewhat differs from SR in Wang et al. (2020), as SR in this project is a duration ratio, as opposed to the distance ratio in Wang et al. (2020), effectively limiting $SR \in [0, 1]$. duration($valid\_trajectory$) and duration($entire\_trajectory$) is the duration in seconds of the $valid\_trajectory$ and $entire\_trajectory$ respectively. $valid\_trajectory$ is any trajectory produced by the VO or SLAM system; an invalid trajectory would be when the system produced no output. SR is a continuous number between 0 and 1, where $SR = 1.0$ represent that the entire trajectory was valid, while $SR = 0$ represents an initialization failure causing no trajectory. The SR metric is used to evaluate robustness in this project.

5.3.4 Median over Several Executions

As with most modern SLAM systems, which rely on random sampling, the ego-motion and map estimation are non-deterministic. Therefore, Kimera-VIO and Semantic-Kimera-VIO were executed several times on each dataset. Calculating the median RPE and ATE over several executions is a standard method for accurately representing the performance of non-deterministic SLAM algorithms (Campos et al., 2020).

For the VIODE dataset, ten executions were performed. As the uHumans2 sequences were substantially longer than the VIODE sequences, performing ten executions was considered too time-consuming, and five were performed instead. The median of the RMSE RPE over several executions was used as the primary metric to indicate the accuracy of the systems on a particular dataset. For robust systems, the median will accurately represent the behavior (Campos et al., 2020). The median SR over several executions was used to evaluate whether the system was robust.

5.3.5 On using RPE to Evaluate Accuracy

In this project, only the RPE metric is used to evaluate accuracy. As explained in chapter 4, RPE measures relative error, while ATE measures accumulated errors. The

ATE metric was calculated to make our results comparable with other full SLAM solutions but was not analyzed further. ATE values was not used in the analysis as the metric is not suitable for evaluating visual odometry submodules of full SLAM solutions.

Both Kimera-VIO and Semantic-Kimera-VIO are VIO components of a complete SLAM system, as explained in section 5.2.5. The complete Kimera SLAM system contains a loop-closure module that corrects for accumulated drift. The ATE metric is suitable for evaluating full SLAM systems, as it measures accumulated error and, hence also, the loop-closure module's ability to eliminate accumulated drift. Although Kimera-VIO and Semantic-Kimera-VIO are submodules of a complete SLAM system, they were run as standalone VIO algorithms in this project. Therefore, they are evaluated based on their relative drift instead of their accumulated drift, which their loop-closure module may correct for when they are used in a full SLAM system. Because of this, the ATE metric is unsuited for evaluating stand-alone Semantic-Kimera-VIO, and we opt to use the RPE as the primary accuracy metric for our analysis.

Only the translational RPE [m] was estimated in this project, and not the rotational RPE [°]. We chose not to estimate rotational RPE[°] both to limit the scope of the thesis and because the translational and rotational RPE usually correlate well (Sturm et al., 2012b).

5.4 Datasets

There are numerous datasets for benchmarking VSLAM, but many do not contain dynamic objects such as moving people or vehicles. This section will focus on three datasets considered for this project due to their focus on displaying challenging dynamic scenes.

5.4.1 The uHumans2 Dataset

The uHumans2 dataset Rosinol et al. (2021) is a Unity-based simulated dataset stored in the rosbag format. The dataset contains stereo, IMU, and ground truth semantic segmentation data. The dataset contains four different scenarios, an office, a neighborhood, a subway, and an apartment dataset. Each scenario contains three sequences with varying amounts of moving people, including an entirely static sequence. The static sequences are suitable as validation datasets when evaluating the effect of dynamic objects on VSLAM algorithms. Figure 5.3 shows a screenshot of Kimera-VIO running on the uHumans2 office 12h dataset, containing 12 slowly moving people.

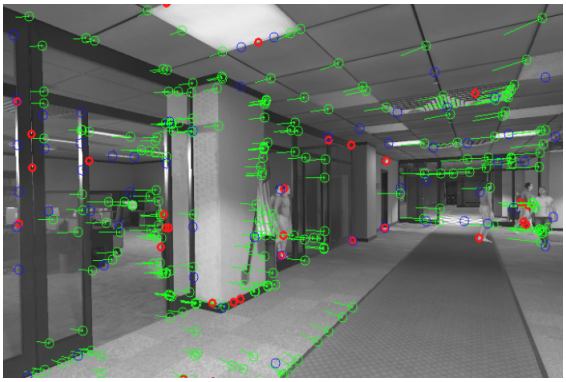

Figure 5.3: The features tracked by Kimera-VIO on the uHumans2 office 12h dataset.

A downside with the uHumans2 dataset is its lack of highly dynamic scenes. The people are walking slowly, and although the dataset contains several moving people, it lacks frames where most of the pixels are from moving people.

5.4.2 The VIODE Dataset

The VIODE dataset Koji et al. (2021) is an Unreal-engine-based simulated dataset stored in the rosbag format. The dataset was created with VIO in mind and outputs both stereo data, IMU data, and ground truth semantic segmentation images.

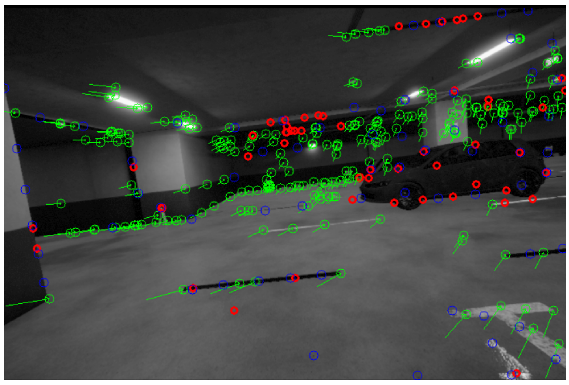

Figure 5.4: The features tracked by Kimera-VIO on the VIODE parking-lot 3-high dataset.

The dataset contains three different scenarios: a parking lot, a city during daytime, and a city during nighttime. Each scenario contains four sequences with varying levels of dynamic vehicles, including an entirely static sequence for each scenario. The highly dynamic VIODE sequences contain quickly moving vehicles covering large frame portions. The highly dynamic sequences also contain stationary vehicles. The nighttime and daytime city datasets are equivalent except for the lighting conditions, making them suitable for testing VSLAM in varying lighting conditions.

5.4.3 The TUM RGB-D Dataset

The TUM RGB-D Sturm et al. (2012a) dataset is a real-world dataset commonly used for evaluating VSLAM in dynamic scenes (Yu et al., 2018), (Cui and Ma, 2019), (Mur-Artal and Tardos, 2017). It contains numerous scenarios, including dynamic environments with moving people in an office setting. This dataset contains depth data and ground truth ego-motion from optical tracking, but some sequences lack IMU data.

Figure 5.5: Snapshots of the TUM-RGB-D freiburg3_walking_xyz dataset (Sturm et al., 2012a), showing the RGB image (left) and depth image (right).

Figure 5.5 shows a screenshot of the camera image and depth image of the freiburg3_walking_xyz rosbag.

5.4.4 Datasets used in this Project

The datasets considered for this project are compared in Table 5.3. The VIODE and uHumans2 datasets are simulated stereo-inertial datasets used in this project. Both datasets contain different environments, where each environment has dynamic sequences. These datasets also contain static sequences, which are suitable as validation data.

Table 5.3: The datasets considered for this project.

Dataset	Dynamic & Static Data	Real World	IMU	Used
VIODE parking-lot	Yes	No	Yes	Yes
VIODE city-day	Yes	No	Yes	Yes
VIODE city-night	Yes	No	Yes	No
uHumans2 office	Yes	No	Yes	Yes
uHumans2 subway	Yes	No	Yes	Yes
uHumans2 apartment	Yes	No	Yes	Yes
uHumans2 neighbourhood	Yes	No	Yes	No
TUM RGB-D freiburg3_sitting	Yes	Yes	No	No

The VIODE city-night and the uHumans2 neighborhood datasets were not used. The city_night dataset would be excessive, as it is equivalent to the city-day dataset except for the lighting and does not further add any additional information on dynamic scenes. Comparing Semantic-Kimera-VIO and Kimera-VIO on the neighborhood dataset was impractical, as they often failed to initialize, even on the static validation dataset. The TUM RGB-D is a much-used dataset for evaluating VSLAM in dynamic scenes but was incompatible with Kimera-VIO due to lacking IMU data in the dynamic sequences.

Table 5.4: An extensive list of all the dataset sequences used in this project.

Dataset	Rosbag Name	Description
VIODE parking-lot-0	parking_lot/0_none.bag	No dynamic objects
VIODE parking-lot-1	parking_lot/1_low.bag	A few dynamic and static vehicles
VIODE parking-lot-2	parking_lot/2_mid.bag	Some dynamic and static vehicles
VIODE parking-lot-3	parking_lot/3_high.bag	Several dynamic and static vehicles
VIODE city-day-0	city_day/0_none.bag	No dynamic objects
VIODE city-day-1	city_day/1_low.bag	A few dynamic and static vehicles
VIODE city-day-2	city_day/2_mid.bag	Some dynamic and static vehicles
VIODE city-day-3	city_day/3_high.bag	Several dynamic and static vehicles
uHumans2 office 0h	uHumans2_office_s1_00h.bag	No people
uHumans2 office 12h	uHumans2_office_s1_12h.bag	Twelve slowly moving people
uHumans2 apartment 0h	uHumans2_apartment_s1_00h.bag	No people
uHumans2 apartment 2h	uHumans2_apartment_s1_02h.bag	Two slowly moving people.
uHumans2 subway 0h	uHumans2_subway_s1_00h.bag	No people.
uHumans2 subway 36h	uHumans2_subway_s1_36h.bag	36 slowly moving people.

Table 5.4 shows an extensive list of all the dataset sequences used in this project. This list displays the original rosbag name of all the dataset sequences and describes the number of dynamic vehicles and people in each sequence. These datasets display various environments, including outdoor and indoor settings with moving people or vehicles. The main downside with the selected datasets is that neither are from real-world data; this can cause the challenges previously described in section 4.3.

5.5 Semantic-Kimera-VIO

Semantic-Kimera-VIO is our modified Kimera-VIO which uses semantic segmentation images to filter out potentially dynamic outliers. The process of acquiring semantic segmentation images is described in section 5.5.1, while section 5.5.2 describes the method of filtering out image feature points from dynamic objects. Finally, in section 5.5.3, the specific modifications done to Kimera-VIO are described.

5.5.1 Semantic Segmentation

Semantic segmentation is the process of assigning semantic labels to every pixel in an image, as shown in Figure 5.6. Semantic segmentation differs from most object detection algorithms such as Yolo (Redmon et al., 2015) which outputs bounding boxes. Semantic

segmentation is a central component of filtering out dynamic outliers in VSLAM using the method described in section 5.5.2. The pixel-level labeling allows a detailed understanding of the scene, which is crucial when detecting whether feature points lie within the contour of a potentially dynamic object or not.

Figure 5.6: Image Segmentation by the Tensorflow's implementation of Mask R-CNN (He et al., 2017). Photo taken by the author.

Traditionally methods such as decision forests and support vector machines were used to perform semantic segmentation, but modern methods have gravitated towards Convolutional Neural Networks (CNN) (Thoma, 2016). A popular CNN-based semantic segmentation network is SegNet (Badrinarayanan et al., 2015), which has commonly been used to filter out dynamic outliers in SLAM applications (Yu et al., 2018), (Cui and Ma, 2019). The results of SegNet on a traffic data are shown in Figure 5.6. This figure shows inaccuracies such as the mislabeled car in the background and the semantic mask not perfectly fitting the objects.

As discussed in section 4.3, the simulated VIODE and uHumans2 datasets produce perfect semantic segmentation labels, shown for VIODE in Figure 5.7. As this project relies on these datasets, a semantic segmentation algorithm was deemed unnecessary. However, a semantic segmentation algorithm such as SegNet would be necessary for real-world applications. SegNet does not produce the same accuracy as the simulated ground truth labels. Inaccuracies can be seen for SegNet in Figure 5.6, which is worth noting when benchmarking semantic SLAM on simulated data, as is done in this project.

5.5.2 Semantic Outlier Removal

Semantic-Kimera-VIO filters out all feature points located on potentially dynamic objects. Specifically, we modified Kimera-VIO to input a semantic segmentation image and filter out feature points on semantically detected vehicles or humans, as they could be dynamic. The result is shown in Figure 5.7, where all detected feature points within the car contour are discarded.

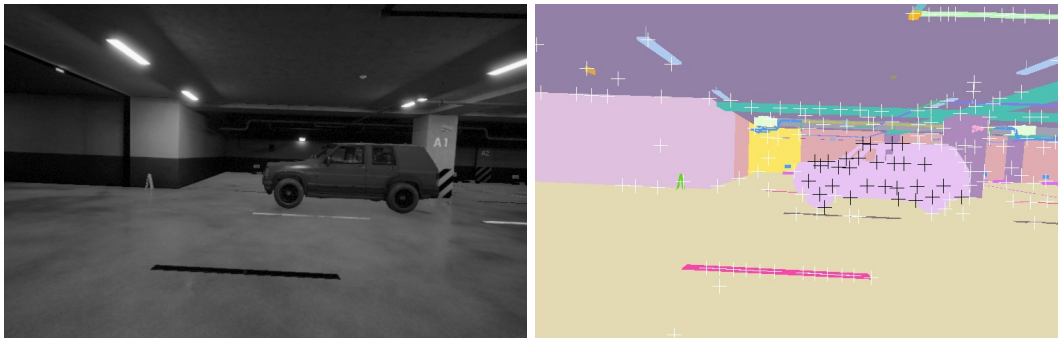

Figure 5.7: The left camera image (left) and the semantic segmentation image (right). Detected feature points are visualized as cross markers on the semantic segmentation image; inliers in white and discarded outliers in black.

The pseudo-code of the semantic outlier removal algorithm is shown in algorithm 2. This algorithm is applied after feature detection in the Kimera-VIO frontend and is relatively straightforward. A detected feature point is considered a semantic outlier if it is located within an area semantically labeled as a vehicle or person. Semantic outliers are discarded, while inliers are added to the list of key points to be tracked at the next time step. Figure 5.7 shows the discarded semantic outliers in black and the static inliers in white.

Algorithm 2: Semantic Outlier Removal

```
for feature_point in detected_features do
    if (feature_point in human) or (feature_point in vehicle) then
        Skip feature_point
    else
        Add feature_point to list of keypoints
        Registrer a landmark
```

It is important to note that this algorithm makes two substantial simplifications. Firstly, only vehicles and humans are considered potentially dynamic. Consequently, other objects, such as animals, vessels, aircraft, or other machines, would be considered static and would not be filtered out. Secondly, all feature points on humans and vehicles are filtered out, even feature points on static humans and vehicles. For this reason, high-quality feature points could be filtered out, which could be a prominent issue in environments with numerous static people or vehicles.

5.5.3 Modifying the Data Flow of Kimera

Kimera-VIOs data flow was changed to allow the frontend access to the semantic segmentation image (Dense 2D Semantics). The original Kimera-VIO's pipeline was previously shown in Figure 1.5, while Figure 5.8 shows the pipeline of Semantic-Kimera-VIO. The difference in the pipeline diagram is the pink arrow representing the flow of the segmentation image to the VIO frontend.

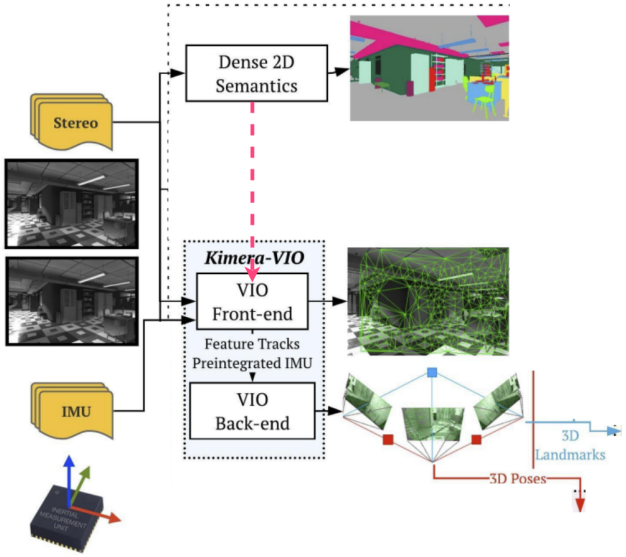

Figure 5.8: Module diagram of Semantic-Kimera-VIO. The pink arrow represents message passing of the 2D semantic image to the VIO frontend.

Modifications in ROS

The Kimera-VIO rosnode was modified to input the semantic segmentation image. The modification was done by subscribing to the semantic segmentation image topic in ROS. The original Kimera-VIO setup is shown in Figure 5.9, while our setup for Semantic-Kimera-VIO is shown in Figure 5.10. The node `/kimera_vio_ros/kimera_vio_ros_node` is Kimera-VIO in Figure 5.9, but Semantic-Kimera-VIO in Figure 5.10.

Figure 5.9: The ROS architecture of the original *Kimera-VIO* with all relevant ROS nodes (circles) and topics (rectangles). The figure was created using ROS rqt_graph.

Semantic-Kimera-VIO differs from the original Kimera-VIO as the segmentation image is a direct input to the VIO frontend, which allows the semantic outlier removal described in section 5.5.2 to be performed. Both Figure 5.9 and Figure 5.10 was created using ROS rqt_graph, a tool visualizing active ROS nodes and topics.

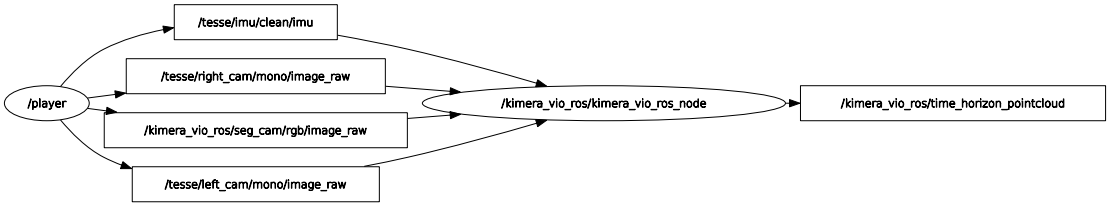

Figure 5.10: The ROS architecture of *Semantic-Kimera-VIO* with all relevant ROS nodes (circles) and topics (rectangles). The figure was created using ROS rqt_graph.

Time Synchronization

The camera images and the semantic segmentation images are received asynchronously in ROS. The semantic outlier removal algorithm uses image feature points from the left image together with semantic information from the segmentation image. Therefore, the semantic segmentation and left image need to be time-synchronized. Both images are added to two separate queues as the ROS node receives them. The queues stores both the images and the timestamp at receival. At each ROS spin, the most recent left image is matched with the segmentation image with the nearest timestamp. This is the same method used for time synchronizing IMU and camera data in Kimera-VIO.

Difference from Previous Work

In section 1.4.5, we presented SLAM systems specifically designed for dynamic environments, such as DS-SLAM, SOF-SLAM, and VINS-Mask. These systems share similarities with our method, as they all filter away outliers using semantic segmentation in the frontend. DS-SLAM and SOF-SLAM are based on the RGB-D mode of ORB-SLAM2 and VINS-Mask on the monocular inertial VINS-Mono. One of the main novelties of our system is that it is based on Kimera.

There are several differences between Kimera, ORB-SLAM2, and VINS-Mono, but the main advantage of Kimera is that it is a stereo inertial algorithm. Advantages of stereo inertial SLAM are previously described in section 1.4, and advantages of stereo inertial RANSAC for robustness in dynamic scenes are explained in section 3.2.2. Other reasons why Kimera was chosen are previously explained in section 5.1.

6 Results and Discussions

This chapter presents the results of Semantic-Kimera-VIO compared to the original Kimera-VIO in dynamic environments. In section 6.1 an overview of the results on both the VIODE and uHumans2 datasets is presented. This overview contains tables of median RMSE RPE, median RMSE ATE, and median SR of both Semantic-Kimera-VIO and Kimera-VIO in static and dynamic scenes. section 6.2 and section 6.3 further go into detail on how the performance of Semantic-Kimera-VIO improved in the highly dynamic VIODE scenes. Finally, in section 6.4 we discuss why Semantic-Kimera-VIO does not improve on the uHumans2 dataset.

6.1 Quantitative Results

This section presents the quantitative results of Kimera-VIO and Semantic-Kimera-VIO on the VIODE and uHumans2 datasets. The metrics used for evaluation are previously described in section 5.3. The median RMSE RPE, median RMSE ATE and median SR for both Kimera versions on each dataset is shown in Table 6.1, Table 6.2 and Table 6.3 respectively. The median was calculated from ten executions for the VIODE dataset and five for the uHumans2 dataset.

Table 6.1: The **median RMSE RPE [m]** over N executions of Kimera-VIO and Semantic-Kimera-VIO. N being five for the uHumans2 datasets and ten for the VIODE datasets. $\% - diff$ shows the change in the RMSE RPE of Semantic-Kimera-VIO relative to Kimera-VIO.

Dataset	Kimera-VIO	Semantic-Kimera-VIO	%-diff
VIODE parking-lot-0	0.076	0.069	-8.92%
VIODE parking-lot-1	0.069	0.068	-2.26%
VIODE parking-lot-2	0.182	0.103	-43.23%
VIODE parking-lot-3	0.18	0.096	-47.0%
VIODE city-day-0	0.256	0.252	-1.78%
VIODE city-day-1	0.288	0.273	-5.34%
VIODE city-day-2	0.283	0.251	-11.39%
VIODE city-day-3	0.504	0.283	-43.7%
uHumans2 office 0h	0.00216	0.00212	-1.92 %
uHumans2 office 12h	0.00249	0.00252	1.29 %
uHumans2 apartment 0h	0.00238	0.00243	2.21 %
uHumans2 apartment 2h	0.0258	0.0259	0.089 %
uHumans2 subway 0h	0.00727	0.00764	5.08 %
uHumans2 subway 36h	0.0457	0.0457	- 0.00547 %

Table 6.2: The **median RMSE ATE [m]** over N executions of Kimera-VIO and Semantic-Kimera-VIO. N being five for the uHumans2 datasets and ten for the VIODE datasets. $\% - diff$ shows the change in the RMSE ATE of Semantic-Kimera-VIO relative to Kimera-VIO.

Dataset	Kimera-VIO	Semantic-Kimera-VIO	%-diff
VIODE parking-lot-0	0.547	0.517	-5.44%
VIODE parking-lot-1	0.495	0.534	7.82%
VIODE parking-lot-2	6.139	1.311	-78.64%
VIODE parking-lot-3	6.193	0.785	-87.33%
VIODE city-day-0	9.397	8.078	-14.03%
VIODE city-day-1	9.403	8.713	-7.34%
VIODE city-day-2	6.944	8.23	18.53%
VIODE city-day-3	12.167	2.508	-79.39%
uHumans2 office 0h	0.744	0.66	-11.3 %
uHumans2 office 12h	1.07	1.25	16.8 %
uHumans2 apartment 0h	0.131	0.14	6.72 %
uHumans2 apartment 2h	0.131	0.115	-12.1 %
uHumans2 subway 0h	5.07	4.47	-11.9 %
uHumans2 subway 36h	0.511	0.653	27.9 %

Table 6.3: The **median SR** over N executions of Kimera-VIO and Semantic-Kimera-VIO. N being five for the uHumans2 datasets and ten for the VIODE datasets. $\% - diff$ shows the change in the SR of Semantic-Kimera-VIO relative to Kimera-VIO.

Dataset	Kimera-VIO	Semantic-Kimera-VIO	%-diff
VIODE parking-lot-0	0.988	0.99	0.187%
VIODE parking-lot-1	0.988	0.99	0.195%
VIODE parking-lot-2	0.988	0.99	0.17%
VIODE parking-lot-3	0.991	0.99	-0.118%
VIODE city-day-0	0.74	0.775	4.74%
VIODE city-day-1	0.721	0.799	10.8%
VIODE city-day-2	0.745	0.704	-5.54%
VIODE city-day-3	0.772	0.401	-48.1%
uHumans2 office 0h	0.999	0.957	-4.15%
uHumans2 office 12h	0.999	0.978	-2.11%
uHumans2 apartment 0h	0.995	0.995	-0.0437%
uHumans2 apartment 2h	0.997	0.997	0.0582%
uHumans2 subway 0h	0.999	0.999	-0.005%
uHumans2 subway 36h	0.909	0.848	-6.76%

6.1.1 Increased Accuracy on the VIODE Dataset

Based on the median RMSE RPE on the VIODE dataset, shown in Table 6.1, Semantic-Kimera-VIO increases accuracy in dynamic scenes. The increased accuracy is observed as the RMSE RPE decreased from $0.18[m]$ to $0.096[m]$ on the highly dynamic VIODE parking-lot-3 dataset. This difference corresponds to a 47% decrease in median RMSE RPE from Kimera-VIO to Semantic-Kimera-VIO.

Similar results are shown on the VIODE city-day dataset, where the RMSE RPE decreased from $0.504[m]$ to $0.283[m]$ on the highly dynamic VIODE city-day-3 dataset. This corresponds to a 43.7% decrease in RMSE RPE. These results show that Semantic-Kimera-VIO decreases its RMSE RPE by $\approx$ 45% on average on the highly dynamic VIODE dataset compared to Kimera-VIO. The drastically decreased RMSE RPE indicates that Semantic-Kimera-VIO improves the performance over Kimera-VIO in highly dynamic environments.

The median RMSE RPE is also slightly decreased for the static and lowly dynamic datasets; VIODE parking-lot-0, parking-lot-1, city-day-0, and city-day-1. The median RMSE RPE of Semantic-Kimera-VIO is decreased by $\approx 4.6\%$ on average for these datasets compared to Kimera-VIO. $\approx 4.6\%$ is significantly lower than the $\approx$ 45% decrease on the dynamic VIODE datasets. This difference is to be expected as our modification is intended to increase performance in dynamic scenes and acquire near equal performance in static scenes.

To conclude, the RMSE RPE is drastically decreased for Semantic-Kimera-VIO compared to Kimera-VIO on the highly dynamic VIODE sequences, and slightly decreased on the lowly dynamic sequences. This indicates that Semantic-Kimera-VIO may have an increased accuracy over the original Kimera-VIO in dynamic environments. In section 6.2, we further investigate why the accuracy seems to increase.

6.1.2 Evaluating SR on the VIODE Dataset

Although Semantic-Kimera-VIO shows an increased accuracy on the VIODE dataset in terms of RMSE RPE, the SR metric also needs to be considered. From Table 6.3, we observe that SR is > 0.98 for Kimera-VIO and Semantic-Kimera-VIO on the entire VIODE parking-lot dataset, meaning they both output ego-motion for $> 98\%$ of the duration of the dataset. The high SR indicates that both Kimera versions have high robustness on both the static and highly dynamic sequences in the parking-lot dataset.

The SR metric shows a different story for the city-day dataset. The city-day dataset is challenging both for Kimera-VIO and Semantic-Kimera-VIO, even in the static case. The difficulties can be observed from the SR being 0.74 and 0.775 respectively for Kimera-VIO and Semantic-Kimera-VIO on the static VIODE city-day-0 dataset. In other words, both Kimera versions managed to track $\approx 75\%$ of the trajectory on the static city-day dataset. The relatively low SR indicates that both Kimera versions lack robustness on the static VIODE city-day dataset.

On the highly dynamic VIODE city-day-3 dataset, we see that Semantic-Kimera-VIO is a lot worse than Kimera-VIO. The poor performance is shown as the SR decreased 48.1% from Kimera-VIO to Semantic-Kimera-VIO. On this dataset, Semantic-Kimera-VIO has an SR of 0.401, meaning it only outputs ego-motion for 40.1% of the duration of the dataset. This is further investigated in section 6.3.

Although the RMSE RPE values indicate that the performance of Semantic-Kimera-VIO surpasses Kimera-VIO on the VIODE dataset, the SR metric shows a different story. For most of the VIODE sequences, both Kimera versions have near equal SR. However,

this is not the case in the city-day-3 dataset. On the city-day-3 sequence, the SR of Semantic-Kimera-VIO is severely decreased, indicating that its increased accuracy may come at the cost of a decreased robustness in certain environments.

6.1.3 Results on the uHumans2 Dataset

The performance of Semantic-Kimera-VIO seems almost identical to Kimera-VIO on the uHumans2 dataset from the median RMSE RPE values. The median RMSE RPE values in Table 6.1 show a slight difference between Kimera-VIO and Semantic-Kimera-VIO of less than $0.0001[m]$ on all uHumans2 datasets. To our surprise, the original Kimera-VIO performs well on all the uHumans2 datasets, including the sequences with several moving people. This is seen from the low median RMSE RPE in the range of $(0.002[m], 0.046[m])$ on the uHumans2 dataset as opposed to $(0.18[m], 0.504[m])$ on the VIODE datasets.

There is a slight difference in SR between Kimera-VIO and Semantic-Kimera-VIO shown in Table 6.3. The SR values show that Semantic-Kimera-VIO performs slightly worse on the uHumans2 dataset. The average percentage difference in SR over all the uHumans2 datasets is -2.15%, meaning Semantic-Kimera-VIO is $\approx 2\%$ worse than Kimera-VIO in terms of SR on the uHumans2 dataset. A reason for this could be a failure in time-synchronizing between the left image and the segmentation image in the Semantic-Kimera-VIO, which could lead to system failure. The time-synchronization method was described in section 5.5.3. The $\approx 2\%$ decrease in SR on the uHumans2 dataset is not further looked into but should be further investigated in future work.

To conclude, the median SR and the median RMSE RPE are near equal for both Kimera versions on the uHumans2 dataset. The SR is very high, and the RMSE RPE is very low on all sequences. This indicates that both versions have a high performance on the uHumans2 dataset. We further discuss why the original Kimera-VIO performed well on the dynamic uHumans2 datasets when it performed poorly on the dynamic VIODE datasets in section 6.4.

6.2 Increased Accuracy on the Highly Dynamic VIODE Dataset

In section 6.1 we remarked that Semantic-Kimera-VIO has an increased accuracy compared to Kimera-VIO on both of the highly dynamic VIODE datasets. This section compares the results of two individual executions of Kimera-VIO and Sematic-Kimera-VIO on the VIODE parking-lot-3 dataset to further understand why the accuracy is improved.

6.2.1 Decreased RPE and ATE

Figure 6.1 shows the ego-motion trajectory of Kimera-VIO (left) and the trajectory of Semantic-Kimera-VIO (right) on the VIODE parking-lot-3 dataset. The ground truth trajectory is visualized as a dotted line in gray. We can visually observe that the Semantic-Kimera-VIO trajectory is similar to the ground truth trajectory. In contrast, Kimera-VIOs trajectory does not coincide well with ground truth.

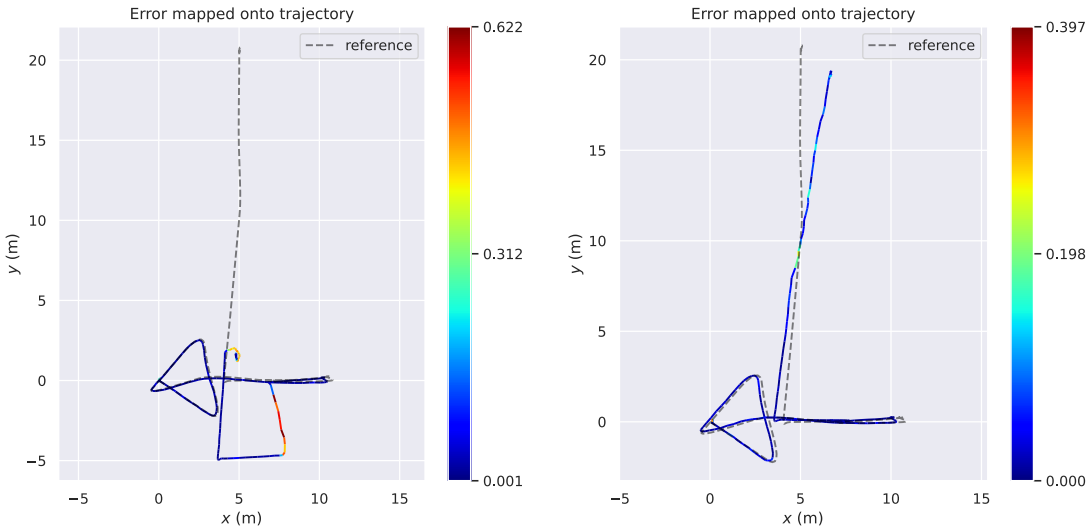

Figure 6.1: The trajectories of Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the VIODE parking-lot-3. The ground truth trajectory is shown as a dotted line in gray. The RPE is mapped onto the trajectories.

The trajectory color represents the RPE value at that point, dark blue corresponding to a low RPE and red to a high RPE. From the RPE colorization, we observe that a segment of Kimera-VIOs trajectory is colorized in dark red. The dark red segment shows that a section of the VIODE parking-lot-3 dataset is particularly challenging for Kimera-VIO.

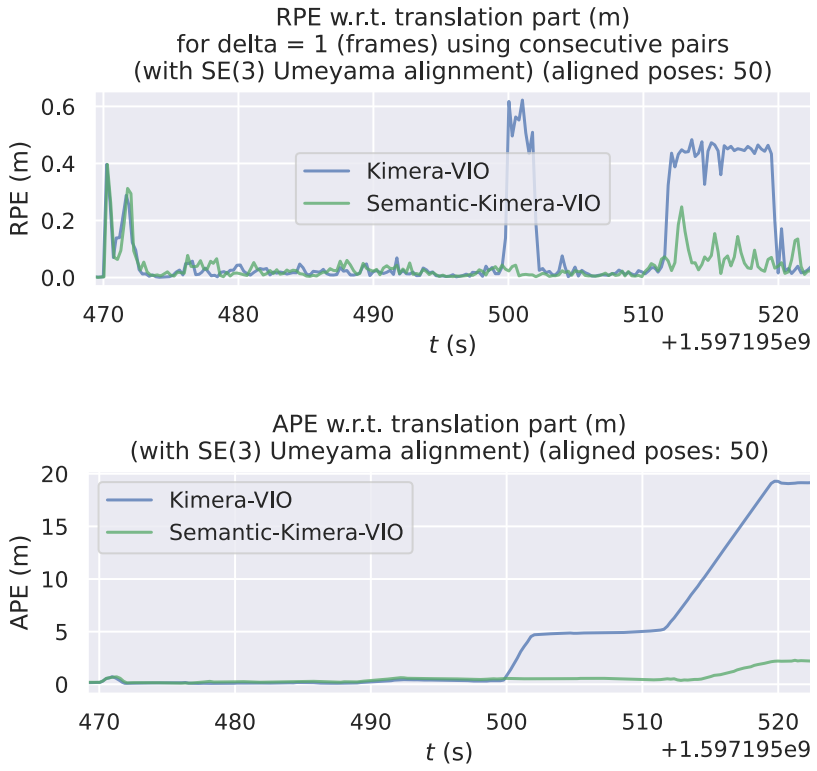

Figure 6.2: A plot of RPE (top) and ATE (bottom) over time of Kimera-VIO (blue) and Semantic-Kimera-VIO (green) on the VIODE parking-lot-3 dataset.

Figure 6.2 shows the RPE over time, RPE(t), of Kimera-VIO in blue and Semantic-Kimera-VIO in green. This plot clearly shows that the dataset is challenging for Kimera-VIO at $t(s) \approx 500$ seconds and $t(s)$ between 510 and 520 seconds. During most of the dataset, the RPE(t) of Kimera-VIO is near zero, but spikes to around 0.5(m) at $t(s) \approx 500$ seconds and $t(s)$ between 510 and 520 seconds. During most of the dataset, Semantic-Kimera-VIO and Kimera-VIO show similar low RPE values. However, during the challenging periods, Semantic-Kimera-VIO severely outperforms Kimera-VIO, showing a low RPE below 0.1(m). We can further look into what is happening during these challenging periods by analyzing the tracked visual features.

6.2.2 Semantic-Kimera-VIO Excels when RANSAC Fails

Figure 6.3 shows a snapshot of the features tracked by Kimera-VIO (left) and Semantic-Kimera-VIO (right). These snapshots are from $t(s) \approx 515$ seconds, which corresponds to Kimera-VIOs second RPE spike shown in Figure 6.2. The red feature points are classified as outliers by Kimera, while the green points are classified as inliers.

We observe that Kimera-VIO (left) classifies static points in the background as outliers, while the majority of the feature points on the moving car are classified as inliers. Kimera-VIO misclassifies inliers and outliers, which results in the severe tracking error shown as a red segment of the left graph in Figure 6.1. The tracking error occurs as motion can only be estimated from static feature points due to the static world assumption, as explained in section 3.1.1.

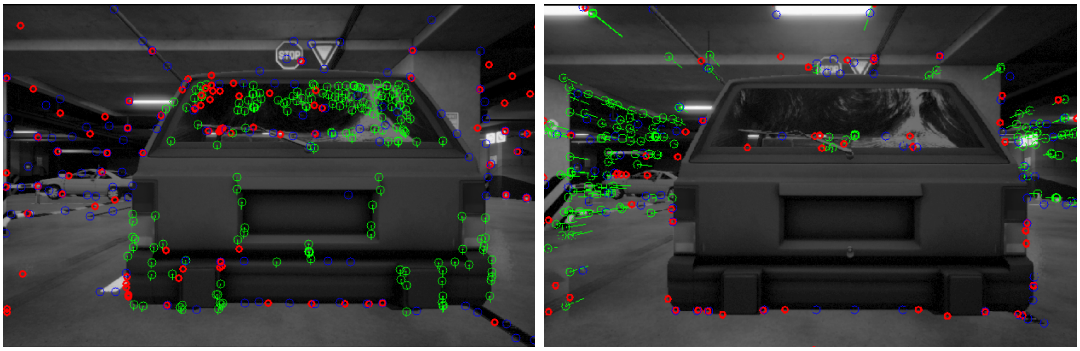

Figure 6.3: Comparison between the feature tracking in Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the VIODE parking lot 3_high dataset at $t(s) \approx 515$ seconds.

As previously explained in section 3.2.3, RANSAC is prone to failures if the majority of data points are outliers. The original Kimera-VIO uses geometric verification based on RANSAC as described in section 3.2.2 to filter out dynamic outliers. RANSAC estimates a mathematical model and outputs an inlier set containing the maximum amount of data points confining with the model. When most feature points are from dynamic objects, the relative motion between the camera and the dynamic object will be estimated instead of the ego-motion of the camera relative to the world. This may be why Kimera-VIO fail to track in this sequence.

For Semantic-Kimera-VIO, there are no tracked feature points on the moving vehicle, as seen in Figure 6.3. Most of the static background features are correctly classified as inliers shown in green. Semantic-Kimera-VIO correctly classifies static inliers, resulting in accurate tracking shown in the right trajectory in Figure 6.1. These results indicate the benefit of semantic outlier removal in highly dynamic environments when RANSAC otherwise would fail.

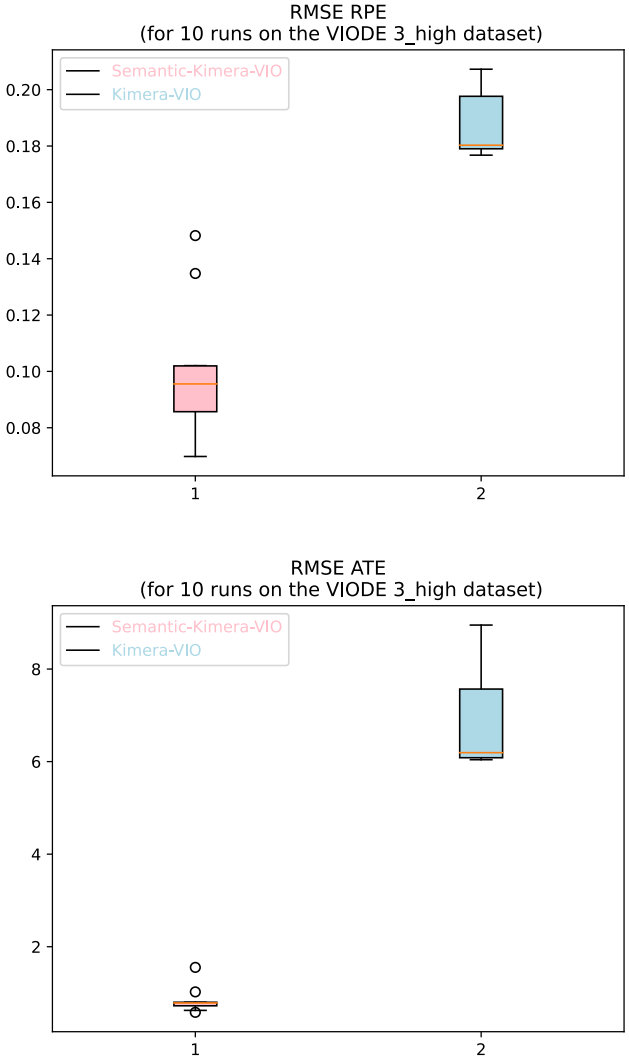

Figure 6.4: Boxplot of Semantic-Kimera-VIO (pink) and Kimera-VIO (blue) on the VIODE parking-lot-3 dataset over ten executions.

In conclusion, Semantic-Kimera-VIO shows an increased accuracy in terms of RPE on the parking-lot-3 dataset. This is observed from visual inspection in Figure 6.1, and in Figure 6.2. To account for randomness in individual runs, the RMSE RPE and RMSE ATE was calculated from ten executions, shown in the boxplot in Figure 6.4, where Semantic-Kimera-VIO is shown in pink and Kimera-VIO in blue. The boxplot shows that the RMSE RPE of Semantic-Kimera-VIO is decreased relative to Kimera-VIO, while the RPE variance remained almost identical. These RMSE RPE values may imply that the addition of outlier removal using semantic segmentation improves accuracy in highly dynamic scenes.

6.3 Decreased Robustness on the Highly Dynamic VIODE city-day Dataset

In section 6.1 we remarked that although Semantic-Kimera-VIO showed an increased accuracy on the highly dynamic VIODE datasets, the SR showed a decreased robustness on the city-day-3 sequence. In this section, individual executions of Kimera-VIO and Semantic-Kimera-VIO on the city-day-3 dataset are compared to investigate why the SR is decreased for Semantic-Kimera-VIO.

6.3.1 Increased Accuracy but Decreased Robustness

Both Kimera-VIO and Semantic-Kimera-VIO do not manage to output an accurate and robust ego-motion on the city-day-3 dataset. Figure 6.5 shows the estimated ego-motion from Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the VIODE city-day-3 dataset. The ground truth trajectory is shown as a dotted gray trajectory. The figure shows that Semantic-Kimera-VIO (right) quickly stops tracking, while Kimera-VIO (left) tracks for longer with a low accuracy and eventually also stops tracking.

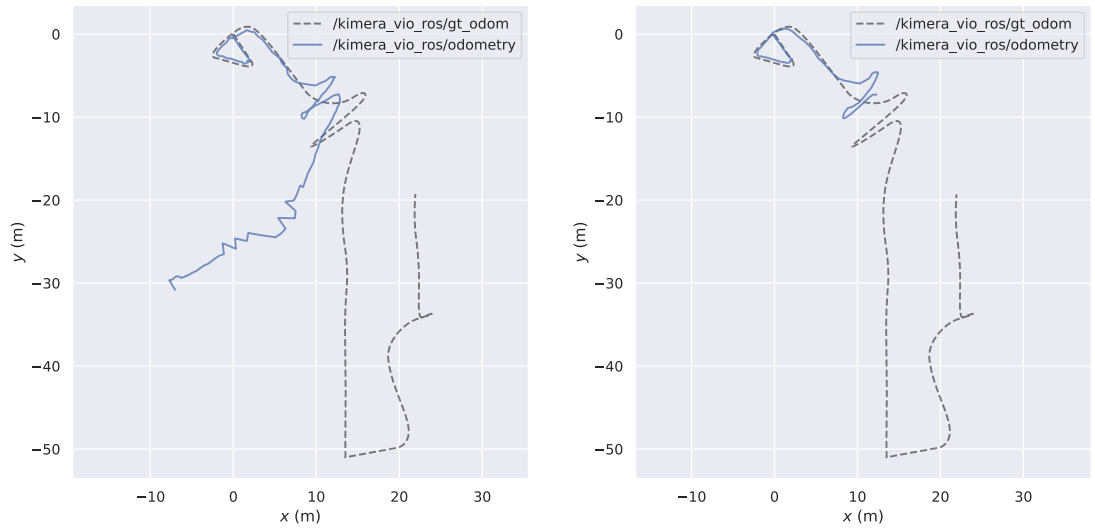

Figure 6.5: The trajectories of Kimera-VIO (left) and Semantic-Kimera-VIO (right) in the highly dynamic VIODE city-day sequence.

To account for randomness, both Kimera versions were executed on the city-day-3 dataset ten times; a boxplot of this is shown in Figure 6.6. This boxplot shows both the RMSE RPE and RMSE ATE of Semantic-Kimera-VIO (pink) and Kimera-VIO (blue). We observe that the RMSE RPE is lower for Semantic-Kimera-VIO than for Kimera-VIO, indicating that Semantic-Kimera-VIO is more accurate. The variance of Kimera-VIO is higher than for Semantic-Kimera-VIO, indicating that Kimera-VIO's performance is less consistent. The high accuracy and consistency of Semantic-Kimera-VIO may be because it completely lost track after a short period of time, as seen from Figure 6.5. This motivates a further investigation of SR to evaluate robustness.

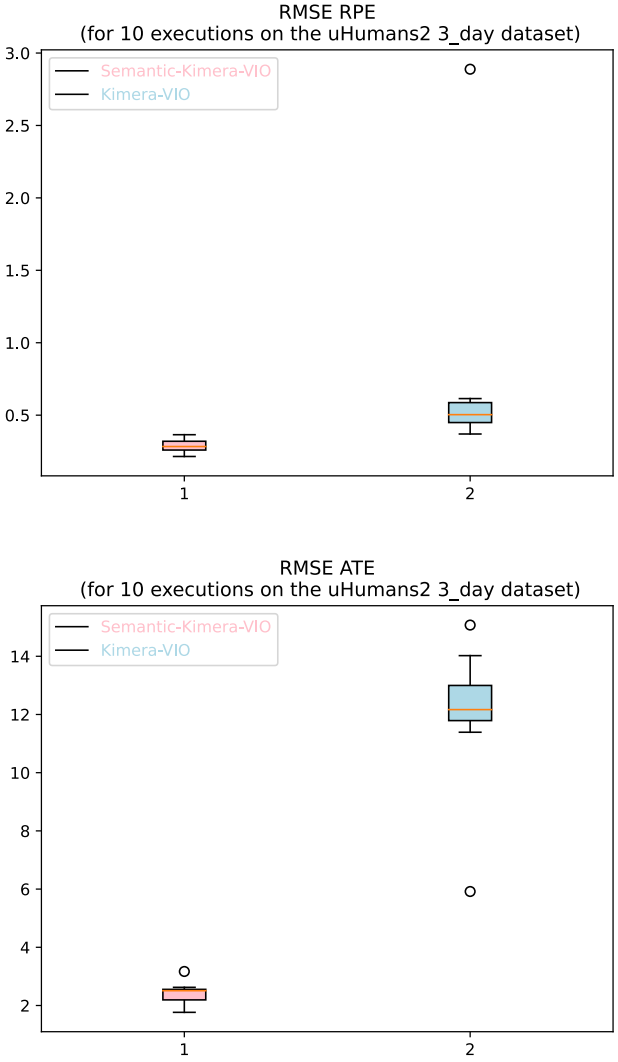

Figure 6.6: Boxplot of Semantic-Kimera-VIO (pink) and Kimera-VIO (blue) on the VIODE city-day-3 dataset over ten executions.

Table 6.3 shows the SR being ≈ 0.77 for Kimera-VIO and ≈ 0.4 for Semantic-Kimera-VIO on the highly dynamic city-day dataset. These values imply that Kimera-VIO outputs ego-motion for around 80% of the dataset duration, while Semantic-Kimera-VIO only output ego-motion for around 40%. The SR indicates relatively low robustness for both versions, but Semantic-Kimera-VIO is severely worse with a decreased SR of $\approx 48\%$ compared to Kimera-VIO.

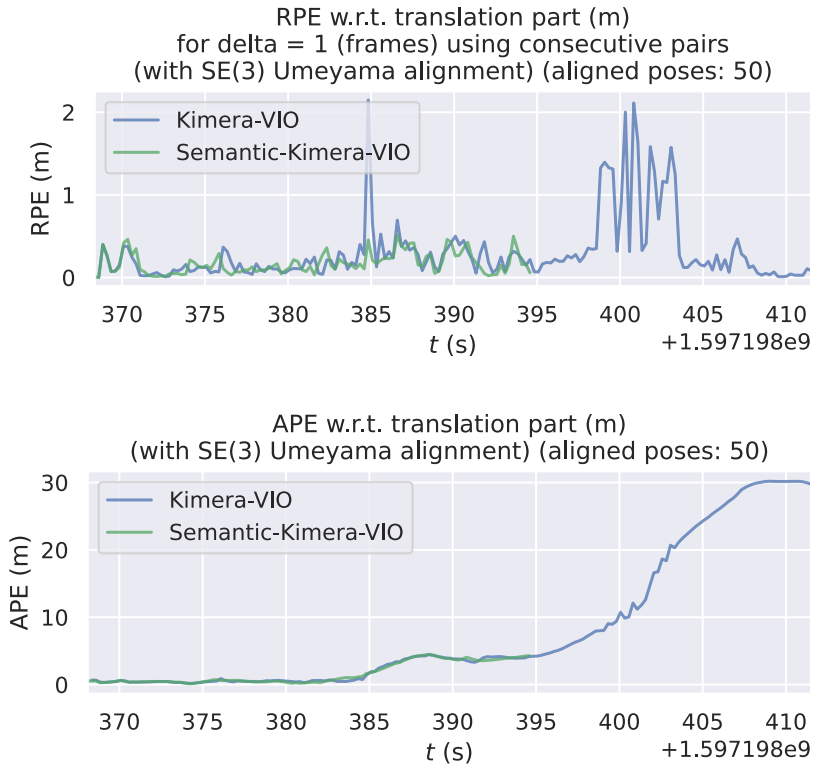

Figure 6.7: A plot of RPE (top) and ATE (bottom) over time of Kimera-VIO (blue) and Semantic-Kimera-VIO (green) on the on the VIODE city-day-3 dataset.

From the RPE plot in Figure 6.7 Kimera-VIO shows spikes in the RPE(t) at $t(s) \approx 385$ seconds and $t(s)$ between 400 and 405 seconds. The spikes during these periods indicate that the dataset is particularly challenging for Kimera-VIO. Semantic-Kimera-VIO does not spike at $t(s) \approx 385$ seconds; the reasoning may be similar to the argument given in section 6.2. Semantic-Kimera-VIO loses track at $t(s) \approx 395$ seconds, while Kimera-VIO continues tracking, indicating that Semantic-Kimera-VIO may be less robust at $t(s) \approx 395$ seconds.

6.3.2 Tracking Failure when the Entire Frame is Dynamic

We further investigate why only Semantic-Kimera-VIO loses track at $t(s) \approx 395$ seconds while Kimera-VIO continues tracking. Figure 6.8 shows a snapshot of the features tracked by Kimera-VIO (left) and Semantic-Kimera-VIO (right) at $t(s) \approx 395$ seconds. The snapshot shows tracked features in green, outliers in red, and new feature tracks in blue. This snapshot shows a dynamic truck covering the entire frame. We observe new feature tracks in blue on the truck for Kimera-VIO (left), while for Semantic-Kimera-VIO (right), no features are detected or tracked on the truck.

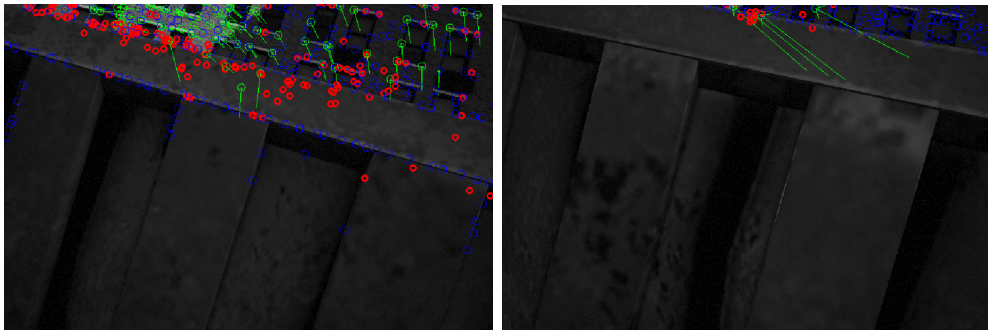

Figure 6.8: Comparison between the features tracked in Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the VIODE city-day-3 dataset at $t(s) \approx 395$ seconds.

When the snapshot in Figure 6.8 was captured, Semantic-Kimera-VIO stops tracking and reports having too few visual features to track. This problem occurs as Semantic-Kimera-VIO filters out all feature points detected on potentially dynamic objects by algorithm 2. This results in all feature points on the truck being filtered out. Kimera-VIO, on the other hand, continues to track dynamic features, although this leads to an erroneous ego-motion estimate as seen in Figure 6.5.

These results show that Semantic-Kimera-VIO loses track when most of the frame is covered by a dynamic vehicle. Kimera-VIO, on the other hand, continues to track, although the accuracy severely drops. Whether no ego-motion, inaccurate ego-motion or a re-initialization is desired depends on the application. However, it is important to be aware of the behavioral differences between Semantic-Kimera-VIO and Kimera-VIO when using them as a component of a larger system.

6.4 Near Equal Performance on the uHumans2 Dataset

In section 6.1 we argued that Semantic-Kimera-VIO and Kimera-VIO perform similarly on the dynamic uHumans2 datasets. These results were surprising as Semantic-Kimera-VIO improved on the highly dynamic VIODE datasets but not on the uHumans2 datasets. This section analyzes and compares the results from two single executions of Kimera-VIO and Semantic-Kimera-VIO on the dynamic uHumans2 datasets. The goal is to understand why Semantic-Kimera-VIO does not improve over Kimera-VIO on this dataset as it did on the VIODE dataset.

6.4.1 High Performance on the Dynamic uHumans2 Dataset

Figure 6.9 shows the ego-motion estimate of Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the most dynamic uHumans2 office dataset. Visually both Kimera versions look similar and track well compared to ground truth.

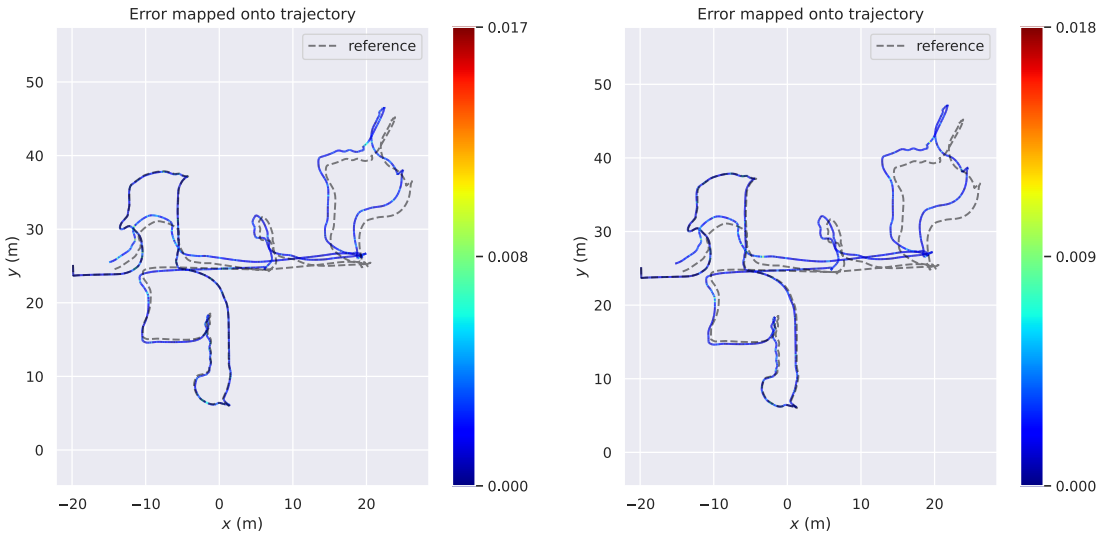

Figure 6.9: The trajectories of Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the uHumans2 office 12h dataset. The ground truth trajectory is shown as a dotted line in gray. The RPE is mapped onto the trajectories.

Figure 6.10 also show that Kimera-VIO and Semantic-Kimera-VIO have similar RPE values over time. These RPE values are small, below 0.02, and mostly below 0.01, which is over ten times smaller than the RPE values for the dynamic VIODE dataset shown in Figure 6.2 and Figure 6.7.

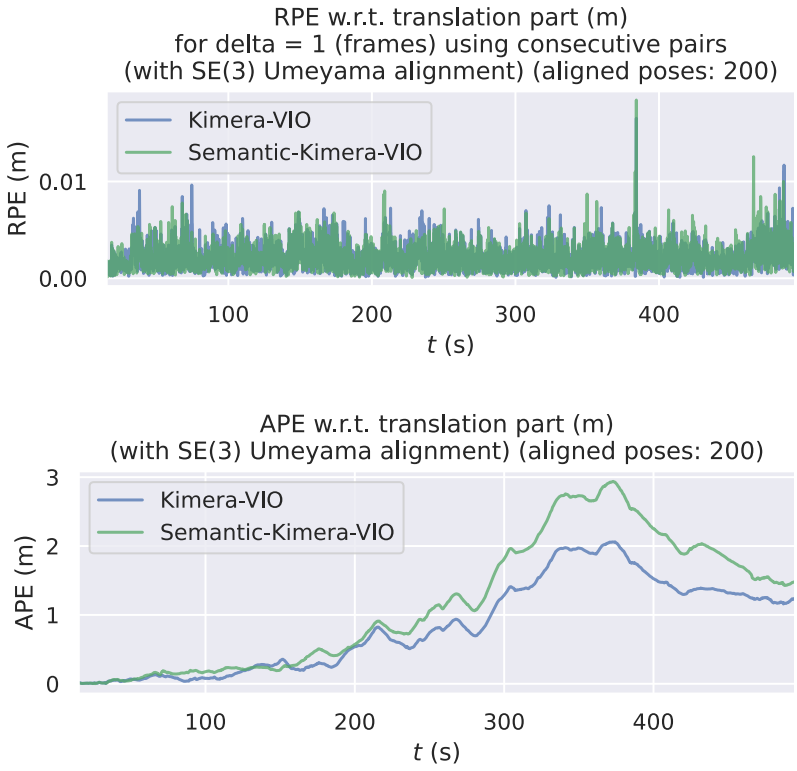

Figure 6.10: A plot of RPE (top) and ATE (bottom) over time of Kimera-VIO (blue) and Semantic-Kimera-VIO (green) on the uHumans2 office 12h dataset.

The results from the individual executions coincide with the quantitative RMSE RPE metrics in Table 6.1. The median RMSE RPE values over five executions for both Kimera versions on the uHumans2 datasets are $< 0.05[m]$. Both the results from the individual runs and the quantitative Mean RMSE RPE indicate that both Kimera versions perform well on the uHumans2 dataset.

6.4.2 The uHumans2 Datasets are only Slightly Dynamic

As Kimera-VIO performed poorly on the most dynamic VIODE dataset, the excellent performance on the most dynamic uHumans2 dataset seems surprising at first. A reason may be that the uHumans2 dataset does not contain any highly dynamic sequences.

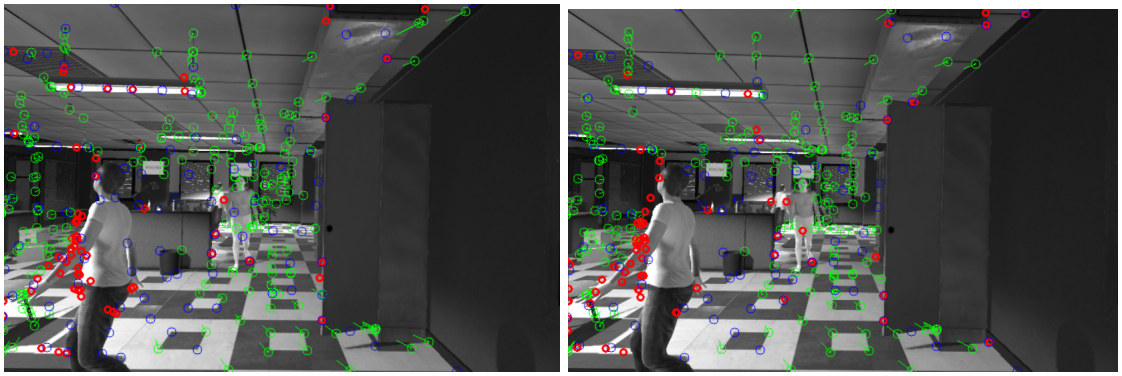

Figure 6.11: Comparison between the feature tracking in Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the uHumans2 office 12h dataset at $t(s) \approx 390$ seconds, where there is a high density of people. Red points are classified as outliers, while the green and blue points are inliers.

Figure 6.11 shows a screenshot of the uHumans2 office 12h, where the density of people is high relative to the rest of the dataset. Although people are present, they are neither as dynamic nor take up as much space as the dynamic vehicles in the snapshot of the highly dynamic VIODE dataset shown in Figure 6.3.

6.4.3 Geometric Verification is Sufficient in Slightly Dynamic Environments

Kimera-VIO shows high accuracy both in the static and dynamic uHumans2 datasets. Figure 6.11 shows a snapshot of the features detected for Kimera-VIO (left) and Semantic-Kimera-VIO (right) on the office 12h dataset. The feature points classified as outliers and inliers are similar in both versions. The features detected on the moving person in the left of the image are correctly classified as outliers by both versions.

These results show that the original Kimera-VIOs geometric verification correctly classifies inliers and outliers in this frame. The high performance of Kimera-VIO on this dataset is reasonable as the geometric verification is based on RANSAC, which is known to perform well when the minority of the data points are outliers, as was explained in section 3.2.2.

To conclude, Kimera-VIO performs well on even the most dynamic uHumans2 dataset. This may be because the dataset is only somewhat dynamic compared to the VIODE dataset, where Kimera-VIO struggled to track. Kimera-VIO's impressive tracking on the uHumans2 dataset indicates that RANSAC-based geometric verification is sufficient in somewhat dynamic environments.

7 Conclusion and Further Work

This chapter summarizes the contributions and the most significant results of this paper. Additionally, we mention further works which were either out of scope or not performed due to limited time.

7.1 Conclusion

This thesis presents Semantic-Kimera-VIO, a real-time VIO algorithm implemented in the Robot Operating System (ROS) and designed for dynamic environments. Our method uses semantic segmentation images to discard feature points located on people or vehicles. In contrast to most other SLAM systems designed for dynamic environments, our system is based on Kimera, which utilizes both stereo and inertial data. Kimera was chosen as it is a modern multi-sensor SLAM system suitable for industrial applications. By modifying Kimera-VIO, our system can also be used as a component of the larger Kimera SLAM system.

Semantic-Kimera-VIO and the original Kimera-VIO were tested in five different environments with varying amounts of dynamic people and vehicles. Their performance was evaluated based on visual inspection, Relative Pose Error (RPE), and the Success Rate (SR) metric introduced in this project. In static and slightly dynamic environments, Semantic-Kimera-VIO and Kimera-VIO performed similarly. However, in highly dynamic scenes, Semantic-Kimera-VIO acquired an RMSE RPE 45% lower than the original Kimera-VIO. Our results indicate that Semantic-Kimera-VIO is more accurate than Kimera-VIO in highly dynamic scenes. However, this accuracy came at the cost of an overall slightly lowered SR. This was especially prominent in certain settings, where the entire frame was covered by a vehicle.

7.2 Further Work

This section presents some limitations of this project and additions left for future work. These additions were either considered out of scope for this thesis, or work that could not be completed within the time limit of this project.

Advanced Semantic Outlier Removal

The current method of semantic outlier removal in Semantic-Kimera-VIO is simple. Firstly, only vehicles and people are used for semantic outlier removal, meaning other dynamic objects could deteriorate performance. Secondly, features from static people and vehicles are discarded, which could affect tracking in environments with a high density of stationary vehicles and people. How large numbers of static objects affect tracking is not currently tested, as the datasets used have a small density of static people and vehicles.

An advanced method for filtering out feature points on dynamic objects could have been implemented. The geometric verification of Kimera-VIO could be combined with

the current semantic outlier removal method in a more sophisticated manner to achieve this. Currently, our method loses track when the entire frame is covered by people or vehicles, which could be avoided by other methods.

Semantic-Kimera-VIO as a Component of SLAM

Semantic-Kimera-VIO is a modified Kimera-VIO, which is a component of a larger SLAM system. Currently, both versions have been tested alone as pure VIO systems and evaluated using RMSE RPE. Combining Semantic-Kimera-VIO with the Kimera loop closure module would make a complete SLAM system suitable for dynamic environments. It would be interesting to investigate whether place recognition could avoid our system from losing track when the entire frame is covered by vehicles. Another point of interest could be how our modifications affect the place recognition, loop closure, and general performance of such a system. Evaluating the accuracy of such system would be done using the RMSE ATE. Benchmarking Semantic-Kimera-VIO as part of a complete SLAM system was left to future work, as the focus of this thesis was constrained to the VIO frontend.

Direct Comparison with Previous Work

Although our Kimera version is evaluated using the standard metrics RMSE RPE and RMSE ATE, it is not directly compared to previous work. The main reason for this is that the datasets used in this project are relatively new and have not yet been tested with earlier systems. The established dynamic environment datasets, such as the TUM RGB-D were incompatible with Kimera as the dataset lacked IMU in the dynamic scenarios. As Semantic-Kimera-VIO is evaluated using standard metrics, future work can easily compare their results to our results. Directly comparing Semantic-Kimera-VIO with other VSLAM systems is therefore left to future work.

Limitations of Current Datasets

Although our system performs well on the simulated datasets, real-world data may provide additional challenges. Firstly, the simulated datasets do not contain enough static objects, such as parked vehicles. Secondly, they provide perfect semantic segmentation images. When running Semantic-Kimera-VIO on real-world data, a semantic segmentation network would need to be run, which would produce imperfect and time-delayed semantic segmentation images. Simulated datasets also provide near perfect sensor data which often fail to accurately simulate common sensor artifacts. These challenges are not yet handled.

Open-source real-world datasets with dynamic environments exist, but many do not contain IMU data, making them incompatible with Kimera. Therefore, creating a new visual-inertial dataset in dynamic scenes would significantly contribute to the field and could further motivate the need for more advanced methods for handling dynamic objects.

Bibliography

Lavish Arora, Sai Aditya Chundi, Mohan Krishna Nutalapati, Balasubramanyam Evani, Ketan Rajawat, and Rajesh M Hegde. Multisensor dataset repository for benchmarking slam in challenging scenarios. *ICRA 2019 Workshop*, 2019.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015. URL `http://arxiv.org/abs/1511.00561`.

Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, 2006. doi: 10.1109/IROS.2006.281644.

Dan Barnes, Will Maddern, Geoffrey Pascoe, and Ingmar Posner. Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1894–1900, 2018. doi: 10.1109/ICRA.2018.8460564.

Shahriar Negahdaripour Berthold K. P. Horn, Hugh M. Hilden. Closed-form solution of absolute orientation using orthonormal matrices. In *Journal of the Optical Society of America*, pages 1127–1134, 03 1988.

Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32 (6):1309–1332, 2016. doi: 10.1109/TRO.2016.2624754.

Carlos Campos, Richard Elvira, Juan J. Gomez, Jose M. M. Montiel, and Juan D. Tardos. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020.

Linyan Cui and Chaowei Ma. Sof-slam: A semantic visual slam for dynamic environments. *IEEE Access*, 7:166528–166539, 2019. doi: 10.1109/ACCESS.2019.2952161.

César Debeunne and D. Vivet. A review of visual-lidar fusion based simultaneous localization and mapping. *Sensors*, 20:2068, 04 2020. doi: 10.3390/s20072068.

Jakob Engel, Thomas Schoeps, and Daniel Cremers. Lsd-slam: large-scale direct monocular slam. In *Eur. Conf. Comput. Vis.*, volume 8690, pages 1–16, 09 2014. doi: 10.1007/978-3-319-10605-2_54.

Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *CoRR*, abs/1607.02565, 2016. URL `http://arxiv.org/abs/1607.02565`.

Epic Games. Unreal game engine, 2022. URL `https://www.unrealengine.com/en-US`.

Nolang Fanani, Alina Stürck, Marc Barnada, and Rudolf Mester. Multimodal scale estimation for monocular visual odometry. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1714–1721, 2017. doi: 10.1109/IVS.2017.7995955.

Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 05 2014. doi: 10.1109/ICRA.2014.6906584.

Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. *Robotics: Science and Systems (RSS) conference*, 01 2015.

Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286, 2013. doi: 10.1109/IROS.2013.6696514.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. *CoRR*, abs/1806.01260, 2018. URL `http://arxiv.org/abs/1806.01260`.

Anders Grunnet-Jepsen, Michael Harville, Brian Fulkerson, Daniel Piro, Shirit Brook, and Jim Radford. Introduction to intel® realsense™ visual slam and the t265 tracking camera, version 1.0. URL `https://dev.intelrealsense.com/docs/intel-realsensetm-visual-slam-and-the-t265-tracking-camera`.

Michael Grupp. evo: Python package for the evaluation of odometry and slam. `https://github.com/MichaelGrupp/evo`, 2017.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. URL `https://arxiv.org/abs/1703.06870`.

Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. Dynamic slam: The need for speed, 2020. URL `https://arxiv.org/abs/2002.08584`.

Berthold Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society A*, 4:629–642, 04 1987. doi: 10.1364/JOSAA.4.000629.

Irani, Rousso, and Peleg. Recovery of ego-motion using image stabilization. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, 1994. doi: 10.1109/CVPR.1994.323866.

Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 12 2007. ISBN 978-1-4244-1749-0. doi: 10.1109/ISMAR.2007.4538852.

Laurent Kneip, Margarita Chli, and Roland Siegwart. Robust real-time visual odometry with a single camera and an imu. 08 2011. doi: 10.5244/C.25.16.

Minoda Koji, Schilling Fabian, Wüest Valentin, Floreano Dario, and Takehisa Yairi. VIODE: A simulated dataset to address the challenges of visual-inertial odometry in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2):1343–1350, 2021. doi: 10.1109/LRA.2021.3058073.

Steven Lovegrove, Andrew J. Davison, and Javier Ibañez-Guzmán. Accurate visual odometry from a rear parking camera. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 788–793, 2011. doi: 10.1109/IVS.2011.5940546.

Anastasios I. Mourikis and Stergios I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572, 2007. doi: 10.1109/ROBOT.2007.364024.

Raul Mur-Artal and Juan D. Tardos. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 10 2017. ISSN 1941-0468. doi: 10.1109/tro.2017.2705103. URL `http://dx.doi.org/10.1109/TRO.2017.2705103`.

Raul Mur-Artal, J. Montiel, and Juan Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147 – 1163, 10 2015. doi: 10.1109/TRO.2015.2463671.

D. Nister. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756–770, 2004. doi: 10.1109/TPAMI.2004.17.

The University of Tokyo JSK laboratory. Jsk visualization, 2022. URL `https://github.com/jsk-ros-pkg/jsk_visualization`.

Vignesh Prasad, Saurabh Singh, Nahas Pareekutty, Balaraman Ravindran, and K. Madhava Krishna. Slam-safe planner: Preventing monocular SLAM failure using reinforcement learning. *CoRR*, abs/1607.07558, 2016. URL `http://arxiv.org/abs/1607.07558`.

Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669. IEEE, 2018.

Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, PP, 08 2017. doi: 10.1109/TRO.2018.2853729.

Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, volume 3, 01 2009.

Intel Realsense. Intel® realsense™ tracking camera t265, 2021. URL `https://www.intelrealsense.com/`.

Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. URL `http://arxiv.org/abs/1506.02640`.

A. Rosinol, A. Violette, N. Hughes M. Abate, Y. Chang, A. Gupta J. Shi, and L. Carlone. Kimera: from SLAM to spatial perception with 3D dynamic scene graphs. In *arxiv*, 2021.

Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. URL `https://github.com/MIT-SPARK/Kimera`.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. Pearson Education Inc., Upper Saddle River, New Jersey 07458, 2010.

Muhamad Risqi U. Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Comput. Surv.*, 51(2), feb 2018. ISSN 0360-0300. doi: 10.1145/3177853. URL `https://doi.org/10.1145/3177853`.

Xuesong Shi, Dongjiang Li, Pengpeng Zhao, Qinbin Tian, Yuxin Tian, Qiwei Long, Chunhao Zhu, Jingwei Song, Fei Qiao, Le Song, Yangquan Guo, Zhigang Wang, Yimin Zhang, Baoxing Qin, Wei Yang, Fangshi Wang, Rosa H. M. Chan, and Qi She. Are we ready for service robots? the OpenLORIS-Scene datasets for lifelong SLAM. In *2020 International Conference on Robotics and Automation (ICRA)*, pages 3139–3145, 2020.

Robert Sim, Pantelis Elinas, and Matt Griffin. Vision-based slam using rao-blackwellised particle filter. *IJCAI 2005 Workshop on Reasoning with Uncertainty in Robotics, RUR 2005*, 01 2005.

J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012a.

Jrgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 10 2012b. ISBN 978-1-4673-1737-5. doi: 10.1109/IROS.2012.6385773.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Martin Thoma. A survey of semantic segmentation. *CoRR*, abs/1602.06541, 2016. URL `http://arxiv.org/abs/1602.06541`.

Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 13, pages 376–380, 04 1991.

Unity Technologies. Unity game engine, 2021. URL `https://unity.com/`.

Wan Mohd Yaakob Wan Bejuri, Mohd Mohamad, and Raja Zahilah Raja Mohd Radzi. Emergency rescue localization (erl) using gps, wireless lan and camera. *International Journal of Software Engineering and Its Applications*, 9:217–232, 09 2015. doi: 10.14257/ijseia.2015.9.9.19.

R. Wang, M. Schwörer, and D. Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916, 2020. doi: 10.1109/IROS45743.2020.9341801.

Gengshan Yang and Deva Ramanan. Learning to segment rigid motions from two frames. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1266–1275, June 2021.

Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry, 2018.

Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174, 2018. doi: 10.1109/IROS.2018.8593691.

Huangying Zhan, Chamara Saroj Weerasekera, Jia-Wang Bian, Ravi Garg, and Ian D. Reid. DF-VO: what should be learnt for visual odometry? *CoRR*, abs/2103.00933, 2021. URL `https://arxiv.org/abs/2103.00933`.

Wu Zhou, Shiju E., Zhenxin Cao, and Ying Dong. Review of slam data association study. 01 2016. doi: 10.2991/icsnce-16.2016.4.