Performance Enhancements to Visual-Inertial SLAM for Robots and Autonomous Vehicles

By

Marcus Abate

B.S, Aerospace Engineering, Massachusetts Institute of Technology, 2020

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Authored by: Marcus Abate
Department of Aeronautics and Astronautics
May 23, 2023

Certified by: Luca Carlone
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: Jonathan P. How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Performance Enhancements to Visual-Inertial SLAM for Robots and Autonomous Vehicles

by

Marcus Abate

Submitted to the Department of Aeronautics and Astronautics
on May 23, 2023, in Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

Abstract

Spatial perception is a key enabler for effective and safe operation of robots and autonomous vehicles in unstructured environments. Two key components of a complete spatial perception system are: identifying where the robot is in space, and constructing a representation of the world around the robot. In this thesis, we study Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM) and present several findings on its application to a variety of robotic platforms to obtain globally-consistent localization for a robot as well as a dense map of its surroundings. In particular, we extend Kimera, an open-source VI-SLAM pipeline, to be more effective in traditional use-cases (*e.g.,* stereo-inertial VI-SLAM) as well as more broadly applicable to different platforms and sensor modalities.

Our first contribution is to present a system built around Kimera for autonomous valet parking of self-driving cars, and test on real-world self-driving car datasets. This system uses a modified version of Kimera to support multi-camera VI-SLAM and perform dense free-space mapping using multiple cameras with non-overlapping field of view. Our second contribution is to describe recent updates to Kimera and showcase their beneficial effect on localization and mapping performance, while also comparing against the state of the art on extensive datasets collected on a variety of platforms. Finally, we present a novel method for detecting and tracking humans in the scene in order to build 3D Dynamic Scene Graphs for high-level perception tasks, and evaluate our method in a photorealistic simulation environment. We conclude by commenting on the advantages of Kimera and identifying areas for future work.

Thesis Supervisor: Luca Carlone
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank my advisor, Professor Luca Carlone, for his guidance and mentorship throughout my time as both an undergraduate and a graduate student. I would also like to thank my labmates for their assistance and for all the great conversations, and I am grateful in particular to Yun, Nathan, and Jingnan. Finally, I would like to thank my parents, my two brothers Alex and Nicky, and Olga, for their support throughout my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Spatial perception, including localization and mapping, is a core capability of robots and autonomous systems operating in unstructured environments. From drones, to self-driving cars, to planetary rovers, robots must be capable of (i) localizing in their environment, (ii) building geometric maps of their surroundings, (iii) understanding the semantic *meaning* of their surroundings, and (iv) tracking things (*e.g.,* moving objects and humans) in their environment to be safe and effective. If we are to ever see autonomous agents deployed among us in a serious manner, these four requirements must be met regardless of the type of platform. Localization is of primary importance; the robot must know where it is in the world to begin any meaningful action. Mapping is key to safe planning and interaction. Semantic understanding is useful for executing high-level tasks; moreover relying solely on geometric understandings of the world denies robotic systems many of the useful heuristics we humans use to accomplish tasks. And finally, tracking dynamic objects (and in particular, humans) is vital to safe operation and task execution. However, performing these tasks is difficult, especially in real time and with inexpensive sensors. And for resource-constrained robots (*e.g.,* quadcopters), the difficulty is compounded.

The autonomy community has largely converged on a few dominant paradigms for localization and mapping: visual (computer-vision), range-based (LiDAR/Radar), and inertial (IMU/GPS). Inertial methods, while simple, are insufficient for most real-world requirements other than pure localization. Range-based methods are effective,

but accurate and long-distance range sensing (*e.g.,* with LiDAR) is still relatively expensive. Methods that rely on cameras are attractive because they provide a great deal of information (akin to what humans work with), and are relatively inexpensive. Because of their cost-effectiveness, they can be deployed in a wide array of contexts, and in a rapid manner. This has led to vision becoming the focal point for academic research into perception systems. Visual-Inertial Simultaneous Localization and Mapping (VI-SLAM) represents a state-of-the-art framework for localization and mapping, and is reguarly deployed on many robotic platforms.

In this thesis, we consider the following problem: how do we perform localization, geometric/semantic mapping, and dynamic tracking, all in a purpose-built system that is robust enough for real world deployment and flexible enough to be deployed on a variety of robotic platforms? To that end, we present recent work done on expanding Kimera [60, 62] - an open-source VI-SLAM pipeline we developed as part of a prior publication - to be more effective in localization and mapping. In particular, after reviewing related work in chapter 2, we present efforts that further extend VI-SLAM to be more robust, more flexible, and applicable in broader contexts in chapters 3 and 4. Finally, we discuss prior work on extending VI-SLAM into higher-level spatial AI through the use of 3D Dynamic Scene Graphs, including how to track humans moving around the robot.

1.1 Thesis Structure and Summary of Contributions

This thesis is structured as follows:

- Chapter 2 discusses relevant literature as a primer to the thesis.
- Chapter 3 describes extensions to Kimera [60, 62] to support autonomous valet-parking for self-driving cars, in collaboration with The Ford Motor Company. We discuss modifications to Kimera-VIO [1] to support monocular cameras, multiple non-overlapping cameras, and other sensor inputs to improve the state estimate and support dense environmental mapping. We also discuss modifications to Kimera-Semantics to support efficient ground-plane mapping with

multiple cameras with non-overlapping field of view, and evaluate our method in simulation and on real-world datasets collected on a self-driving testbed at Ford. This work has been submitted to the Robotics and Automation Letters (RA-L) in 2023 [1].

- Chapter 4 showcases recent updates to the open-source version of Kimera that improve performance for a wide range of applications. These updates include changes to the Kimera-VIO frontend and backend to support more sensor schemes as input, enable more robust feature tracking, and improve outlier rejection for pose-graph optimization. We perform ablation studies on various new features to evaluate performance increase, and present an experimental comparison of Kimera against the state of the art on many datasets collected from heterogeneous robotic platforms. In this chapter we show the robustness and flexibility of Kimera as compared to other open-source pipelines.
- Chapter 5 describes previous work on Kimera-Semantics [60, 62] to support dynamic human tracking during the construction of hierarchical map representations, namely 3D scene graphs. In particular, we highlight how a unified approach for tracking the robot's pose and for tracking dynamic objects in the scene (*e.g.,* humans) is efficient and effective using the Kimera backend engine. This work was published in the International Journal of Robotics Research in 2021 [62].
- Chapter 6 concludes the thesis with closing thoughts and avenues for future work.

Chapter 2

Related Works

2.1 VI-SLAM Systems

Previous-generation open-source VI-SLAM algorithms, such as Vins-Mono [58] and ORB-SLAM2 [54], used monocular camera and IMU input. Their modern counterparts support stereo cameras for accurate depth estimation, including Vins-Fusion [57], ORB-SLAM3 [14], Open-VINS [28], and Kimera [62]. Additional performance can be gained from RGB-D (depth) cameras [70]. While some of these algorithms can perform vision-only SLAM, results are best with the inclusion of an IMU sensor. At a high-level, the structure of these systems is fairly similar, with a front-end module that performs feature detection and tracking (or matching), and a back-end that estimates the trajectory and a sparse landmark-based map via factor graph optimization or an Extended Kalman Filter; the front-end is typically based on OpenCV [10] for image processing, while the backend is commonly based on optimization libraries, *e.g.,* Kimera uses GTSAM [23], while Vins-Fusion uses Ceres [2]. For graph methods, the optimization is done at keyframe rate, usually much lower than camera and IMU rate but fast enough to provide accurate odometry in real time. Generally speaking, these backend architectures are flexible enough to accept different factors modeling different types of sensor data. This has led to novel approaches for sensor fusion that combine visual-inertial or inertial factors with other sensors like LiDAR, such as Shi *et al.* [69] and Chang *et al.* [16]. These systems for multi-sensor fusion have been demonstrated

in different applications, including drone and ground robot navigation [13, 20]. However, doing so requires sacificing the low cost of camera-based systems and frequently becomes too expensive for commercial usage. The additional computation required to process 3D LiDAR can also slow down state estimation and require more powerful hardware. On the other hand, simple monocuar or stereo-inertial systems miss a lot of important information for mapping and have less accurate state estimation due to the sparsity of tracked features.

2.2 Multi-Camera VI-SLAM

One option for increasing the estimation accuracy and robustness of VI-SLAM methods is to leverage multiple cameras mounted on the robot. Frequently in the literature, the use of multiple cameras arranged around the robot is referred to as "surround-view". This is particularly helpful for larger platforms like cars, where a single camera cannot capture all of the relevant data for mapping, obstacle avoidance, navigation, etc. Eckenhoff *et al.* [21] implement an EKF-based visual-inertial odometry (VIO) pipeline that supports multiple cameras and IMUs. Cameras are processed independently and asynchronously, so feature tracks are not shared between sensors. In addition, the method does not include a loop-closing solution or global mapping solution. Yang *et al.* [82] use several pinhole cameras with fixed intrinsics and extrinsics, but unlike Eckenhoff *et al.* [21], they group frames from multiple cameras together temporally and permit feature-track sharing between cameras with overlapping fields-of-view. The method uses Cubic B-splines for pose estimation and also produces a local map, and uses a bag-of-words approach to perform loop closure detection [27]. Zhang *et al.* [88] improve on these concepts by using a single set of feature tracks among all cameras without requiring overlapping fields-of-view, along with a factor-graph-based backend. The authors show that this increases accuracy while reducing the size of the factor graph and resultant optimization complexity, however they use synchronized cameras and in particular stereo cameras in addition to monocular cameras. Jaekel [33] do cross-camera feature tracking with several stereo pairs.

He *et al.* [30] also implement cross-camera feature sharing without constraints on the camera field of view to produce notable results, but at the cost of requiring GPU acceleration for the frontend and bundle-adjustment steps, thereby limiting the platforms on which the method can be deployed. Wang *et al.* [77] implement an efficient scheme for surround-view localization using 4 cameras on a ground robot by assuming planar motion only. The distinguishing features of our system described in Chapter 3 are its capability to fuse multiple asynchronous camera feeds, support robust monocular loop closure optimization, and perform free-space mapping using monocular cameras, all without relying on GPU acceleration.

2.3 Autonomous Valet Parking

Several recent works have tackled the specific problem of autonomous parking in the context of self-driving vehicles. This is distinct from the general autonomous driving problem, which is often studied in the context of either indoor environments with small robots and few obstacles, or on car platforms driving on roads and highways. The parking problem involves low speeds in obstacle-rich environments as frequently there are pedestrians and other cars involved. Additionally, indoor parking in particular can be challenging due to the similarity of scene in many parts of the environment, making loop closure and mapping difficult. Tripathi and Yogamani [74] summarize the challenges of using visual-inertial odometry to build global maps for relocalization, which is a necessary component of real-time autonomous parking. Shao *et al.* [66, 65] introduce a VIO system that leverages surround-view cameras, but for the purpose of detecting parking spots instead of in the SLAM loop. Yu *et al.* [85] and Xiang *et al.* [80] use surround-view images in a similar way, but take the extra step of performing feature tracking on these bird's-eye view images and incorporating those features in a hierarchical factor-graph optimization. Performing SLAM on surround-view images in this way comes at the cost of assuming strictly planar movement on a 2D map, which is sufficient for ground vehicles in most cases, but limits performance in multi-story parking lots and limits extensibility to future applications that

require 3D navigation (*e.g.,* off-road navigation). These limitations are partially addressed by Khoche *et al.* [39], where the authors also propose a 3D mapping method that remains efficient but requires LiDAR. Our proposed method in Chapter 3 does not make planar assumptions but rather tracks movement in 3D, yet makes use of multiple cameras for improving localization and mapping. As we do not rely on a birds-eye view, our localization method is more generalizable to broader autonomy tasks, working in the direction of a unified autonomy solution as opposed to a highly specialized parking solution.

2.4 Free-Space Mapping for Autonomous Parking

A major component of the autonomous parking problem is the path planning phase, which is made possibly by a dense and semantically annotated map of the environment. These maps can be generated by the SLAM algorithm, however at a bare minimum they must be annotated to show free-space in the environment. In the case of self-driving cars, this is empty road and empty parking spots. Shao *et al.* [66, 65] and their recent extension [67] use surround-view cameras to generate a planar ground-plane map around the vehicle, and used this map in conjunction with a CNN to detect parking spaces for high-level planning. Tripathi *et al.* [74] and Shao *et al.* [67] both use semantic segmentation networks to identify common labels like pedestrians, speed bumps, etc. Wu *et al.* [79] use an object detector in the loop with VIO to identify dynamic obstacles and remove features on those obstacles from the visual SLAM frontend. Building these semantic maps requires a segmentation network to classify free-space (ground-plane), but it also requires a way to get a dense depth map of image pixels that are on the ground. A popular solution has been to use mono-depth estimation networks like in Wimbauer *et al.* [78]. An alternative solution in the vein of sensor fusion is to use LiDAR data to gather depth information and align it with the semantically segmented image, as in Khoche *et al.* [39]. Our proposed mapping solution in Chapter 3 does not require expensive LiDAR sensors or complex depth estimation networks to perform ground-plane mapping. In addition,

our implementation supports arbitrary label spaces for semantic segmentation [62], opening up possibilities for including other labels in the map (*e.g.,* parking spots) when available.

2.5 Dynamic Scene Graphs

Scene graphs are popular computer graphics models to describe, manipulate, and render complex scenes and are commonly used in game engines [76]. Scene graphs have been mostly used used in 2D contexts, typically in computer vision applications to abstract the content. Krishna *et al.* [46] use a scene graph to model attributes and relations among objects in 2D, using natural language captions defined manually. Xu *et al.* [81] and Li *et al.* [48] develop algorithms for 2D scene graph generation. 2D scene graphs have been used for image retrieval [37], captioning [45, 3, 36], high-level understanding [18, 89, 31, 35], visual question-answering [26, 92], and action detection [51, 49, 87]. Armeni *et al.* [7] propose a 3D scene graph model to describe 3D static scenes, and describe an algorithmic way to build 3D scene graphs. In parallel to Armeni *et al.* [7], Kim *et al.* [40] propose a 3D scene graph model for robotics, which however only includes objects as nodes and misses multiple levels of abstraction afforded by Armeni *et al.* [7] and by our system as proposed in Chapter 5.

2.6 Human Motion Tracking

Human pose and shape estimation from a single image is a growing research area. Kolotouros *et al.* [44] and others [42, 43, 41] provide a substantive review of the literature, though we will mention that related work includes optimization-based approaches, which fit a 3D mesh to 2D keypoints [9, 47, 86, 42, 84], and learning-based methods, which infer the mesh from the pixels directly [71, 38, 55, 56, 44, 42]. Human models are typically parametrized using the *Skinned Multi-Person Linear Model* (SMPL) [50], which provides a compact pose and shape description and can be rendered as a mesh with 6890 vertices and 23 joints. The common approach to monoc-

ular human tracking is to predict joint probabilities in the 2D image space, which are optimized to 3D joints based on multiple time-series observations and motion priors [5, 4, 8, 11, 22, 91, 75]. Taylor *et al.* [72] combine a learned motion model with particle filtering to predict 3D human poses. In Chapter 5, we aim to not only estimate the 3D pose of the human, but also the full SMPL shape without maintaining the persistent image history required by many of the approaches above. Some efforts, like the work done by Arnab *et al.* [8], fully reconstruct the SMPL shape of the human; however, in this case they reconstruct the shape after performing data association over multiple timesteps. In contrast, we use the method of Kolotouros *et al.* [44] to directly get the full 3D pose of the human at each timestep, simplifying pose estimation, and allowing us to do data association based on the SMPL body shape.

Chapter 3

Multi-Camera VI-SLAM for Autonomous Valet Parking

3.1 Introduction

Visual-inertial (VI) SLAM algorithms have seen widespread use in a variety of robotics platforms, from drones to rockets and ground robots [25, 53, 13]. Typically, these algorithms employ a monocular or stereo camera and an inertial measurement unit (IMU). Using a single camera works well for aerial vehicles and small robots, where payload constraints limit the number of onboard sensors. However, in other applications, such as self-driving cars, one would prefer to use multiple cameras around the vehicle to improve accuracy and robustness of visual-inertial SLAM and enable a broader coverage for 3D mapping of the vehicle's surroundings.

In this chapter, we study multi-camera VI-SLAM for autonomous valet parking. Vision-based autonomous parking is generally less well-studied than highway or city driving in the research literature, and presents unique challenges. Parking can happen in outdoor environments with many dynamic obstacles such as pedestrians or other cars, which necessitates very accurate free-space mapping for safe navigation. Parking

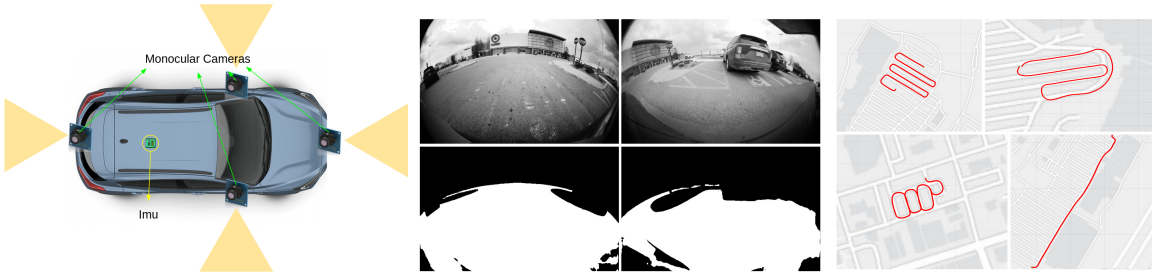
The work in this chapter was partially funded by the Ford Motor Company.

can also happen in indoor parking garages, which are generally GPS denied environments with visually similar scenes throughout (*e.g.,* think about the different floors of an indoor parking garage), making place recognition and drift correction difficult. Additionally, parking scenarios see the car traveling at low speeds for long stretches, often creating degenerate conditions for visual-inertial odometry estimation. A comprehensive autonomous parking solution must be accurate and robust both in state estimation and mapping, and must function in GPS-denied environments, while simultaneously being capable of accepting other sensor inputs when available. There exist several efforts in the literature for performing multi-camera SLAM with IMUs. Of the methods that have been evaluated on datasets targeted at the autonomous parking problem, most of the approaches build 2D representations of the environment, which limits their applicability to multi-story parking garages [66, 65, 85, 80]. Methods that do build 3D maps either require sensor fusion with more expensive sensors (*e.g.,* LiDAR) [39, 69] or are not close to real-time operation.

In this chapter, we develop a multi-camera VI-SLAM pipeline that can perform efficient and globally consistent trajectory estimation and builds a dense 3D map of the free space around the vehicle, which enables obstacle avoidance and navigation. The proposed system builds on Kimera [62, 60] and extends it to (i) accept multi-camera and external odometry sources, (ii) enable robust monocular or multi-camera loop closures, and (iii) perform efficient ground-plane mapping for autonomous valet parking applications. Kimera's frontend and backend are modified to improve tracking and factor-graph optimization, and support multi-sensor fusion, using a heavily parallelized architecture. Several loop-closure methods are implemented including monocular loop closure techniques that are shown to outperform popular approaches based on the Perspective-n-Point (PnP) method. Finally, our SLAM system uses Kimera-Semantics [62, 60] in conjunction with a fast semantic segmentation network to create a 3D map of the free space around the robot. The method is validated in photo-realistic simulations and on several real datasets collected using a car prototype developed by the Ford Motor Company, spanning both indoor and outdoor parking scenarios. Our multi-camera system is shown to outperform state-of-the art

open-source VI-SLAM pipelines (Vins-Fusion, ORB-SLAM3), and exhibits an average trajectory error under 1% of the trajectory length across more than 8 km of distance traveled (combined across all datasets).

This chapter is organized as follows: Section 3.2 describes the proposed method, Section 3.3 details experimental results, and Section 3.4 presents closing thoughts.



(a) (b) (c)

Figure 3-1: (a) Illustration of the Ford test bed and sensor setup. (b) Four sample images from an outdoor dataset; the top two are from the front and right cameras onboard the car and the bottom two are the output of the semantic segmentation network that identifies free-space road for the mapping module. (c) Sample of outdoor trajectories collected on the car in Detroit, Michigan, USA. The pictured trajectories are on average 450 m in length.

3.2 System Architecture

This section describes the hardware (Section 3.2.1) and software architecture (Section 3.2.2) of the proposed system.

3.2.1 Hardware Architecture and Data Collection

Hardware Architecture. The real-world platform used for experiments was a modified Lincoln MKZ sedan with custom engine control units and custom onboard sensors, including four monocular fisheye cameras, an IMU, and an onboard wheel-odometry system that uses wheel-encoders as well as other proprietary sensors to estimate the car's motion. The sensors are all production-equivalent except for the IMU. Figure 3-1 shows the arrangement of the sensor suite on the car. The IMU data was provided by an RT3000 unit, and the cameras were 1-MegaPixel production

automotive cameras. IMU data is collected at 100Hz while camera data at 20Hz. Wheel odometry data is also provided at IMU rate. Ground-truth data (only used for benchmarking) is provided by differential GPS for the outdoor datasets at IMU rate. All sensors use the same onboard clock but are not synchronized and even data from sensors running at the same rate (*e.g.,* IMU and wheel odometry) may arrive at different instants. Data was collected via a ROS node developed to interface with the raw sensor data, which were communicated via CAN bus and UDP onboard the car. While the car was capable of running Kimera online, the results presented in this chapter are obtained using a desktop computer running Ubuntu 20.04 with a 24-core Intel processor.

Data Collection. For evaluation, we used both simulated datasets and real datasets collected with the vehicle described above. The simulated datasets were recorded in the TESSE simulation environment [62]. The scene was an outdoor urban area, and the data came from a simulated car with a realistic dynamics model. Ground-truth pixel-wise semantic labels were extracted from the simulator in place of a segmentation network, and the labels associated with roads were used for the free-space reconstruction in Kimera-Semantics.

For the real-world datasets, we collected 22 datasets recorded in 5 different locations, both indoor and outdoor, over the course of 5 months. The speeds of the car in the datasets varied, as did trajectories, environment, obstacles, and weather (see sample trajectories in Fig. 3-1c). Some of the datasets had occlusions on one or multiple cameras for extended periods, others had long stops for traffic, crowded markets with many pedestrians, and long straight-line motion beyond the VIO module's time horizon that made scale estimation difficult. For the indoor datasets, GPS was unreliable inside of the parking garages, so we used Ford's proprietary wheel-odometry motion estimate as ground truth as it proved extremely accurate, especially at low and medium speeds.

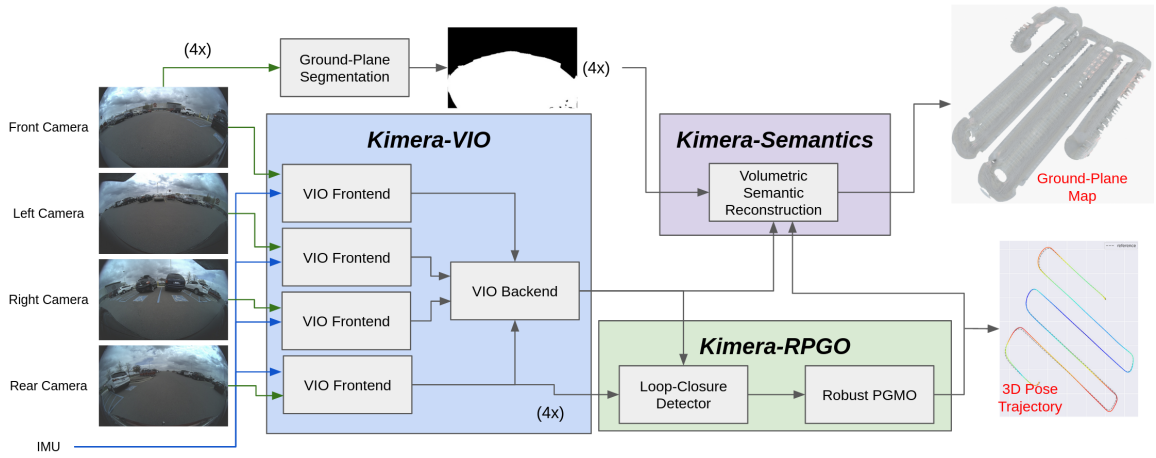


Figure 3-2: Overview of the proposed system architecture. Inputs are RGB monocular images from all four sides of the car, as well as a single IMU. Our modified Kimera-VIO processes all camera inputs in parallel and generates a robust state estimate, which is fed to the Robust Pose Graph Optimization (RPGO) module for loop closure detection and correction. Simultaneously, a semantic segmentation network identifies the ground plane in the image, which is used by the modified Kimera-Semantics module to generate a 3D reconstruction of the free space. For a more in-depth description of Kimera's modules, refer to [62].

3.2.2 Software Architecture

The proposed system includes several major modifications to the existing Kimera-VIO and Kimera-Semantics pipelines described in [62, 60]. A diagram showing the general architecture of the proposed system is given in Fig. 3-2.

Multi-Camera Kimera-VIO. Previously, Kimera was limited to perform tightly-coupled visual-inertial odometry using a stereo camera. The proposed version of Kimera is modified to accept monocular image data —potentially from multiple cameras— coupled with IMU data. The Monocular-VIO frontend is split into two; IMU data is preintegrated with interpolation on both endpoints according to standard methods [24]. The image data is then processed using tools from OpenCV [10] to detect features (`goodFeaturesToTrack`) and track them across frames. 5-point RANSAC is then applied on the tracked features to remove outliers at each keyframe; keyframes are triggered depending on the number and quality of the tracked image features. In addition to visual and inertial data, we use also wheel odometry measurements. Towards this goal, we chain relative poses provided from the onboard

wheel odometry sensors (at 100Hz) to estimate relative poses between keyframes and then use those as relative pose factors in the factor-graph-based VIO backend. The features, preintegrated IMU measurements, and odometry measurements are then sent to the VIO backend module, which performs fixed-lag smoothing using all the available measurements.

In order to take full advantage of the surround-view camera setup onboard the self-driving car platform, Kimera was modified to accept any number of monocular or stereo camera inputs. In the multi-camera configuration, each camera is processed using its own frontend module, and all frontends are run in parallel. As there may be slight delays in how the data from each camera is served to the pipeline, jointly optimizing the features from all cameras in one `RANSAC` problem would cause a slowdown while the pipeline waits for slower cameras. In our implementation, each of the frontend modules processes new feature detections from incoming frames and keeps track of its own feature tracks. The frontend modules then send their outputs to a single backend, which in turn keeps track of which camera provides which factors and performs a single factor-graph optimization over a receding horizon of 10 s, using GTSAM [23]. Since the cameras have a wide field of view and have a large distortion (beyond what can be captured by standard distortion models in OpenCV), the factors used in the factor-graph optimization for visual features were modified to increase robustness. In the standard version of Kimera, GTSAM's `SmartStereoProjectionFactor` is used for each of the visual landmarks [23]. For the proposed method, the triangulation occurring in the `SmartStereoProjectionFactor` was modified to optionally use the Huber norm to increase robustness to outliers. As part of this effort, we have also implemented several novel factors in GTSAM, including factors for automatic extrinsic camera calibration, rolling shutter correction, and projection factors that use the spherical camera model proposed by Scaramuzza *et al.* [63]. We have omitted them from this thesis since we haven't seen them produce more accurate results in our tests.

Loop Closure Optimization. While the VIO backend produces a locally consistent trajectory, our goal is to obtain a *globally* consistent estimate of the trajectory

and the map. The proposed architecture passes the VIO motion estimates to a robust pose graph optimization module that detects loop closures and optimizes the trajectory accordingly. For loop-closure detection, Kimera uses visual Bag-of-Words to detect similar images. This is done within the scheme described in [62] and [27]. Once a pair of putative matching images is identified, Kimera must generate a relative pose between the two frames before the factor can be included in the pose-graph optimization. Below we describe three approaches to compute the loop closure pose, namely PnP, a scale-less approach, and a rotation-only approach. After the relative pose between frames is computed, it is passed to Kimera-RPGO, for pose graph optimization. In Kimera-RPGO, we use graduated non-convexity [83] to robustify the estimate to spurious loop closures.

PnP Loop Closure Pose Computation. In the PnP approach, ORB descriptors are extracted at each keyframe and associated to each tracked feature point. At the same time, 3D landmark data from the VIO backend is sent to the loop-closure-detection module alongside each image. We use the ORB descriptors to obtain putative correspondences between the optimized landmarks from the backend and the 2D features, and use the PnP algorithm with RANSAC to find inlier correspondences. The RANSAC inliers are then used to generate the relative pose between the frames for the loop closure factor in Kimera-RPGO [62]. While this approach is fairly popular and used in other pipeline (*e.g.,* [14]), our experiments show that the resulting poses are not very accurate. This is due to the fact that there are typically few matches between 2D and 3D features (mostly due to the sparsity of the 3D landmark-based map), and hence few inliers after RANSAC.

Scale-less Loop Closure Pose Computation. In order to alleviate the problems with the PnP approach, we consider a version of the loop-closure-detection module that uses only the 2D image data to compute a relative loop closure pose up to scale. The pose up to scale is obtained using a standard 5-point RANSAC method, which can now directly rely on the many 2D-2D correspondences established between image features using ORB descriptor matching. As we do not have the scale factor on the translation in this case, we modify the information matrix of the noise model associated with the

loop closure factor to carry zero information along the direction of the translation vector. The pose-graph optimization then uses only the rotation part of the relative pose in the loop closure factor and the translation's direction but not magnitude, both of which are very accurate. More formally, the scale-less relative pose factors take a form similar to standard odometry and loop closure factors:

$$f(\boldsymbol{T}_i, \boldsymbol{T}_j) = \left\| \mathrm{Log}(\bar{\boldsymbol{T}}_{ij}^{-1} \boldsymbol{T}_i^{-1} \boldsymbol{T}_j) \right\|_{\boldsymbol{\Omega}_{ij}}^2 \tag{3.1}$$

where the loop closure measurement $\bar{\boldsymbol{T}}_{ij} \in \mathrm{SE}(3)$ relates two poses $\boldsymbol{T}_i, \boldsymbol{T}_j \in \mathrm{SE}(3)$ along the trajectory of the car, and $\mathrm{Log}(\cdot)$ is the standard logarithm map. The key difference in our case is that we set the information matrix $\boldsymbol{\Omega}_{ij}$ to be:

$$\boldsymbol{\Omega}_{ij} = \begin{bmatrix} \boldsymbol{\Omega}_R & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_3 - \bar{\boldsymbol{t}}_{ij} \bar{\boldsymbol{t}}_{ij}^{\mathsf{T}} \end{bmatrix} \tag{3.2}$$

where $\boldsymbol{\Omega}_R$ is the 3×3 information matrix describing the uncertainty in the relative rotation component of the loop closure pose $\bar{\boldsymbol{T}}_{ij}$, and $\bar{\boldsymbol{t}}_{ij}$ is the translation direction (assumed to be a unit-norm vector) in $\bar{\boldsymbol{T}}_{ij}$. The matrix $\boldsymbol{M} = \mathbf{I}_3 - \bar{\boldsymbol{t}}_{ij} \bar{\boldsymbol{t}}_{ij}^{\mathsf{T}}$ is the orthogonal projector of the vector $\bar{\boldsymbol{t}}_{ij}$ and is such that, for any 3D vector $\boldsymbol{v}$, $\boldsymbol{M}\boldsymbol{v} = 0$ if $\boldsymbol{v}$ is aligned with $\bar{\boldsymbol{t}}_{ij}$. Intuitively, the matrix simply disregards the component of the translation error in the direction of $\bar{\boldsymbol{t}}_{ij}$.

Rotation-only Loop Closure Computation. This last variant is similar to the previous one, but it disregards the translation component and only uses the relative rotation (computed via the 5-point method) of the loop closure. In our implementation, we still use the factor (3.1) but set the translation information matrix —*i.e.,* the bottom-right block in (3.2)— to zero.

Free-space Mapping via Kimera-Semantics. In our previous work [62, 60], we used Kimera-Semantics to generate dense semantically annotated 3D meshes from stereo or depth camera data. As the system in this chapter uses multiple non-overlapping monocular cameras, there is no simple way to use stereo reconstruction to generate a dense depth map. As we are only concerned with free-space mapping

and since the road can be assumed to be locally planar, we first detect the ground plane in the image using a CNN for pixel-wise binary classification, and then map the ground plane to a 3D plane using a homography transformation [29, Chapter 13]. The homography matrix is calculated during camera calibration for each camera. Using the homography, we map every pixel belonging to the ground plane to a 3D point and then pass the corresponding 3D point cloud to Kimera-Semantics, which performs ray-casting to infer a 3D voxel-based map, and then extracts a textured 3D mesh via marching cubes.

3.3 Experiments

This section showcases the effectiveness of each component of the proposed system, including the visual-inertial odometry (Section 3.3.1), the loop closure modeling (Section 3.3.2), and the free-space mapping (Section 3.3.3). The results are also compared against state-of-the-art VI-SLAM methods.

3.3.1 Visual-Inertial Odometry

Table 3.1 shows the performance of our monocular extension of Kimera-VIO, comparing the performance of each camera in isolation. We ran each dataset using only one camera at a time, all with the same IMU data. Both trajectory ATE RMSE and drift are reported for each configuration and each dataset, and the best results for each dataset are highlighted in green. The table covers both simulated datasets ("sim") as well as real outdoors ("out") and indoor ("in") datasets. Overall, the rear camera performed poorly as compared to the rest of the cameras, likely due to the fact that the camera was tilted up slightly. This resulted in few nearby features to track as the sky/ceiling took up most of the image. The front camera had the highest number of best-scores on the real datasets, while the left and right cameras were superior in simulation and performed reasonably well on the real datasets. The left and right cameras were mounted on the car's side mirrors, which were not stable mounting platforms. It is possible that the extrinsic calibration of the cameras changed between

datasets when the car doors were opened, or during driving if the mirrors experienced any flex. This is likely the reason the front-camera, which was more rigidly mounted, performed better on the real datasets.

	VIO Absolute Translation Error							
	MonoFront		**MonoLeft**		**MonoRear**		**MonoRight**	
Dataset	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]
sim1 (250m)	5.4	2.2	0.9	0.3	6.9	2.8	1.1	0.4
sim2 (468m)	1.5	0.3	1.4	0.3	2.7	0.6	0.9	0.2
sim3 (749m)	6.1	0.8	-	-	4.7	0.6	3.3	0.4
sim4 (804m)	7.1	0.9	5.5	0.7	13.3	1.7	-	-
out0 (498m)	0.5	0.1	1.3	0.3	1.9	0.4	6.1	1.2
out1 (445m)	1.6	0.4	1.8	0.4	4.7	1.0	4.0	0.9
out2 (521m)	1.9	0.4	1.4	0.3	195.6	37.5	3.3	0.6
out3 (807m)	7.2	0.9	3.8	0.5	10.7	1.3	7.2	0.9
out4 (537m)	1.5	0.3	0.5	0.1	5.1	0.9	1.3	0.2
out5 (514m)	1.3	0.2	2.0	0.4	1.7	0.3	29.4	5.7
out6 (448m)	3.1	0.7	2.6	0.6	4.5	1.0	3.0	0.7
out7 (406m)	1.2	0.3	2.1	0.5	24.0	5.9	3.9	1.0
out8 (48m)	1.5	3.1	1.0	2.0	8.9	2.5	0.3	0.6
out9 (415m)	2.3	0.5	2.0	0.5	6.6	1.6	1.5	0.4
out10 (486m)	2.2	0.4	1.4	0.3	5.9	1.2	2.1	0.4
out11 (44m)	0.8	1.8	0.4	0.9	4.2	1.0	0.7	1.5
out12 (437m)	1.2	0.3	1.4	0.3	162.4	37.2	2.5	0.6
out13 (341m)	0.9	0.3	1.2	0.4	34.4	10.1	1.3	0.4
out14 (517m)	1.2	0.2	1.3	0.2	6.9	1.3	1.7	0.3
out15 (194m)	0.4	0.2	0.5	0.3	2.1	1.1	1.1	0.6
in0 (421m)	3.2	0.8	3.1	0.7	12.5	3.0	3.0	0.7
in1 (321m)	4.8	1.5	3.5	1.1	5.5	1.7	4.3	1.4
in2 (563m)	12.2	2.2	65.8	11.7	13.4	2.4	10.0	1.8
in3 (416m)	4.2	1.0	11.7	2.8	6.5	1.6	11.1	2.7
in4 (723m)	22.8	3.1	9.7	1.3	24.7	3.4	8.9	1.2
in5 (647m)	18.8	2.9	14.0	2.2	32.3	5.0	11.4	1.8

Table 3.1: VIO accuracy for each of the four cameras. Best results for each dataset are highlighted in green. Dashes are used to indicate tracking failures (drift $> 100\%$).

Table 3.2 shows the performance of the multi-camera configurations of Kimera. In the 1-camera configuration, only the left camera was used. For the 2-camera configuration, both the left and right cameras were used. Then we added the front camera, and finally the rear camera. The results highlighted in green are the best results, and those highlighted in blue are the second-best. Our expectation was that estimation error would decrease with each added camera, however this was not

universally true. In the case of the simulated datasets, for 3 of the 4 datasets the best performance was found in the 4-camera configuration, while one dataset did best in the 2-camera configuration. However even in this case the 4-camera results were not far off, and the degraded performance was likely due to the fact that the front and rear cameras in this dataset would have captured a lot of featureless planes as it was recorded in a section of the simulation environment with more skyline and empty roads. For the real datasets, it was clear that the 1-camera configuration had the highest number of best-performance scores, though the other configurations also performed well in most cases. The degradation of performance with added sensors was likely due to the added error in extrinsic calibration with each camera, which could be partially alleviated through the use of extrinsic auto-calibration. In particular the 4-camera results for the real datasets were generally the poorest, which makes sense given the performance of the rear-camera in the monocular-VIO ablation study (Table 3.1) were consistently subpar.

The final columns are walled off from the rest of the results as these were taken from the monocular VIO with the proprietary external odometry. In this case, wheel odometry was fed into Kimera-VIO and included in the backend as an odometry factor. This greatly improved estimation error, and the bolded results are the true best results of the table for the associated dataset. However these were separated from the rest of the table because comparing against pure VIO systems directly would have been inappropriate. Additionally, this data was not available in the simulated datasets or in indoor datasets. For the indoor datasets, wheel odometry was used in place of ground truth as the indoor scenarios were GPS-denied environments, so ground-truth was unavailable.

Table 3.3 shows the estimation performance of Kimera-VIO with one camera compared against two state of the art monocular VIO systems: Vins-Fusion [59], and Open-Vins [28]. Results are shown for VIO systems without loop-closure. As these competitors do not support multiple surround-view cameras, the comparison is done in monocular mode for fairness. For each of the competitors, parameters were tuned for the best performance using an automated parameter regression script. All datasets

Dataset	VIO Absolute Translation Error									
	1cam		**2cam**		**3cam**		**4cam**		**1camWheel**	
	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]
sim1 (250m)	0.6	0.2	0.7	0.3	0.6	0.2	0.4	0.2		
sim2 (468m)	1.8	0.4	1.4	0.3	1.6	0.3	1.2	0.3		
sim3 (749m)	2.8	0.4	1.8	0.2	1.5	0.2	1.2	0.2		
sim4 (804m)	3.2	0.4	1.8	0.2	2.5	0.3	1.9	0.2		
out0 (498m)	6.1	1.2	8.4	1.7	2.4	0.5	3.6	0.7	**1.8**	**0.4**
out1 (445m)	3.5	0.8	2.3	0.5	3.4	0.8	4.9	1.1	**0.7**	**0.2**
out2 (521m)	3.3	0.6	2.0	0.4	2.2	0.4	2.6	0.5	**0.7**	**0.1**
out3 (807m)	6.7	0.8	2.9	0.4	1.8	0.2	9.7	1.2	**1.1**	**0.1**
out4 (537m)	1.3	0.2	4.6	0.9	5.8	1.1	3.3	0.6	1.5	0.3
out5 (514m)	26.7	5.2	27.8	5.4	31.9	6.2	-	-	**1.7**	**0.3**
out6 (448m)	2.6	0.6	4.0	0.9	6.4	1.4	6.9	1.5	**1.0**	**0.2**
out7 (406m)	3.6	0.9	4.9	1.2	8.8	2.2	6.0	1.5	**1.2**	**0.3**
out8 (48m)	0.3	0.6	1.5	3.1	0.4	0.9	2.9	6.1	**0.3**	**0.6**
out9 (415m)	2.0	0.5	1.3	2.3	0.7	1.3	3.9	6.9	1.2	0.3
out10 (486m)	2.0	0.4	1.8	0.4	1.5	0.3	3.8	0.8	**1.1**	**0.2**
out11 (44m)	0.7	1.5	0.6	1.5	0.2	0.6	1.0	2.2	0.3	0.7
out12 (437m)	2.4	0.6	4.3	1.0	4.8	1.1	86.7	19.8	**1.1**	**0.2**
out13 (341m)	1.5	0.4	0.8	0.2	0.5	0.2	1.1	0.3	0.9	0.3
out14 (517m)	1.4	0.3	1.3	0.3	1.9	0.4	2.0	0.4	**0.9**	**0.2**
out15 (194m)	1.1	0.6	0.7	0.4	0.8	0.4	1.0	0.5	**0.2**	**0.1**
in0 (421m)	3.1	0.7	28.4	6.7	340.0	80.7	146.2	34.7		
in1 (321m)	4.4	1.4	-	-	68.3	21.3	103.4	32.2		
in2 (563m)	12.4	2.2	56.9	10.1	53.9	9.6	68.5	12.2		
in3 (416m)	16.4	3.9	-	-	325.9	78.2	405.3	97.2		
in4 (723m)	16.4	2.3	217.7	30.1	691.7	95.6	645.6	89.2		
in5 (647m)	12.4	1.9	322.3	49.7	93.8	14.5	115.4	17.8		

Table 3.2: Multi-camera VIO accuracy. Dataset length is omitted for brevity, see Tables 3.3 or 3.1 for length. Best results are in green, second best in blue. Dashes are used to indicate tracking failures (drift $>$ 100%). The last column uses external odometry, results are boldfaced in cases where this is the best result. Wheel odometry was not present in simulated datasets or indoor datasets.

were evaluated using each of the four cameras for all competitors, and the best camera system was picked for each pipeline. For Vins-Fusion, the rear camera was used for all datasets. For all other systems, the left camera was used for all datasets. For Kimera-VIO, we used the left camera as well even though the front camera outperformed it for Kimera-VIO, to be fair to the competitors. Kimera-VIO out-performed both Vins-Fusion and Open-Vins in all but 4 datasets. Kimera-VIO struggled most with indoor scenes, though was within 0.3% error from Vins-Fusion in all three of the

indoor datasets in which it underperformed. Parameters for competitor pipelines are provided at github.com/MIT-SPARK/ford-paper-params.

	VIO Error (No Loop Closures) Absolute Translation Error					
	Kimera-1cam		**Vins-Fusion**		**Open-Vins**	
Dataset	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]
sim1 (250m)	0.6	0.2	-	-	15.0	6.0
sim2 (468m)	1.8	0.4	-	-	-	-
sim3 (749m)	2.8	0.4	-	-	5.0	0.7
sim4 (810m)	3.2	0.4	-	-	17.0	2.1
out0 (498m)	6.1	1.2	-	-	112.2	22.6
out1 (445m)	3.5	0.8	202.9	55.9	419.7	94.4
out2 (521m)	3.3	0.6	-	-	42.1	8.1
out3 (807m)	6.7	0.8	-	-	-	-
out4 (537m)	1.3	0.2	-	-	-	-
out5 (514m)	26.7	5.2	12.1	3.2	96.5	18.8
out6 (448m)	2.6	0.6	3.3	0.9	17.0	3.8
out7 (406m)	3.6	0.9	3.6	1.3	22.2	5.5
out8 (48m)	0.3	0.6	10.2	4.0	18.1	5.1
out9 (415m)	2.0	0.5	5.3	1.8	13.3	3.2
out10 (486m)	2.0	0.4	10.7	2.9	35.0	7.2
out11 (44m)	0.7	1.5	4.7	1.8	20.3	4.9
out12 (437m)	2.4	0.6	8.0	2.2	45.1	10.3
out13 (341m)	1.5	0.4	2.7	1.2	15.1	4.4
out14 (517m)	1.4	0.3	4.7	1.3	33.5	6.5
out15 (194m)	1.1	0.6	1.9	1.5	7.2	3.7
in0 (421m)	3.1	0.7	10.2	3.5	81.6	19.3
in1 (321m)	4.4	1.4	2.6	1.1	79.8	24.8
in2 (563m)	12.4	2.2	10.7	2.4	97.2	17.2
in3 (417m)	16.4	3.9	-	-	63.1	15.1
in4 (723m)	16.4	2.3	15.6	2.3	191.1	26.4
in5 (647m)	12.4	1.9	13.2	2.1	420.5	64.9

Table 3.3: VIO accuracy (no loop closures) of Kimera, Vins-Fusion, and Open-Vins. Best results for each dataset are highlighted in green. Dashes are used to indicate tracking failures (drift $> 100\%$).

3.3.2 Loop-Closure Detection

The proposed system provides several schemes for closing the SLAM loop by performing loop-closures on the monocular image data. As the cameras had very little image overlap, it was not feasible to use the standard stereo-matching methods to generate depth data for calculating relative poses between loop closure match candi-

dates as in [62, 60]. Table 3.4 shows the results of an ablation study on these various methods for performing loop closures. The impact of each method on the estimation error and drift of the SLAM system are shown, with best results highlighted in green. The first pair of columns are for the VIO system without loop closures ("VIO"). The second is using the proposed scale-less factor ("Scale-less Factor"). The third uses the rotation-only factor ("Rot Only"). The fourth uses PnP to estimate the relative pose ("PnP"). Only datasets with loops are included in the analysis. The proposed scale-less factor was far superior to the other methods and to simple VIO except in a few select circumstances, and even in those cases RPGO was able to reject bad loop closure candidates and prevent the estimate from being worse than the simple VIO estimate. In some cases this method was able to reduce drift by a factor of four, and in most datasets we were able to obtain drift less than 1% of the trajectory length.

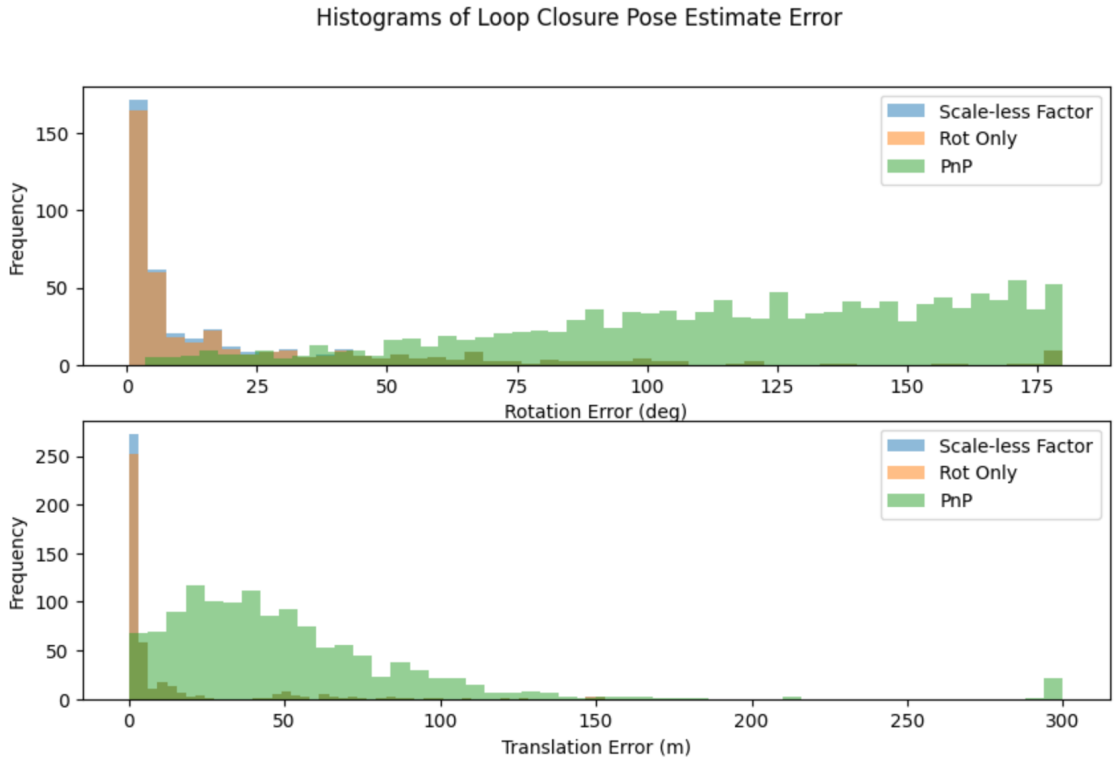


Figure 3-3: Histograms of the proposed loop-closure detection methods. Each bin is error on rotation and translation, and contains the sum total of all loop-closure candidates across all datasets that scored within that bin.

Figure 3-3 compares histograms of the rotation and translation errors for loop

	Loop Closure Detection Ablation Study Absolute Translation Error							
	VIO		**Scale-less Factor**		**Rot Only**		**PnP**	
Dataset	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]
sim2 (468m)	1.8	0.4	0.5	0.1	1.1	0.2	0.8	0.2
sim3 (749m)	2.8	0.4	2.5	0.3	4.0	0.5	3.3	0.4
sim4 (804m)	3.2	0.4	3.2	0.4	3.2	0.4	3.2	0.4
out6 (448m)	2.6	0.6	2.6	0.6	2.6	0.6	2.6	0.6
out7 (290m)	9.7	2.4	3.9	1.0	7.7	2.7	13.0	3.2
out8 (48m)	0.4	0.9	0.4	0.9	0.2	0.5	0.3	0.7
out9 (415m)	1.6	0.4	1.3	0.3	2.0	0.5	4.0	1.0
out10 (486m)	2.4	0.5	1.6	0.3	2.6	0.5	2.2	0.4
out11 (44m)	0.7	1.5	0.6	1.5	0.7	1.5	0.7	1.5
out12 (437m)	2.4	0.6	1.9	0.4	2.6	0.6	6.4	1.5
out13 (341m)	1.5	0.4	1.2	0.3	1.6	0.5	1.1	0.3
out14 (517m)	1.4	0.3	1.1	0.2	1.2	0.2	4.0	0.8
in0 (421m)	3.1	0.7	2.7	0.7	3.1	0.7	5.5	1.3
in1 (321m)	4.4	1.4	2.0	0.6	5.4	1.7	3.1	1.0
in2 (563m)	12.4	2.2	12.4	2.2	12.4	2.2	12.7	2.3
in3 (417m)	16.5	4.0	14.1	3.4	16.4	3.9	16.4	3.9
in4 (723m)	16.4	2.3	12.8	1.8	16.3	2.3	16.4	2.3
in5 (647m)	12.3	1.9	12.3	1.9	12.4	1.9	12.4	1.9

Table 3.4: Pose estimation accuracy (including loop closures) restricted to datasets that contain loops. First column is VIO only (no loop closures), and all configurations use only 1 camera.

closure candidates generated by all three methods. It is clear that the scale-less approach has the lowest errors consistently, and that the PnP method failed to get accurate relative poses. In the PnP method, RANSAC found on average of 5 inliers across all candidates across all datasets. This is likely due to a mismatch between the detected ORB keypoints and the backend-tracked 3D landmarks. On the other hand, both the scale-less and the rotation-only approaches had upwards of 10 inliers in average.

Table 3.5 reports the VIO performance of Kimera (monocular) with loop closures against Vins-Fusion with Loop-Fusion [59] and ORB-SLAM3 [14]. All three pipelines use visual-bag-of-words to generate loop closure candidates [27]. Only datasets with loop closures are included for brevity, and the scale-less approach is used for relative pose estimation in Kimera. Open-Vins did not have a functioning loop-closure system

	VI-SLAM Comparison With Loop Closures Absolute Translation Error					
	Kimera-1cam		**Vins-Fusion**		**ORB-SLAM3**	
Dataset	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]	RMSE [m]	Drift [%]
sim2 (468m)	0.5	0.1	-	-	22.6	4.8
sim3 (749m)	2.5	0.3	-	-	61.2	8.2
sim4 (810m)	3.2	0.4	-	-	15.3	1.9
out6 (448m)	2.6	0.6	3.4	0.8	37.8	8.4
out7 (406m)	3.9	1.0	4.1	1.0	8.9	2.2
out8 (48m)	0.4	0.9	12.2	3.4	12.7	3.6
out9 (415m)	1.3	0.3	5.2	1.3	6.4	1.5
out10 (486m)	1.6	0.3	14.3	2.9	10.5	2.2
out11 (44m)	0.6	1.5	7.9	1.9	9.0	2.2
out12 (437m)	1.9	0.4	7.5	1.7	1.9	1.8
out13 (341m)	1.2	0.3	3.3	1.0	11.9	3.5
out14 (517m)	1.1	0.2	5.3	1.0	20.9	4.0
in0 (421m)	2.7	0.7	10.4	2.5	3.6	0.8
in1 (321m)	2.0	0.6	2.7	0.8	7.2	2.2
in2 (563m)	12.4	2.2	13.2	2.3	43.4	15.4
in3 (417m)	14.1	3.4	-	-	48.1	11.5
in4 (723m)	12.8	1.8	16.3	2.3	50.6	7.0
in5 (647m)	12.3	1.9	15.3	2.4	33.0	5.1

Table 3.5: Pose estimation accuracy (including loop closures) for Kimera, Vins-Fusion, and ORB-SLAM3. Best results for each dataset are highlighted in green. Dashes are used to indicate tracking failures (drift $> 100\%$).

and so it was not included. ORB-SLAM3 is a SLAM-only pipeline so to turn off loop-closures and use it as a VIO-only pipeline would have been an unfair evaluation, so it was not included in Table 3.3. Kimera beats the other competitors in all of the datasets in this case.

3.3.3 Ground Plane Reconstruction

Table 3.6 shows geometric reconstruction accuracy (as defined in [62]) for the simulated datasets, where ground-truth ground-plane maps are available. The system is evaluated using both ground-truth poses as well as poses from Kimera. Kimera was used in the 1-camera configuration without wheel odometry. The third column of Table 3.6 shows the reconstruction accuracy when using the RPGO trajectory: more precisely, in this case we use the pose graph and mesh optimization approach in [62] to jointly optimize the mesh and trajectory. Some additional trajectory error comes

from using Kimera's poses in all of the simulated datasets, which is expected. However, we observe that the results in the last column remain close to the ones obtained with ground-truth poses.

	Kimera-Semantics Geometric Reconstruction Accuracy ATE RMSE [m]		
Dataset	Homography GT Poses	Homography Kimera-VIO Poses	Homography Kimera-RPGO Poses
sim1 (250m)	0.22	0.26	0.22
sim2 (468m)	0.37	0.40	0.37
sim3 (749m)	0.23	0.32	0.29
sim4 (810m)	0.29	0.35	0.34

Table 3.6: Geometric reconstruction accuracy for the modified Kimera-Semantics using three different configurations.

Figure 3-4 shows several free-space reconstructions of the Ford datasets using the proposed homography-based method. The maps were generated using Kimera-VIO's pose with 4-cameras and external odometry.

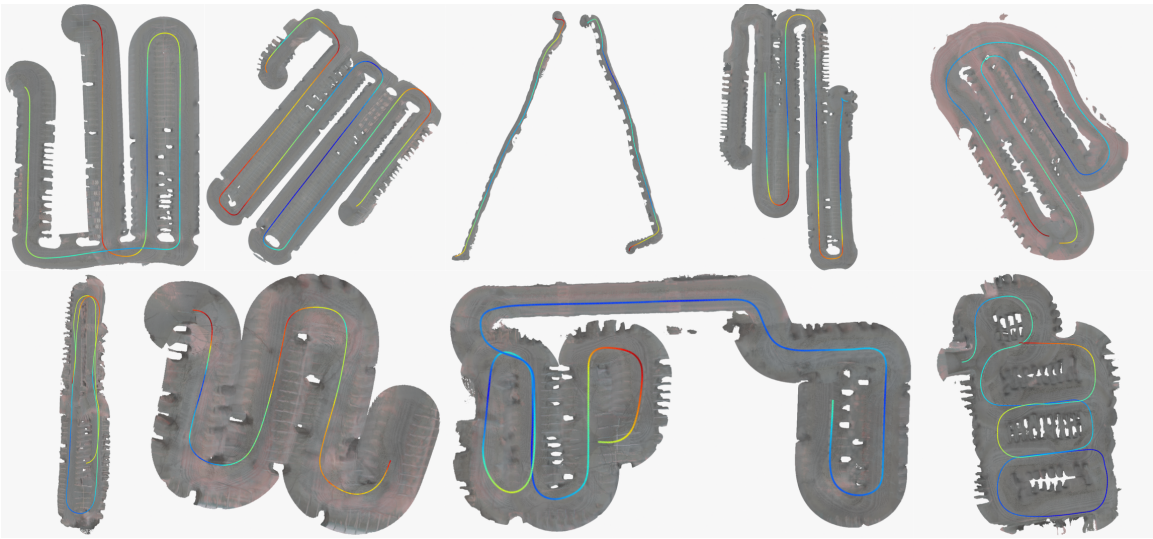


Figure 3-4: 3D reconstructions produced by the proposed free-space mapping approach on several Ford datasets. All four cameras were used for reconstruction, and Kimera's visual-inertial odometry was performed with all four cameras and external odometry. A colormap of the estimated trajectory is plotted over each reconstruction, with cooler colors representing lower ATE RMSE.

3.4 Conclusions

We proposed important modifications to Kimera to support monocular and multi-camera input data, and to incorporate external (wheel) odometry inputs. To complete the SLAM system, we modified the loop-closure module to utilize monocular inputs. Additionally, Kimera-Semantics was modified to perform efficient free-space mapping for autonomous valet parking applications, which works with cameras with non-overlapping FOV and does not require learning for depth-estimation. We tested the system on simulated car data and real-world datasets collected with a test car at the Ford Motor Company. The proposed system exhibits small trajectory and mapping errors and consistently outperforms state-of-the-art open-source VIO and VI-SLAM systems. Real-world results for the multi-camera system also suggest room for improvement, in particular with respect to extrinsic calibration. Automatic calibration in the backend factor graph could alleviate some of the issues. Additionally, Kimera-Semantics supports semantic-annotations for any number of object classes; therefore, a potential extension could include mapping other semantic classes relevant to autonomous parking.

Chapter 4

Pushing the Boundary of Kimera and Open-Source VI-SLAM Systems

4.1 Introduction

Kimera in its original form [60, 62] was open-sourced under a permissive MIT license for use by the broader research community as well as industry. Since then, several other open-source pipelines have been released and updated to compete with Kimera's performance on publically available datasets such as EuRoC [12]. Vins-Mono [58] received an update to Vins-Fusion [59] to support stereo-IMU inputs. ORB-SLAM2 [54] was updated more recently to ORB-SLAM3 [14]. End-users for these VI-SLAM pipelines can have diverse system requirements, but generally desire fast (online) performance and accurate state estimation and mapping. Occasional third-party surveys are performed to provide the community with a basic understanding of performance and capabilities in the wild. In many of these studies, Kimera performs well in terms of state estimation error with online performance [68, 34]. However, as Kimera was originally designed for stereo-IMU inputs on a relatively limited set of publically available datasets (EuRoC) [12], most of these surveys do not properly evaluate Kimera's performance on a wide range of platforms but instead understandably opt to use the baseline configurations we provided in the original release. In the years since the initial release in 2019, Kimera has also evolved and improved in its performance as it has

been deployed in other contexts. While some of the modifications to Kimera discussed in Chapter 3, such as the multi-camera support (Section 3.2.2), were not open-sourced due to their development as part of a collaboration with The Ford Motor Company, several more features were implemented in synergistic efforts to improve VIO tracking performance, robust pose graph optimization, and semantic-mapping that were not discussed in Chapter 3. Additionally, Kimera-Multi [17, 73] and Hydra [32] made improvements to Kimera-VIO's tracking to serve as a baseline VI-SLAM pipeline for Multi-Robot Mapping and 3D Scene-Graph Creation respectively.

In this chapter, we present several improvements to the open-source version of Kimera that will soon be relased to the public as a part of a version-update to various Kimera packages, including key modules such as Kimera-VIO, Kimera-Semantics, Kimera-RPGO, and Kimera-PGMO. Many of these improvements relate to Kimera-VIO and in particular the frontend of the system, while there are also improvements to Kimera-RPGO and Kimera-Semantics. To that end, we showcase ablation studies on select features added to Kimera since its release in 2019, and perform comparisons against the state of the art (ORB-SLAM3 and Vins-Fusion). We evaluate Kimera's strengths and weaknesses as well as its flexibility in usage and adaptability to variuos platforms. Experiments are conducted on datasets gathered from various real-life platforms as well as simulated environments.

This chapter is organized as follows: Section 4.2 provides a detailed explanation of selected features added to Kimera that will be evaluated. Section 4.3 describes ablation tests and comparisons against the state of the art, and presents the results of various experiments. Finally, Section 4.4 concludes the chapter and provides avenues of future research and development.

4.2 Improvements to Kimera

Newly implemented features are broken up into two groups. First, in Section 4.2.1 we show improvements in Kimera-VIO's frontend; these consist of visual-tracking improvement as well as pipeline input flexibility enhancements. Then, in Section 4.2.2 we

demonstrate improvements to Kimera-VIO's backend and pose-graph-optimization.

4.2.1 Kimera-VIO Frontend

Kimera-VIO's frontend serves as an initial data-processing module to prepare raw sensor measurements for optimization in the backend. The frontend is flexible enough to be implemented for a variety of sensor inputs. In the original version of Kimera-VIO [60, 62] the frontend was implemented for stereo cameras and IMU, assumed synchronized. Kimera was designed from the ground up to be highly modular and developer-friendly, to support expansion in the future. This allowed us to quickly add implementations for monocular-IMU input, RGB-D cameras with IMU, and external (*e.g.,* wheel) odometry, which we have since released to the community. We have also implemented approximate synchronization between camera and IMU at the level of the Kimera-VIO library. Finally, we improved the keyframe management and feature extraction process.

External odometry. Various experimental platforms that have seen use with Kimera provide alternative odometric inputs that would remain un-used by most open-source VI-SLAM pipelines. For example, in Chapter 3 we included the wheel-odometry provided by the localization stack onboard Ford's self-driving car. Some cameras that are widely available to the VI-SLAM research community, such as the Realsense T265, provide odometry from onboard VIO for ease-of-use by roboticists. Platforms with onboard LiDAR sensors can provide additional odometry using a fast LiDAR-SLAM implementation. Combining these various inputs with Kimera-VIO's estimate without implementing sensor-specific frontends serves as a fast way to improve the state estimate without slowing down Kimera's pose-estimation thread. Therefore, we developed code to optionally process external odometry as relative poses between measurements in a separate submodule of the frontend. These relative poses are chained together between keyframes and passed to the backend alongside visual features and preintegrated IMU measurements. The backend then combines the visual features (stereo, mono, or RGB-D), IMU measurements, and odometric measurements in the factor graph. More formally, the relative pose factors describing

the odometry measurements take the following form:

$$f(\boldsymbol{T}_i, \boldsymbol{T}_j) = \left\| \text{Log}(\bar{\boldsymbol{T}}_{ij}^{-1} \boldsymbol{T}_i^{-1} \boldsymbol{T}_j) \right\|_{\boldsymbol{\Omega}_{ij}}^2 \tag{4.1}$$

where the external odometry relative pose $\bar{\boldsymbol{T}}_{ij} \in \text{SE}(3)$ relates two poses $\boldsymbol{T}_i, \boldsymbol{T}_j \in \text{SE}(3)$ along the trajectory of the robot, as in 3.1. Note that once again, $\text{Log}(\cdot)$ is the standard logarithm map. Unlike in the case of the scale-less loop closure factor (described in 3.2), the noise model associated with the external odometry factor is a standard isotropic diagonal model:

$$\boldsymbol{\Omega}_{ij} = \begin{bmatrix} \boldsymbol{\Omega}_R & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{\Omega}_t \end{bmatrix} \tag{4.2}$$

where $\boldsymbol{\Omega}_R$ and $\boldsymbol{\Omega}_t$ are diagonal matrices describing the precision of the measurements.

External odometry processing can happen at any rate, however since backend optimization only happens at keyframe rate the odometry is only passed to the backend with keyframes. The user may set whatever noise model they wish to be included in the factor graph with each chain of odometric measurements by providing the precision of the odometry's estimation of rotation and translation. This precision corresponds to the "trust" which is given to the measurement; high precision indicates that the external odometry is expected to be very accurate. The factor graph optimization takes this noise model in account directly in optimization and marginalization.

Feature binning and non-max suppression. For visual inputs, we implement two small improvements that enable more efficient processing of images and keypoint tracking. Feature binning allows the user to provide an abstracted pixel-mask of the image defining which portions of the image to include in feature detection and which to ignore. This is most useful in situations where parts of the image are expected to be unusable: for example, in Chapter 3, the Ford test car cameras were fisheye cameras. Parts of the chassis of the car were visible within the field of view of all four cameras. Some drone datasets also include in the field-of-view the drone's rotors. Beyond

these mechanical occlusions, there are situations where one might want to ignore certain parts of the image as a heuristic for performance. For example, in simulation we found that certain orientations for cameras in outdoor environments guaranteed that part of the image would only see the sky, which provided no useful features for motion estimation. By using feature-binning masks, we are able to prevent features detected in these regions from interfering with the feature outlier-rejection problem and damaging the inlier set of features that are passed to the backend for optimization, thereby refining the final state estimate. In addition, we implement various flavors of non-max suppression [15]. This allows the user to determine how aggressively to cull stale feature tracks from the frontend, enabling the use of a large number of features at the feature detection stage without slowing down frontend processing by mandating that all of those features be tracked through the entire tracking window. For high resolution cameras operating in large, empty scenes, this dramatically improves the odds of outlier-rejection selecting a good set of inliers.

Keyframe logic. Kimera's frontend is tasked with determining which of the incoming frames are designated as keyframes. Keyframes are specific frames chosen at regular intervals to trigger and take part in backend factor-graph optimization. At the time a keyframe is identified, all frontend measurements (including visual features and pre-integrated IMU measurements and any other optional data) between the previous keyframe and the current keyframe are sent to the backend for inclusion in the factor graph. This is important as camera frame rates on modern robotic platforms typically meet or exceed 20 frames per second, which is much faster than is necessary for accurate pose estimation. Performing factor-graph optimization at these rates would preclude real-time performance on most robotic platforms, in particular resource constrained ones such as drones or other small autonomous agents. By restricting optimization to only keyframes, which are a subset of the total frames, we can include more visual mesaurements in the factor graph without slowing down the optimization thread. Kimera's previous logic for choosing keyframes was fixed by a parameter set by the user (`intra_keyframe_time_ns`) which determined the elapsed time between keyframes. In addition, keyframes were chosen if the number of

tracked features in the image dropped below a threshold count, so as to trigger more frequent backend optimizations during regions of visual uncertainty. This formulation was simple, yet effective for the majority of test cases, in particular for aerial vehicles which are constantly moving. However, for other vehicles such as cars, the platform had long periods of minimal-to-zero movement. During these times, choosing keyframes and triggering backend optimizations at a constant rate was unnecessary as the pose had not deviated significantly from the previous keyframe. For this reason we modified the keyframe logic selection to choose keyframes either when a maximum amount of time had passed since the previous keyframe, or when there was sufficient disparity between keyframes (in terms of optical flow of the features) to warrant a new keyframe. The latter condition pushed the frontend to only select keyframes after the robot had moved, saving on computation. Additionally, because the backend factor graph operates on a sliding time-horizon (local optimization), by choosing keyframes only after the robot has moved we prevent Kimera from forgetting the entire recent trajectory prior to the robot standing still. In these cases the previous version of the backend would frequently reach degenerate conditions as the robot had not moved at all within the fixed time horizon, leading to frequent failures. In the updated version of Kimera, the user may tune either condition for keyframe selection to suit their needs. This generally leads to smaller factor-graph sizes while retaining enough information about the past to maintain tracking during longer periods of minimal movement.

4.2.2 Kimera-VIO Backend and Kimera-RPGO

Kimera-VIO's backend creates and optimizes a factor graph of various measurements collected from the frontend over a receding horizon, to estimate the robot odometry. It does not perform global optimization - this is handled instead by Kimera-RPGO (Robust Pose GRaph Optimization) through the use of Kimear-LCD (Loop-Closure Detection) - however the state estimate obtained by the VIO backend is generated at keyframe rate and is fast enough for applications requiring online localization. The Kimera-VIO backend uses GTSAM [23] as its factor-graph optimization engine, and

performs marginalization to remove factors outside the receding horizon. Odometric measurements are then passed to Kimera-LCD. Kimera-LCD processes backend odometry in conjunction with frontend data (images) associated with each keyframe in order to identify loop closures, using a visual-Bag-of-Words approach [27]. Both odometry factors and loop-closure factors are added to a separate pose-graph which is optimized using Kimera-RPGO [60, 62].

In the past, RPGO relied on Incremental Consistent Measurement Set Maximization (PCM) [52] for outlier rejection on the pose-graph. This enabled the rejection of bad loop-closure candidates, which can be frequent when using the visual-Bag-of-Words method in scenes where the environment is visually similar in many areas. Rosinol *et al.* [62] showed that Kimera-RPGO with PCM led to drastic improvements in global pose estimation. However, since then newer outlier rejection methods have come to the forefront of the field. Yang *et al.* [83] presented Graduated-Non-Convexity (GNC) in conjunction with non-minimal solvers to optimize pose-graphs, among other applications. The authors proposed a general-purpose approach and validated its superiority to RANSAC and PCM on several applications [6], including the pose-graph-optmization (PGO) problem. As this is relevant to VI-SLAM, GNC is now implemented in Kimera-RPGO as an option for outlier rejection on the pose-graph optimization. Finally, Rosinol *et al.* [62] presented Kimera-PGMO for jointly optimizing the pose-graph and the dense volumetric mesh. GNC can be used here as well since the underlying optimization framework is shared with Kimera-RPGO, so we have also modified Kimera-PGMO to use GNC for more accurate mesh reconstruction.

4.3 Experiments

As Kimera is easily adaptable to a variety of robotic platforms, in this section we provide experimental results for Kimera on a diverse array of datasets. Each of the new features discussed in Section 4.2 are validated in ablation studies. Additionally, we provide comparisons against the state-of-the-art (ORB-SLAM3 [14] and Vins-Fusion [59]). The experimental analysis is organized as follows: Section 4.3.1

discusses the datasets used in our evaluation. Section 4.3.2 discusses the effect of including external odometry in the factor graph for datasets that have external odometry sources. Section 4.3.3 showcases an ablation study on the effect of feature binning. Section 4.3.4 presents the effect of the changes to keyframe logic discussed in Section 4.2.1 on localization error. Section 4.3.5 shows an ablation study on the effect of using `GNC` as compared to just `PCM`. Section 4.3.6 presents the effect of PGMO (joint pose-graph-mesh optimization) on Kimera-Semantics' dense mapping accuracy. Finally, Section 4.3.7 includes an evaluation of Kimera as compared to the state-of-the-art.

4.3.1 Datasets

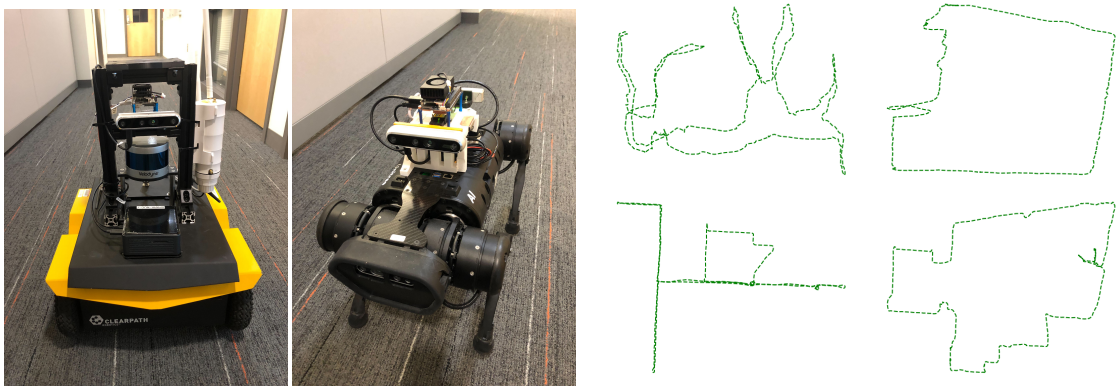
We include results on a wide range of datasets - most of which are publically available - so as to highlight the specific effects of each feature discussed and prove the flexibility of Kimera as a broadly applicable VI-SLAM library.

A1 and Jackal

Many of the datasets come from the Kimera-Multi [17, 73] project, which included a public release of datasets collected on Unitree A1 robots and Clearpath Robotics Jackal robots. The A1 is a quadrapedal robot with an onboard Realsense D455 RGB-D Camera for sensing, as well as IMU and external odometry. The Jackal is a small four-wheeled ground robot with a stereo camera and IMU, as well as wheel odometry. Datasets were recorded in a wide range of locations on MIT's campus, including indoor and outdoor locations, underground tunnels, and an undergraduate dorm.

Datasets labeled `campus_indoor_x` are datasets collected on the Jackal robot in indoor environments across MIT's campus. Datasets labeled `campus_outdoor_x` are Jackal datasets collected in outdoor environments across MIT. Datasets labeled `campus_hybrid_x` are Jackal datasets where the robot transitions from indoor to outdoor or vice-versa. Datasets labeled `simmons_a1_x` are A1 datasets recorded inside an undergraduate dorm hall (Simmons). Each dataset is a single-robot experiment.

More information on the datasets is available in figure 4-1.



(a) (b) (c)

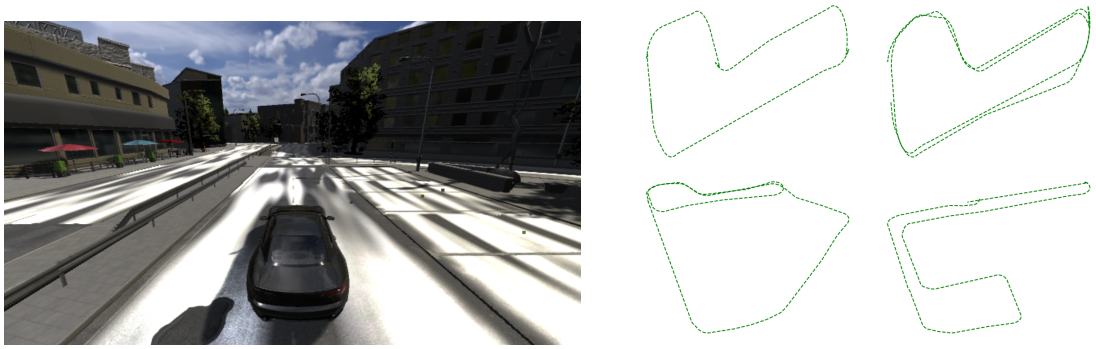
Figure 4-1: KimeraMulti datasets, from the A1 and Jackal robots. 4-1a shows the Clearpath Robotics Jackal, and 4-1b is the Unitree A1. 4-1c shows the ground-truth trajectories of four sequences from this dataset.

Car-Sim

Datasets labeled `car_outdoor_sim_x` are collected inside the TESSE environment [60, 62, 61]. However, unlike in the uHumans2 [60, 62] dataset, these datasets are recorded in a photorealistic simulated outdoor urban environment, using a car as the simulated robotic agent. These datasets were developed as a part of the work presented in Chapter 3. The simulated car has four monocular cameras mounted in the front, rear, left, and right. For these ablation tests, we use Kimera in monocular-mode with the right camera. To see multicamera results, see Section 3.3. More information about the datasets can be found in figure 4-2

uHumans2

The uHumans2 dataset was presented as part of earlier work on Kimera [62]. The simulation environment was also open-sourced, as were the datasets. The agent is a simulated "pill" with a forward-facing stereo camera and simulated IMU. Simulation environments are varied, from a small apartment to an underground subway station. The datasets include varying numbers of dynamic agents (humans) moving in the



(a) (b)

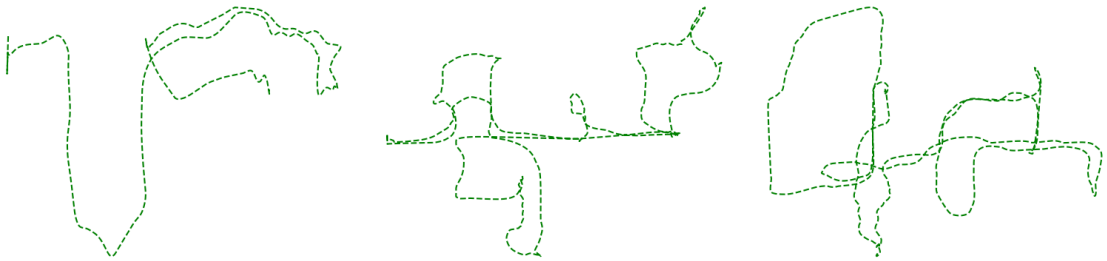
Figure 4-2: Car-sim datasets, collected in a simluation environment. 4-2a shows a third-person POV of the car model in a large urban environment developed in the TESSE simulator. The car has a realistic dynamics model. 4-2b shows the ground-truth trajectories of the four Car-sim datasets. These datasets were also used in Chapter 3.

scene. For the purposes of this chapter, we do not include any datasets with humans. Figure 4-3 provides more information on the datasets.

4.3.2 External Odometry

External odometry was available on the Jackal robot. Table 4.1 shows datasets from the Jackal, and the effect the inclusion of external odometry had on the localization performance. The absolute translation error is reported as RMSE over the entire trajectory. Three trials were performed for each dataset, and the reported metrics are the mean and standard deviation of the ATE RMSE across all trials. Kimera was run with in the stereo-imu configuration, both with and without external odometry measurements. It is clear that the external odometry offers an advantage in many cases, in particular in outdoor datasets. However, in indoor datasets the error was slightly higher with external odometry factors. This is likely because visual features are easier to track in indoor environments, as they are mostly close to the camera, so using stereo-matching to determine feature depth is less erroneous in these cases. Any error in the wheel odometry is therefore more impactful in the factor graph optimization, as the vision factors are so good. Nonetheless, the difference in localization error was relatively small between the two configurations. This was not the case in

(a) Four simulation environments used in the uHumans2 datasets.



(b) Sample ground truth trajectories, not to scale.

Figure 4-3: uHumans2 datasets. Collected in a simulation environment developed for the original Kimera release [60]. The datasets have been released to the community, and the simulator code is open-source. 4-3a shows snapshots of the four simulation environments used, and 4-3b shows some of the ground-truth trajectories from this dataset.

experiments performed in Chapter 3, which were done with monocular cameras. External odometry was significantly more important there, as it helped to fix the scale of the backend optimization problem.

	Absolute Translation Error RMSE			
	Without External Odometry		**With External Odometry**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]
campus_hybrid_0	3.25	0.03	3.21	0.04
campus_hybrid_1	4.1	0.35	3.73	0.39
campus_hybrid_2	–	–	8.3	1.09
campus_hybrid_3	9.67	0.67	11.8	7.02
campus_indoor_0	9.17	1.64	11.4	1.19
campus_indoor_1	3.86	2.0	3.98	0.83
campus_indoor_2	8.67	3.24	6.97	2.05
campus_indoor_3	7.06	0.88	6.04	0.81
campus_outdoor_0	10.6	0.85	10.6	1.99
campus_outdoor_1	15.9	1.02	12.3	1.35
campus_outdoor_2	17.0	3.99	21.3	2.09

Table 4.1: VIO accuracy with and without external (wheel) odometry. Datasets come from the KimeraMulti [17, 73] project, and were taken from Jackal robots. Metrics reported are RMSE. Each experiment was run for 3 trials, reported metrics are mean and standard deviation. The best result for each dataset is highlighted in green.

4.3.3 Feature Binning

Feature binning was performed on the A1 dataset from KimeraMulti, as well as the Car-Sim datasets. In the case of the A1, the binning mask was designed to remove features from the body of the robot, visible in the bottom of the camera image. For the Car-Sim datasets, features typically associated with the sky (center and high in the frame) were masked off to improve performance. Table 4.2 shows the results of this ablation study. In the A1 case, Kimera failed completely without the binning mask, and this was observed in other datasets recorded on the A1 as well. For the Car-Sim datasets, localization error was better across the board when binning was enabled. For applications with known regions of bad features, this method seems to be an effective solution for reducing localization error.

	Absolute Translation Error RMSE			
	No Binning		**Binning**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]
car_outdoor_sim_1	1.22	0.5	0.65	0.03
car_outdoor_sim_2	0.82	0.43	0.51	0.01
car_outdoor_sim_3	3.67	0.76	2.55	0.52
car_outdoor_sim_4	8.52	3.5	3.22	0.26
simmons_a1_0	–	–	1.74	0.33

Table 4.2: VIO accuracy with and without feature binning. Two different sets of datasets were used in this experiment; on top are sequences from the Car-Sim (4.3.1) dataset. These were evaluated using Kimera in monocular mode, with the front camera. The single dataset labeled `simmons_a1_0` came from an A1 robot, and was evalued using Kimera with the RGB-D camera. Best result for each dataset is highlighted in green. A dash is used to denote that Kimera failed to get a reasonable trajectory for that dataset in the given configuration.

4.3.4 Keyframe Logic

In table 4.3, we perform an ablation study on the `max_disparity_since_lkf` parameter in Kimera-VIO's frontend. This determines what the maximum elapsed distance between keyframes can be, in terms of optical flow. The higher the value, the more the features can move in the frame before a keyframe is detected and the backend factor-graph optimization is triggered. When set to `1000`, the system is essentially disabled,

defaulting to the logic of the previous version of Kimera and identifying keyframes based solely on elapsed time. All experiments were done with Kimera's maximum elapsed time between keyframes set to a large number, so keyframes were only identified when disparity exceeded the threshold. We see that the best results are generally biased towards smaller values for `max_disparity_since_lkf`, confirming that disparity in optical flow is a superior method for identifying keyframes. In some cases, the difference between the best and worst result for each dataset are quite large (by an order of magnitude). Overall, selecting a `max_disparity_since_lkf` value between `50-100` appears to give consistently good results.

	Absolute Translation Error RMSE											
	MDSL 25		**MDSL 50**		**MDSL 75**		**MDSL 100**		**MDSL 150**		**MDSL 1000**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]
campus_hybrid_0	3.25	0.12	5.57	2.14	3.74	0.81	3.31	0.04	3.29	0.15	3.27	0.09
campus_hybrid_1	3.25	0.36	2.78	1.48	3.56	0.12	3.41	0.54	3.56	0.66	3.71	0.35
campus_hybrid_2	11.7	0.43	–	–	9.58	2.62	11.6	1.59	17.1	4.65	–	–
campus_hybrid_3	70.6	3.62	54.3	25.2	37.7	42.6	9.79	3.27	26.3	16.2	–	–
campus_indoor_0	1.53	2.63	7.68	0.61	7.99	0.81	7.2	1.76	6.32	1.71	8.56	1.8
campus_indoor_1	2.73	0.49	2.83	0.52	4.59	1.89	2.67	0.31	3.21	0.69	2.91	0.06
campus_indoor_2	13.9	8.1	–	–	9.08	1.76	8.58	3.11	9.07	3.05	6.85	1.15
campus_indoor_3	5.2	2.56	6.36	2.27	4.46	0.24	6.17	2.23	7.38	3.45	4.64	1.38
campus_outdoor_0	4.64	1.38	5.5	0.68	5.04	0.41	4.23	0.61	6.94	3.65	5.33	0.51
campus_outdoor_1	9.78	1.36	12.5	1.56	9.35	2.36	12.4	0.61	13.2	1.05	12.4	1.68
campus_outdoor_2	10.7	2.03	16.2	3.17	15.5	2.48	15.7	0.18	11.9	1.21	16.4	2.95
simmons_a1_0	1.13	2.21	1.66	2.33	0.92	0.64	1.14	4.1	0.99	0.16	1.19	2.10

Table 4.3: VIO accuracy ablation study on keyframe logic for Jackal and A1 datasets. Jackal datasets are prefixed with `campus`, and the last dataset comes from the A1 robot. Each configuration is increasing values for `max_disparity_since_lkf` (last keyframe), which are increasing optical flow requirements between keyframes. Mean and standard-deviation are reported for the RMSE of translation error across 3 trials for each dataset. Dashes are used to denote tracking failures (very high error). The best result for each dataset is highlighted in green.

4.3.5 GNC vs PCM

We compare GNC with a baseline based on Pairwise Consistency Maximization (PCM) [52] for robust pose graph optimization. On all datasets, the rotation threshold for PCM was `0.01` and the translation threshold was `0.05`. There were several loop closure candidates in all of the datasets surveyed. Table 4.4 shows that GNC improves localiza-

tion performance substantially in the majority of cases. Sequences from KimeraMulti and uHumans2 are included. In the case of the A1 robot (`simmons_a1_0`), `GNC` was required to make Kimera work. With only `PCM`, localization error exceeded `100%` of the length of the dataset.

	Absolute Translation Error RMSE			
	PCM		**GNC**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]
campus_hybrid_0	3.28	0.08	3.23	0.03
campus_hybrid_1	3.79	0.16	3.91	0.33
campus_hybrid_2	14.6	5.34	12.3	3.78
campus_hybrid_3	14.7	3.39	11.2	6.62
campus_indoor_0	9.84	3.03	8.02	1.19
campus_indoor_1	3.41	0.9	3.99	1.11
campus_indoor_2	6.39	0.96	6.13	0.73
campus_indoor_3	7.55	3.28	6.33	2.53
campus_outdoor_0	12.4	1.75	11.2	0.69
campus_outdoor_1	13.8	6.52	16.2	0.83
campus_outdoor_2	16.4	4.06	14.0	4.14
simmons_a1_0	–	–	28.5	2.43
uHumans2_apartment	0.1	0.6	0.1	0.0
uHumans2_suburb	–	–	2.47	0.67
uHumans2_office	0.33	0.4	0.33	0.09
uHumans2_subway	4.97	0.91	4.11	1.43

Table 4.4: VI-SLAM accuracy using PCM and GNC for loop closure outlier rejection. Sequences from KimeraMulti (Jackal, A1) are included, along with sequences from uHumans2. Dashes are used to denote tracking failures. The best result for each dataset is highlighted in green.

4.3.6 PGMO

In [62] we showed the effect of Kimera-PGMO on mesh reconstruction. We found that by jointly optimizing the pose-graph with loop closures and the mesh, we were able to close loops on the dense volumetric mesh and obtain a higher accuracy in the mesh. Table 4.5 compares Kimera-Semantics with Kimera-PGMO, where Kimera-Semantics is the original version of the dense mapping algorithm released in [60]. Kimera-PGMO provides better mesh reconstruction accuracy due to the inclusion of loop closure factors.

	Absolute Translation Error RMSE			
	Kimera-Semantics		**Kimera-PGMO**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]
car_outdoor_sim_1	0.26	0.05	0.22	0.06
car_outdoor_sim_2	0.40	0.1	0.37	0.05
car_outdoor_sim_3	0.32	0.11	0.29	0.07
car_outdoor_sim_4	0.35	0.09	0.34	0.12
campus_hybrid_0	0.76	0.01	0.67	0.01
campus_hybrid_1	1.59	0.16	1.42	0.09
campus_hybrid_2	1.61	0.21	1.61	0.21
campus_hybrid_3	2.49	0.9	2.48	0.89
campus_indoor_0	4.73	0.16	4.56	0.19
campus_indoor_1	2.97	0.25	3.08	0.26
campus_indoor_2	4.81	0.39	4.15	0.65
campus_indoor_3	3.34	0.15	3.34	0.13
campus_outdoor_0	2.51	0.26	2.49	0.25
campus_outdoor_1	2.5	0.15	2.49	0.15
campus_outdoor_2	2.19	0.05	2.18	0.06

Table 4.5: Dense semantic map accuracy (ATE RMSE) with and without PGMO. Mean and standard-deviation of ATE RMSE are reported over 3 trials. The best result for each dataset is highlighted in green.

4.3.7 Competitor Evaluation

Since Kimera's original release other VI-SLAM pipelines have also had updates to improve their performance and capabilities. In particular, Vins-Fusion [59], the successor to the popular Vins-Mono [58], is highly regarded in the community. ORB-SLAM3 [14], which provided improvements over the successful ORB-SLAM2 [54], is considered the best of the graph-based VI-SLAM pipelines available to the community. In this section, we compare Kimera's performance to these pipelines with the latest improvements to Kimera-VIO. As ORB-SLAM3 is a SLAM-only pipeline, we only provide comparisons against ORB-SLAM3 with loop closures enabled in Kimera. Vins-Fusion can do either VIO alone or VIO with loop closures for a full SLAM pipeline, so we compare in both cases. Because Vins-Fusion cannot do RGB-D VIO, we omit results for the A1 dataset, which uses the D455 camera. This is denoted with blank space in that region of the table. Additionally, as there are no stereo cameras in the Car-Sim dataset, we show results for Kimera-VIO in monocular mode, and omit results for Vins-Fusion in stereo mode.

Table 4.6 compares Kimera-VIO (without loop closures) with Vins-Fusion [59]. Kimera is evaluated with external odometry for the Jackal and A1 datasets. and in monocular mode for the Car-Sim datasets. Kimera is also evaluated using the RGB-D frontend for the A1 dataset. All three pipelines are evaluated without loop closures, to show raw VIO tracking performance. Overall, Kimera outperformed Vins-Fusion in the majority of cases, with Vins-Fusion showing failures in several datasets (represented by dashes). The exception was in the uHumans2 datasets, where Vins-Fusion in stereo was better by a significant margin. It is not immediately clear why Vins-Fusion performs so well on these sequences, but it is possibly due to the fact that the agent in these simulations is a very idealized robot, and Vins-Fusion responds disproportionately well to those conditions. The agent doesn't have any dynamics that would cause disturbances in the IMU data, unlike in the Car-Sim datasets where car dynamics are simulated and there are frequent accelerations and braking maneuvers. Car-Sim and uHumans2 were both developed in the same simulation environment, just with different scenes and agent dynamics, making this discrepancy very interesting.

Table 4.7 compares Kimera-VIO to Vins-Fusion and ORB-SLAM3, all with loop closures. Vins-Fusion and ORB-SLAM3 are evaluated in monocular and stereo/RGB-D mode. Note that as ORB-SLAM3 supports RGB-D-Inetial VI-SLAM, we used that configuration for the A1 dataset, however ORB-SLAM3 was uanble to maintain consistent tracking. Kimera outperforms its competitors in most cases; Vins-Fusion has the lowest trajectory error in a few datasets while ORB-SLAM3 does not outperform any other pipeline in any dataset. The exception, again, was with the uHumans2 datasets, where Vins-Fusion performed the best.

4.4 Conclusions

In this chapter, we presented several key improvements to Kimera since its initial release in 2019. In particular, we discussed modifications to the Kimera-VIO frontend in Section 4.2.1 including the addition of other sensor-frontends (*e.g.,* monocular,

	VIO Absolute Translation Error RMSE (No Loop Closures)					
	Kimera-VIO		**Vins-Fusion Mono**		**Vins-Fusion Stereo**	
Dataset	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]
car_outdoor_sim_1	0.65	0.03	–	–		
car_outdoor_sim_2	1.82	0.09	–	–		
car_outdoor_sim_3	2.81	0.32	–	–		
car_outdoor_sim_4	3.24	0.22	–	–		
campus_hybrid_0	3.21	0.03	5.6	0.43	3.75	0.04
campus_hybrid_1	3.73	0.48	9.71	1.1	13.1	15.0
campus_hybrid_2	8.3	1.09	10.57	2.76	–	–
campus_hybrid_3	11.8	7.02	41.42	50.36	47.8	68.7
campus_indoor_0	11.5	1.19	6.96	1.7	7.7	1.67
campus_indoor_1	4.48	0.16	12.0	7.17	3.75	1.69
campus_indoor_2	10.7	2.45	–	–	–	–
campus_indoor_3	6.05	0.81	–	–	–	–
campus_outdoor_0	10.6	2.0	–	–	–	–
campus_outdoor_1	12.3	1.35	–	–	21.6	10.9
campus_outdoor_2	21.3	2.09	–	–	–	–
simmons_a1_0	0.92	0.64	–	–		
uHumans2_apartment	0.11	0.0	0.03	0.1	0.01	0.01
uHumans2_suburb	2.25	0.13	1.51	0.07	0.23	0.03
uHumans2_office	0.34	0.4	0.23	0.03	0.05	0.01
uHumans2_subway	4.11	2.78	0.28	0.01	0.16	0.01

Table 4.6: VIO localization accuracy for Kimera (with external odometry) compared to Vins-Fusion. No loop closures were used in any of the configurations here. Datasets that failed to maintain tracking are noted with dashes. The best result is highlighted in green for each dataset. Blank space denotes that either the pipeline was unable to run on that dataset (*e.g.,* no support for RGB-D) or the dataset did not contain relevant sensors (*e.g.,* Car-Sim does not have stereo cameras). For Car-Sim datasets, Kimera-VIO is evaluated in monocular mode using the right-facing camera. Vins-Fusion was evaluated in monocular mode and in stereo mode.

RGB-D), optional external odometry sources, image feature binning, and updated keyframe-selection logic. We also discussed changes to the backend, most notably the inclusion of `GNC` as an outlier-rejection method for robust pose-graph optimization. We provided extensive ablation studies on the effects of these improvements on localization error, across a variety of datasets. Additionally, we showcased improvements to the dense volumetric mapping of Kimera-Semantics with evaluations of Kimera-PGMO on a diverse set of single-robot datasets. Finally, we demonstrated Kimera's performance as compared to other open-source VI-SLAM pipelines on these datasets.

Dataset	VI-SLAM Absolute Translation Error RMSE (With Loop Closures)									
	Kimera-VIO		**Vins Mono**		**Vins Stereo**		**ORB Mono**		**ORB Stereo/D**	
	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]	Avg [m]	Std [m]
car_outdoor_sim_2	0.51	0.01	–	–			22.6	4.23		
car_outdoor_sim_3	2.55	0.52	–	–			61.2	20.5		
car_outdoor_sim_4	3.22	0.26	–	–			15.3	6.33		
campus_hybrid_0	3.21	0.04	5.12	1.37	3.85	0.05	25.0	21.6	–	–
campus_hybrid_1	3.73	0.39	6.15	5.39	10.1	14.8	44.2	9.08	–	–
campus_hybrid_2	8.3	1.09	4.47	1.39	–	–	–	–	–	–
campus_hybrid_3	11.8	7.02	42.4	52.4	48.1	69.3	–	–	–	–
campus_indoor_0	11.4	1.19	6.14	0.91	8.01	1.89	–	–	11.9	5.03
campus_indoor_1	3.98	0.83	11.8	8.08	3.54	1.61	16.5	6.07	5.86	5.64
campus_indoor_2	6.97	2.05	–	–	–	–	72.4	0.31	15.2	5.49
campus_indoor_3	6.04	0.81	–	–	–	–	–	–	23.3	14.9
campus_outdoor_0	10.6	1.99	–	–	20.3	17.3	–	–	19.8	24.8
campus_outdoor_1	12.3	1.35	–	–	4.53	3.46	88.9	10.8	15.7	9.38
campus_outdoor_2	21.3	2.09	5.94	2.97	25.3	–	–		18.1	17.3
simmons_a1_0	0.92	0.64	–	–		–	–		–	–
uHumans2_apartment	0.06	0.0	0.02	0.01	0.01	0.0	0.12	0.53	0.02	0.42
uHumans2_suburb	0.69	0.13	0.61	0.4	0.14	0.02	32.6	0.4	–	–
uHumans2_office	0.11	0.18	0.02	0.01	0.02	0.0	0.69	0.4	0.04	0.05
uHumans2_subway	0.41	2.78	0.02	0.04	0.01	0.0	0.72	0.32	0.16	0.21

Table 4.7: VI-SLAM localization accuracy for Kimera (with external odometry) compared to Vins-Fusion and ORB-SLAM3. Loop closures are included for all pipelines represnted here. Datasets that failed to maintain tracking are noted with dashes. Blank space denotes that either the pipeline was unable to run on that dataset (*e.g.*, no support for RGB-D) or the dataset did not contain relevant sensors (*e.g.*, Car-Sim does not have stereo cameras). For Car-Sim datasets, Kimera-VIO is evaluated in monocular mode using the right-facing camera. The best result is highlighted in green for each dataset.

Chapter 5

Kimera-Humans: A First Step Towards Dynamic Agent Tracking in 3D Scene Graphs

5.1 Introduction

Kimera is not just a VI-SLAM pipeline. In our previous work, we developed several modules on top of Kimera in an effort towards realizing high-level spatial perception via 3D Dynamic Scene Graphs (DSGs) [61, 62]. DSGs allow for scalable representations of the environment that are actionable at different levels; from the abstract objects level for high-level task planning to the metric base-layer for low-level obstacle avoidance, the DSG is a powerful perception tool. Figure 5-1 shows a visualization of a DSG from our previous work [62]. Each layer represents a hierarchical abstraction of the world; at the bottom we have the metric-semantic mesh as a full volumetric

The work in this chapter was done in collaboration with Arjun Gupta in [62], who mentions it in his thesis: dspace.mit.edu/handle/1721.1/127404. Arjun focused his work on the detection of humans using `GraphCMR`, while I focused on the tracking and data-association of humans over time. In addition, I was responsible in large part for the development of the simulation environment and datasets (uHumans1 and uHumans2) used in this chapter.

map, and as we go up the graph we have layers for objects and agents (humans), places and structures, rooms, and buildings.

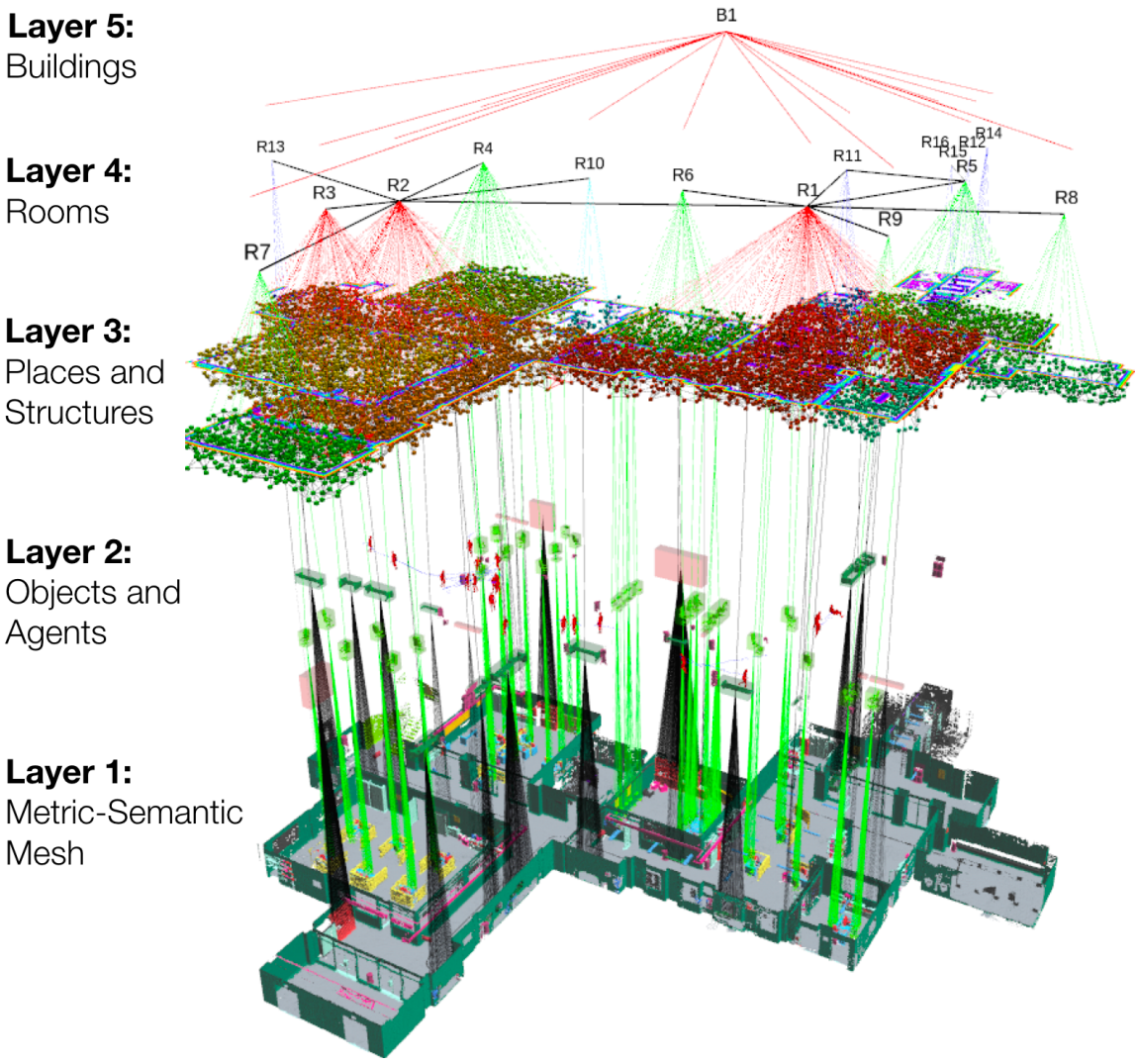


Figure 5-1: A 3D Dynamic Scene Graph (DSG) constructed by Kimera in the office scene of the uHumans2 dataset. The DSG is a hierarchical representation, with the dense 3D metric-semantic mesh at the lowest level. Further abstractions of this mesh are built up in the higher layers by Kimera; layer 2 contains all objects and agents (*e.g.,* robots, humans), enabling the user to efficiently model spatial relations between objects in the world. Layer 3 segments places and structures (*e.g.,* walls) on the 3rd layer, then rooms and buildings on layers 4 and 5 respectively. Figure courtesy of [62].

In order for our scene-graph to be "dynamic," we implemented agent tracking at the second layer of our scene-graph representation and focused on humans as the relevant dynamic agent. While in general there may be many classes of dynamic

entities in the environment, humans represent the most interesting for many robotic applications, especially where robots are expected to work symbiotically with humans in the real world. Humans are also challenging from a perception standpoint, due to our constantly changing morphology from the point-of-view of the robot: not only is our center of mass mobile, each of our limbs can move as well. A full breakdown of the layers of the DSG formulation is available in Rosinol *et al.* [62], for more information we refer the reader to that paper. For the purposes of this chapter, we highlight that in this formulation humans and robots are represented at Layer 2, and both have three attributes:

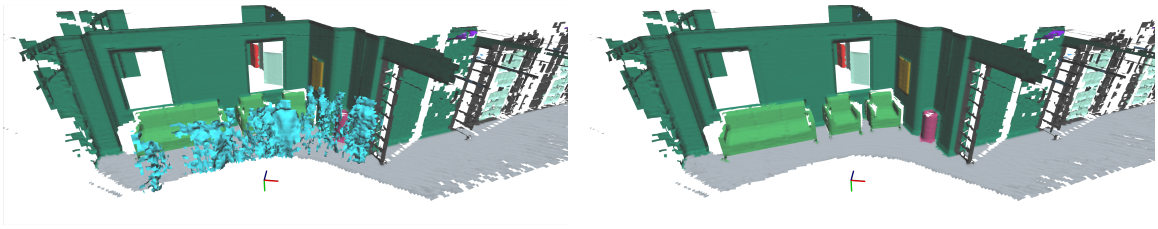
- a 3D pose graph describing their trajectory over time
- a mesh model describing their (non-rigid) shape
- a semantic class (*i.e.,* human, robot)

This shared framework between robots and humans enabled us to use the same tools we used for determining the ego-motion of the robot on humans in the scene. This was the central design choice of Kimera-Humans; by representing human motion through a pose-graph, one can leverage the powerful pose-graph optimization tools in Kimera-RPGO to efficiently and effectively model human motion. Any relevant factors could be included in the optimization; odometry factors to represent observed motion, loop closure factors to represent data association between observations, etc.

This chapter discusses the framework of Kimera-Humans as implemented in Rosinol *et al.* [62], and showcases an experimental evaluation on simulated datasets released with that paper. Section 5.2 discusses the design choices for the Kimera-Humans module. Section 5.3 presents an evaluation on the simulated data, and Section 5.4 concludes the chapter.

5.2 Kimera-Humans

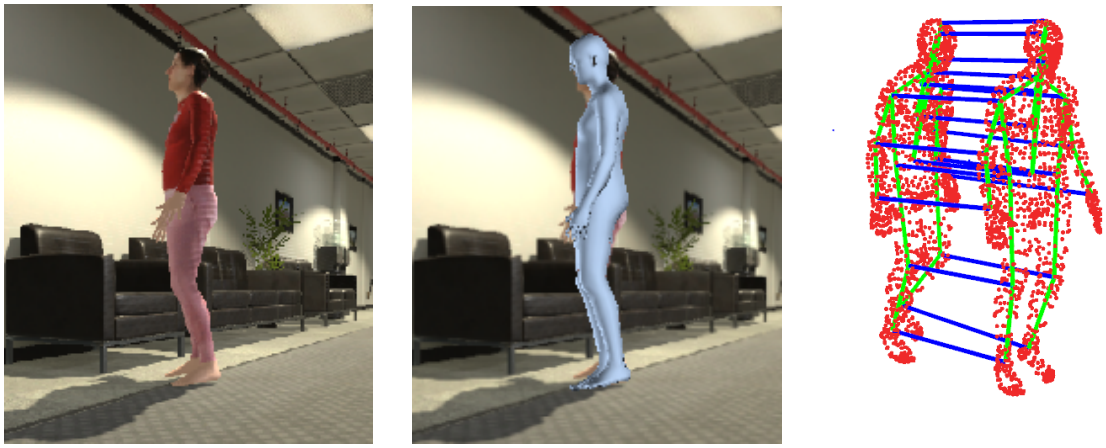
Kimera-Humans tracks a dense time-varying mesh model describing the shape of each human in the scene over time. Therefore, Kimera-Humans needs to detect and



(a) (b)

Figure 5-2: 3D mesh reconstruction without (5-2a) and with (5-2b) *dynamic masking*. Note that the human moves from right to left, while the robot with the camera rotates back and forth when mapping this scene. Figure courtesy of [62].

estimate the shape of a human in the camera images, and then track the human over time. Besides using them for tracking, we feed the human detections back to Kimera-Semantics, such that dynamic elements are not reconstructed in the 3D mesh of the environment. We achieve this by only using the free-space information when ray-casting the depth for pixels labeled as humans, an approach we dubbed *dynamic masking* (see results in Figure 5-2).



(a) (b) (c)

Figure 5-3: 5-3a: Input camera image from Unity, 5-3b: SMPL mesh detection and pose/shape estimation using [44], 5-3c: Temporal tracking and consistency checking on the maximum joint displacement between detections. Figure courtesy of [62].

For human shape and pose estimation, we use the Graph-CNN approach of Kolotouros *et al.* [44] (`GraphCMR`), which directly regresses the 3D location of the vertices of an SMPL [50] mesh model from a single image. An example mesh is shown in

figure 5-3.

Given a pixel-wise 2D segmentation of the image, we crop the left camera image to a bounding box around each detected human, which then becomes an input to `GraphCMR`. `GraphCMR` outputs a 3D SMPL mesh for the corresponding human, as well as camera parameters (x and y image position and a scale factor corresponding to a weak perspective camera model). We then use the camera model to project the human mesh vertices into the image frame. After obtaing the projection, we then compute the location and orientation of the full-mesh with respect to the camera using PnP [90] to optimize the camera pose based on the reprojection error of the mesh into the camera frame. The translation is recovered from the depth-image, which is used to get the approximate 3d position of the pelvis joint of the human in the image. Finally, we transform the mesh location to the global frame based on the world transformation output by Kimera-VIO.

Human Tracking and Monitoring. The above approach relies heavily on the accuracy of `GraphCMR` and discards useful temporal information about the human. In fact, `GraphCMR` outputs are unreliable in several scenarios, especially when the human is partially occluded. In this section, we describe our method for (i) maintaining persistent information about human trajectories, (ii) monitoring `GraphCMR` location and pose estimates to determine which estimates are inaccurate, and (iii) mitigating human location errors through pose graph optimization using motion priors. We achieve these results by maintaining a pose graph for each human the robot encounters and updating the pose graphs using simple but robust data association.

Pose Graph. To maintain persistent information about human location, we build a pose graph for each human where each node in the graph corresponds to the location of the pelvis of the human at a discrete time. Consecutive poses are connected by a factor [19] modeling a zero velocity prior on the human motion with a permissive noise model to allow for small motions. The location information from `GraphCMR` is modelled as a prior factor, providing the estimated global coordinates at each timestep. In addition to the pelvis locations, we maintain a persistent history of the SMPL parameters of the human as well as joint locations for pose analysis.

The advantage of the pose-graph system is two-fold. First, using a pose-graph for each human's trajectory allows for the application of pose-graph-optimization techniques to get a trajectory estimate that is smooth and robust to misdetections. Many of the detections from `GraphCMR` propagate to the pose-graph even if they are not immediately rejected by the consistency checks described in the next section. However, by using Kimera-RPGO and PCM outlier rejection, the pose-graphs of the humans can be regularly optimized to smooth the trajectory and remove bad detections. PCM outlier rejection is particularly good at removing detections that would require the human to move/rotate arbitrarily fast. Second, using pose graphs to model both the humans and the robot's global trajectory allows for unified visualization tools between the two use-cases. Figure 5-4 shows the pose graph (blue line) of a human in the office environment, as well as the detection associated with each pose in the graph (rainbow-like color-coded human mesh).

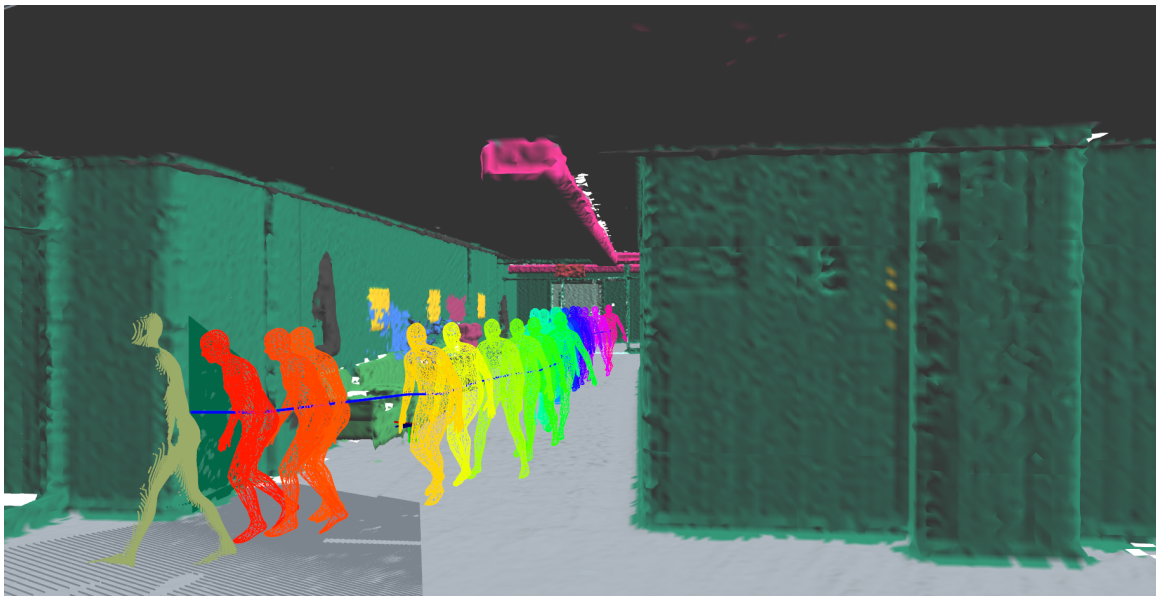


Figure 5-4: Optimized pose-graph (blue line) for a single human. The detected human shape is shown as a 3D mesh, color-coded from the most recent detection in red to the oldest one in pink. Figure courtesy of [62].

Data Association. A key issue in the process of building the pose graph is associating which nodes belong to the same human over time and then linking them

appropriately. We use a simplified data association model which associates a new node with the node that has the closest euclidean distance to it. This form of data association works well under the mild assumption that the distance a human moves between timesteps is smaller than the distance between humans.

We do not have information for when a human enters the frame and when they leave (although we do know the number of people in a given frame). To avoid associating new humans with the pose graphs of previous humans, we add a spatio-temporal consistency check before adding the pose to the human's pose graph, as discussed below.

To check consistency, we extract the human skeleton at time $t-1$ (from the pose graph) and t (from the current detection) and check that the motion of each joint (figure 5-3c) is physically plausible in that time interval (*i.e.,* we leverage the fact that the joint and torso motion cannot be arbitrarily fast). This check is visualized in figure 5-3c. We first ensure that the rate of centroid movement is plausible between the two sets of skeletons. Median human walking speed being about 1.25 m/s [64], we use a conservative 3m/s bound on the movement rate to threshold the feasibility for data association. In addition, we use a conservative bound of 3m on the maximum allowable joint displacement to bound irregular joint movements.

The data association check is made more robust by using the beta-parameters of the SMPL model [50], which encode the various shape attributes of the mesh in 8 floating-point parameters. These shape parameters include, for example, the width and height of different features of the human model. We check the current detection's beta parameters against those of the skeleton at time $t-1$ and ensure that the average of the difference between each pair of beta parameters does not exceed a certain threshold (0.1 in our experiments). This helps to differentiate humans from each other based on their appearance. In Kimera-Humans, the beta parameters are estimated by `GraphCMR` [44].

If the centroid movement and joint movement between the timesteps are within the bounds and the beta-parameter check passes, we add the new node to the pose graph that has the closest final node as described earlier. If no pose graph meets the

consistency criteria, we initialize a new pose graph with a single node corresponding to the current detection.

Node Error Monitoring and Mitigation. As mentioned earlier, `GraphCMR` outputs are very sensitive to occluded humans, and prediction quality is poor in those circumstances. To gain robustness to simple occlusions, we mark detections when the bounding box of the human approaches the boundary of the image or is too small (≤ 30 pixels in our tests) as incorrect. In addition, we use the size of the pose graph as a proxy to monitor the error of the nodes. When a pose graph has few nodes, it is highly likely that those nodes are erroneous. We determined through experimental results that pose graphs with fewer than 10 nodes tend to have extreme errors in human location. We mark those graphs as erroneous and remove them from the DSG, a process we refer to as pose-graph pruning. This is similar to removing short feature tracks in visual tracking. Finally, we mitigate node errors by running optimization over the pose graphs using the stationary motion priors and we see that we can achieve a great reduction in existing errors.

5.3 Experiments

In this section, we evaluate the performance of Kimera-Humans on several simulation environments. Sequences are taken from the uHumans1 and uHumans2 datasets, which were released to the public as part of our prior work on Kimera [62]. These datasets were recorded in the TESSE simulation environment, which was also open-sourced, and are referenced in Chapter 4 as well. As a part of this work, we developed a method for including humans in the simulation environment on-demand, with randomized or pre-planned trajectories in the navigable space of the scene. Using SMPL [50], we maintain realistic human motion and obtain ground-truth SMPL joint and beta (body shape) parameters for each human in the scene. Beta parameters are randomized (or specified fully) for each human in the scene, leading to more realistic conditions for data association. Ground truth SMPL parameters are only used to validate `GraphCMR`, not in-the-loop for Kimera-Humans. For more information on

these datasets, we refer the reader to web.mit.edu/sparklab/datasets/uHumans2.

Human Nodes. 5.1 shows the average localization error (mismatch between the pelvis estimated position and the ground truth) for each human on the uHumans1 datasets. Each column adds a feature of the proposed model that improves performance. The first column reports the error of the detections produced by [44] (label: "Single Image"). The second column reports the error for the case in which we filter out detections when the human is only partially visible in the camera image, or when the bounding box of the human is too small ($\leq$ 30 pixels, label: "Single Image filtered"). The third column reports errors with the proposed pose graph model discussed in 5.2 (label: "Pose-Graph track") and includes PCM outlier rejection and pose-graph pruning. The fourth column reports errors when the mesh feasibility check for data association is enabled (label: "Mesh Check"), and the fifth reports errors when the beta-parameter data-association technique is also enabled (label: "Beta Check"). The simulator's humans all have randomized beta parameters within a known range to better approximate the distribution of real human appearance.

The Graph-CNN approach [44] for SMPL detections tends to produce incorrect estimates when the human is occluded. Filtering out these detections improves the localization performance, but occlusions due to objects in the scene still result in significant errors. Adding the mesh-feasibility check decreases error by making data association more effective once detections are registered. The beta-parameter check also significantly decreases error, signifying that data association can be effectively done using SMPL body-parameter estimation.

Only the apartment scene did not follow the trend; results are best without any of the proposed techniques. These are outlier results; the apartment environment had many specular reflections that could have led to false detections.

5.4 Conclusions

In this chapter we presented Kimera-Humans, part of a prior work on expanding Kimera to support high-level spatial perception. Kimera-Humans is a module for

Dataset	Scene	#H	Localization Errors [m]				
			Humans				
			Single Image	Single Image Filtered	Pose Graph Track.	Mesh Check	Beta Check
uH1	Office	12	2.51	1.82	1.60	1.57	**1.52**
		24	2.54	2.03	1.80	1.67	**1.50**
		60	2.03	1.78	1.65	1.65	**1.63**
uH2	Office	0	–	–	–	–	–
		6	1.87	1.21	0.86	0.82	**0.63**
		12	2.00	1.43	1.16	1.05	**0.61**
	Suburb	0	–	–	–	–	–
		24	21.3	2.02	1.06	1.03	**0.74**
		36	14.0	2.50	1.44	1.14	**0.55**
	Subway	0	–	–	–	–	–
		24	8.34	6.56	5.53	5.31	**1.92**
		36	7.61	5.80	5.20	5.12	**2.83**
	Apartment	0	–	–	–	–	–
		1	**4.32**	4.79	5.38	5.64	6.43
		2	2.83	**2.52**	2.66	2.69	3.79

Table 5.1: Human localization errors in meters. A dash (–) indicates that the human is not present in the scene. '#H' column indicates the number of humans in the scene. 'uH1' and 'uH2' stand for the uHumans1 and uHumans2 datasets respectively. Table courtesy of [62].

tracking human agents in a scene, and integrating them into a 3D Dynamic Scene Graph. We are also able to remove dynamic humans from the metric mesh of the environment via *dynamic masking* in Kimera-Semantics, thereby cleaning the static scene graph for use with robotic planners and controllers. The shared representations between dynamic humans and dynamic robots is synergistic; by using the same pose-graph tools we use to obtain globally consistent trajectories for robots, we are able to localize humans in large-scale and small-scale environments with relatively little error. Kimera-Humans was an early step in the development of DSGs for perception and spatial AI; its release led to seminal works in the field such as Hydra [32] and Kimera-Multi [17, 73] where the idea of representing dynamic agents with the

same tools used for robot localization was extended to support multi-agent VI-SLAM. Nonetheless, there remain several avenues of further research with Kimera-Humans. In particular, methods to better apply the concepts behind loop closures to human tracking seem likely to yield results. The specificity of data assocation (recognizing individual humans) could be further improved through the use of additional descriptors assigned to humans (*e.g.,* information about clothing), and more complex motion models can be used to both identify unique individuals and build more accurate pose graphs of human motion.

Chapter 6

Conclusion and Future Work

Developing a robust VI-SLAM pipeline to support localization, mapping, semantic scene understanding, and dynamic tracking is a challenge, but a worthwhile one in the pursuit of autonomous robots. This thesis does not solve the problem entirely, but it marks a significant step in the right direction.

With the original release of Kimera, we were able to perform stereo-inertial VI-SLAM in real-time, with dense semantic mapping. In this thesis, we developed features to improve localization performance and increase accuracy in mapping. In Chapter 3 we extended Kimera to support multi-camera VI-SLAM, making strides in the specific application of autonomous valet parking on self-driving cars. Furthermore, we developed an efficient method for performing dense ground-plane mapping using all four monocular cameras around the car to obtain an accurate map of the free-space around the car using a modified version of Kimera-Semantics. Our system was evaluated on over 20 datasets collected in various autonomous parking contexts by The Ford Motor Company, and outperformed state-of-the-art VI-SLAM algorithms. We then further developed the open-source version of Kimera, as discussed in Chapter 4. Among these developments were improvements to Kimera-VIO to support better feature tracking, more efficient keyframe management, the use of external (wheel) odometry, and the use of graudated non-convexity [83, 6] for outlier rejection in pose-graph optimization. The systems described in chapters 3 and 4 show the robustness and adapability of the Kimera architecture, which can be flexed from

a drone, to a self-driving car, to a small ground vehicle, to a quadrapedal robot, to much more. Finally, as we expanded our VI-SLAM pipeline into the field of spatial perception and 3D Dynamic Scene Graphs (DSGs), we developed techniques for human tracking and discussed them in Chapter 5. This enabled Kimera to become more than just a VI-SLAM pipeline, but instead the foundation of a larger vision system capable of tackling the problem of high-level robotic perception.

However, spatial perception is far from solved; there are several directions for future work that are apparent from the efforts presented in this thesis. First, the findings in Chapter 3 show that the underlying VI-SLAM system has room for improvement; features like online sensor calibration and better support for other camera models (*e.g.,* that better support wide field-of-view cameras) can lead to lower localization error. Second, the nature of the dynamic human tracking formulation expressed in Chapter 5 gives rise to the idea of spatio-temporal scene graphs, a topic not well-covered in the existing literature. Current DSGs can mostly capture the motion of humans in the scene, but cannot deal with arbitrary moving objects. Adding time as a dimension in the formulation would permit online change detection, a tough challenge in its own right. Finally, by detecting and tracking changes in the scene graph over time, a robot or a team of robots could begin to predict the motion of dynamic objects, as well as identify causal explanations for changes observed in the world. All of these avenues for research require at their core a fast, accurate VI-SLAM system with semantic-mapping and dynamic tracking capabilities. For this reason, we hope that our improved version of Kimera can serve as the foundation for these perception engines of the future.

Bibliography

[1] Marcus Abate, Ariel Schwartz, Xue Iuan Wong, Wangdong Luo, Rotem Littman, Marc Klinger, Lars Kuhnert, Douglas Blue, and Luca Carlone. Multi-camera visual-inertial simultaneous localization and mapping for autonomous valet parking. 2023.

[2] Sameer Agarwal, Keir Mierle, et al. Ceres solver. 2012.

[3] P. Anderson, B. Fernando, M. Johnson, and S. Gould. Spice: Semantic propositional image caption evaluation. In *European Conf. on Computer Vision (ECCV)*, pages 382–398, 2016.

[4] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[5] M. Andriluka, S. Roth, and B. Schiele. Monocular 3D pose estimation and tracking by detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 623–630, 2010.

[6] P. Antonante, V. Tzoumas, H. Yang, and L. Carlone. Outlier-robust estimation: Hardness, minimally tuned algorithms, and applications. *IEEE Trans. Robotics*, 38(1):281–301, 2021. (pdf).

[7] I. Armeni, Z. He, J. Gwak, A. Zamir, M. Fischer, J. Malik, and S. Savarese. 3D scene graph: A structure for unified semantics, 3D space, and camera. In *Intl. Conf. on Computer Vision (ICCV)*, pages 5664–5673, 2019.

[8] Anurag Arnab, Carl Doersch, and Andrew Zisserman. Exploiting temporal context for 3D human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3395–3404, 2019.

[9] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3d human pose and shape from a single image. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conf. on Computer Vision (ECCV)*, 2016.

[10] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Lewis Bridgeman, Marco Volino, Jean-Yves Guillemaut, and Adrian Hilton. Multi-person 3D pose estimation and tracking in sports. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[12] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M.W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. *Intl. J. of Robotics Research*, 2016.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robotics*, 32(6):1309–1332, 2016. arxiv preprint: 1606.05830, (pdf).

[14] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Trans. Robotics*, 2021.

[15] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Machine Intell.*, 8(6):679–698, November 1986.

[16] Y. Chang, K. Ebadi, C. Denniston, M. Fadhil Ginting, A. Rosinol, A. Reinke, M. Palieri, J. Shi, Chatterjee A, B. Morrell, A. Agha-mohammadi, and L. Carlone. LAMP 2.0: A robust multi-robot SLAM system for operation in challenging large-scale underground environments. *IEEE Robotics and Automation Letters (RA-L)*, 7(4):9175–9182, 2022. (pdf).

[17] Y. Chang, Y. Tian, J.P. How, and L. Carlone. Kimera-Multi: a system for distributed multi-robot metric-semantic simultaneous localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021. arXiv preprint: 2011.04087, (pdf).

[18] W. Choi, Y. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 33–40, 2013.

[19] F. Dellaert and M. Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1-2):1–139, 2017.

[20] K. Ebadi, L. Bernreiter, H. Biggie, G. Catt, Y. Chang, A. Chatterjee, C.E. Denniston, S-P. Deschênes, K. Harlow, S. Khattak, L. Nogueira, M. Palieri, P. Petráček, P. Petrlík, A. Reinke, V. Krátký, S. Zhao, A. Agha-mohammadi, K. Alexis, C. Heckman, K. Khosoussi, N. Kottege, B. Morrell, M. Hutter, F. Pauling, F. Pomerleau, M. Saska, S. Scherer, R. Siegwart, J.L. Williams, and L. Carlone. Present and future of SLAM in extreme underground environments. *arXiv preprint: 2208.01787*, 2022. (pdf).

[21] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Mimc-vins: A versatile and resilient multi-imu multi-camera visual-inertial navigation system, 2020.

[22] Ahmed Elhayek, Carsten Stoll, Nils Hasler, Kwang In Kim, H-P Seidel, and Christian Theobalt. Spatio-temporal motion tracking with unsynchronized cameras. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1870–1877. IEEE, 2012.

[23] F. Dellaert et al. Georgia Tech Smoothing And Mapping (GTSAM). `https://gtsam.org/`, 2019.

[24] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems (RSS)*, 2015. accepted as oral presentation (acceptance rate 4%) (pdf) (video) (supplemental material: (pdf)).

[25] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. Robotics*, 33(1):1–21, 2017. arxiv preprint: 1512.02363, (pdf), technical report GT-IRIM-CP&R-2015-001.

[26] A. Fukui, D. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas, Nov 2016. Association for Computational Linguistics.

[27] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, October 2012.

[28] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: A research platform for visual-inertial estimation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[29] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[30] Yao He, Huai Yu, Wen Yang, and Sebastian A. Scherer. Toward efficient and robust multiple camera visual-inertial odometry. *ArXiv*, abs/2109.12030, 2021.

[31] S. Huang, S. Qi, Y. Zhu, X. Xiao, Y. Xu, and S. Zhu. Holistic 3D scene parsing and reconstruction from a single rgb image. In *European Conf. on Computer Vision (ECCV)*, pages 187–203, 2018.

[32] N. Hughes, Y. Chang, and L. Carlone. Hydra: a real-time spatial perception engine for 3D scene graph construction and optimization. In *Robotics: Science and Systems (RSS)*, 2022. (pdf).

[33] Joshua Jaekel. Towards robust multi camera visual inertial odometry. 2020.

[34] Jinwoo Jeon, Sungwook Jung, Eungchang Lee, Duckyu Choi, and Hyun Myung. Run your visual-inertial odometry on nvidia jetson: Benchmark tests on a micro aerial vehicle. 2021.

[35] C. Jiang, S. Qi, Y. Zhu, S. Huang, J. Lin, L. Yu, D. Terzopoulos, and S. Zhu. Configurable 3D scene synthesis and 2D image rendering with per-pixel ground truth using stochastic grammars. *Intl. J. of Computer Vision*, 126(9):920–941, 2018.

[36] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2901–2910, 2017.

[37] J. Johnson, R. Krishna, M. Stark, L. Li, D.A. Shamma, M.S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3668–3678, 2015.

[38] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[39] Ajinkya Khoche, Maciej K. Wozniak, Daniel Duberg, and Patric Jensfelt. Semantic 3d grid maps for autonomous driving. *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2681–2688, 2022.

[40] U. Kim, J. Park, T. Song, and J. Kim. 3-D scene graph: A sparse and semantic representation of physical environments for intelligent agents. *IEEE Trans. Cybern.*, PP:1–13, Aug. 2019.

[41] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020.

[42] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to Reconstruct 3D Human Pose and Shape via Model-fitting in the Loop. *arXiv e-prints*, page arXiv:1909.12828, Sep 2019.

[43] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2252–2261, 2019.

[44] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[45] J. Krause, J. Johnson, Ranjay R. Krishna, and L. Fei-Fei. A hierarchical approach for generating descriptive image paragraphs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3337–3345, 2017.

[46] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Visual Genome: Connecting language and vision using crowdsourced dense image annotations. *arXiv preprints arXiv:1602.07332*, 2016.

[47] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J. Black, and Peter V. Gehler. Unite the people: Closing the loop between 3D and 2D human representations. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[48] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. In *International Conference on Computer Vision (ICCV)*, 2017.

[49] X. Liang, L. Lee, and E. Xing. Deep variation structured reinforcement learning for visual relationship and attribute detection. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4408–4417, 2017.

[50] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[51] Cewu Lu, Ranjay Krishna, Michael Bernstein, and Fei-Fei Li. Visual relationship detection with language priors. In *European Conference on Computer Vision*, pages 852–869, 2016.

[52] J. G. Mangelson, D. Dominic, R. M. Eustice, and R. Vasudevan. Pairwise consistent measurement set maximization for robust multi-robot map merging. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 2916–2923, 2018.

[53] Anastasios I. Mourikis, Nikolas Trawny, Stergios I. Roumeliotis, Andrew E. Johnson, Adnan Ansar, and Larry Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280, 2009.

[54] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Trans. Robotics*, 33(5):1255–1262, 2017.

[55] Mohamed Omran, Christoph Lassner, Gerard Pons-Moll, Peter Gehler, and Bernt Schiele. Neural body fitting: Unifying deep learning and model based human pose and shape estimation. *Intl. Conf. on 3D Vision (3DV)*, pages 484–494, 2018.

[56] Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3d human pose and shape from a single color image. *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 459–468, 2018.

[57] Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv preprint: 1901.03642*, 2019.

[58] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[59] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint: 1901.03638*, 2019.

[60] A. Rosinol, M. Abate, Y. Chang, and L. Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020. arXiv preprint: 1910.02490, (video), (code), (pdf).

[61] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Robotics: Science and Systems (RSS)*, 2020. (pdf), (media), (video).

[62] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone. Kimera: from SLAM to spatial perception with 3D dynamic scene graphs. *Intl. J. of Robotics Research*, 40(12–14):1510–1546, 2021. arXiv preprint: 2101.06894, (pdf).

[63] D. Scaramuzza, A. Martinelli, and R. Siegwart. A toolbox for easy calibrating omnidirectional cameras. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[64] Michaela Schimpl, Carmel Moore, Christian Lederer, Anneke Neuhaus, Jennifer Sambrook, John Danesh, Willem Ouwehand, and Martin Daumer. Association between walking speed and age in healthy, free-living individuals using mobile accelerometer – a cross-sectional study. *PloS one*, 6(8):e23299, 2011.

[65] Xuan Shao, Ying Shen, Lin Zhang, Shengjie Zhao, Dandan Zhu, and Yicong Zhou. Slam for indoor parking: A comprehensive benchmark dataset and a tightly coupled semantic framework. *ACM Transactions on Multimedia Computing, Communications and Applications*, 19:1 – 23, 2022.

[66] Xuan Shao, Lin Zhang, Tianjun Zhang, Ying Shen, Hongyu Li, and Yicong Zhou. A tightly-coupled semantic slam system with visual, inertial and surround-view sensors for autonomous indoor parking. *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[67] Xuan Shao, Lin Zhang, Tianjun Zhang, Ying Shen, and Yicong Zhou. Mofisslam: A multi-object semantic slam system with front-view, inertial, and surround-view sensors for indoor parking. *IEEE Transactions on Circuits and Systems for Video Technology*, 32:4788–4803, 2022.

[68] Dinar Sharafutdinov, Mark Griguletskii, Pavel Kopanev, Mikhail Kurenkov, Gonzalo Ferrer, Aleksey Burkov, Aleksei Gonnochenko, and Dzmitry Tsetserukou. Comparison of modern open-source visual slam approaches. 2023.

[69] Shuyao Shi, Jiahe Cui, Zhehao Jiang, Zhenyu Yan, Guoliang Xing, Jianwei Niu, and Zhenchao Ouyang. Vips: real-time perception fusion for infrastructure-assisted autonomous driving. *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*, 2022.

[70] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.

[71] Vince Tan, Ignas Budvytis, and Roberto Cipolla. Indirect deep structured learning for 3D human body shape and pose prediction. In *British Machine Vision Conf. (BMVC)*, 2017.

[72] Graham W Taylor, Leonid Sigal, David J Fleet, and Geoffrey E Hinton. Dynamical binary latent variable models for 3d human pose tracking. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 631–638. IEEE, 2010.

[73] Y. Tian, Y. Chang, F. Herrera Arias, C. Nieto-Granda, J.P. How, and L. Carlone. Kimera-Multi: Robust, distributed, dense metric-semantic slam for multi-robot systems. *IEEE Trans. Robotics*, 2022. accepted, arXiv preprint: 2106.14386, (pdf).

[74] Nivedita Tripathi and Senthil Yogamani. Trained trajectory based automated parking system using visual slam on surround view cameras, 2020.

[75] Manchen Wang, Joseph Tighe, and Davide Modolo. Combining detection and tracking for human pose estimation in videos. *arXiv preprint arXiv:2003.13743*, 2020.

[76] R. Wang and X. Qian. *OpenSceneGraph 3.0: Beginner's Guide*. Packt Publishing, 2010.

[77] Yifu Wang, Kun Huang, Xin-Zhong Peng, Hongdong Li, and Laurent Kneip. Reliable frame-to-frame motion estimation for vehicle-mounted surround-view camera systems. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1660–1666, 2020.

[78] Felix Wimbauer, Nan Yang, Lukas von Stumberg, Niclas Zeller, and Daniel Cremers. Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. 2020.

[79] Xiru Wu, Feng Huang, Yaonan Wang, and Hui Jiang. A vins combined with dynamic object detection for autonomous driving vehicles. *IEEE Access*, 10:91127–91136, 2022.

[80] Zhenzhen Xiang, Anbo Bao, and Jianbo Su. Hybrid bird's-eye edge based semantic visual slam for automated valet parking. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11546–11552, 2021.

[81] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3106, 2017.

[82] Anqi Joyce Yang, Can Cui, Ioan Andrei Bârsan, Raquel Urtasun, and Shenlong Wang. Asynchronous multi-view slam. 2021.

[83] H. Yang, P. Antonante, V. Tzoumas, and L. Carlone. Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1127–1134, 2020. arXiv preprint:1909.08605 (with supplemental material), (pdf).

[84] H. Yang and L. Carlone. In perfect shape: Certifiably optimal 3D shape reconstruction from 2D landmarks. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. Arxiv version: 1911.11924, (pdf).

[85] Jingrui Yu, Zhen-Zhen Xiang, and Jianbo Su. Hierarchical multi-level information fusion for robust and consistent visual slam. *IEEE Transactions on Vehicular Technology*, 71:250–259, 2022.

[86] A. Zanfir, E. Marinoiu, and C. Sminchisescu. Monocular 3D pose and shape estimation of multiple people in natural scenes: The importance of multiple scene constraints. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2148–2157, 2018.

[87] H. Zhang, Z. Kyaw, S. Chang, and T. Chua. Visual translation embedding network for visual relation detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3107–3115, 2017.

[88] LinTong Zhang, David Wisth, Marco Camurri, and Maurice F. Fallon. Balancing the budget: Feature selection and tracking for multi-camera visual-inertial odometry. *IEEE Robotics and Automation Letters*, 7:1182–1189, 2021.

[89] Y. Zhao and S. Zhu. Scene parsing by integrating function, geometry and appearance models. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3119–3126, 2013.

[90] Yinqiang Zheng, Yubin Kuang, Shigeki Sugimoto, Kalle Astrom, and Masatoshi Okutomi. Revisiting the PnP problem: A fast, general and optimal solution. In *Intl. Conf. on Computer Vision (ICCV)*, pages 2344–2351, 2013.

[91] Xiaowei Zhou, Menglong Zhu, Georgios Pavlakos, Spyridon Leonardos, Konstantinos G Derpanis, and Kostas Daniilidis. MonoCap: Monocular human motion capture using a CNN coupled with a geometric prior. *IEEE Trans. Pattern Anal. Machine Intell.*, 41(4):901–914, 2018.

[92] Y. Zhu, O. Groth, M. Bernstein, and L. Fei-Fei. Visual7W: Grounded question answering in images. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4995–5004, 2016.