DISS. ETH NO. 27042

COLLABORATIVE VISUAL-INERTIAL STATE AND SCENE ESTIMATION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

MARCO KARRER

MSc ETH in Mechanical Engineering

born on October 18, 1988
citizen of Röschenz BL, Switzerland

accepted on the recommendation of

Prof. Dr. Margarita Chli, Examiner
Prof. Dr. Giorgio Grisetti, Co-examiner
Prof. Dr. Marc Pollefeys, Co-examiner

2020

Department of Mechanical and Process Engineering
ETH Zurich
Switzerland

Abstract

The capability of a robot to create a map of its workspace on the fly, while constantly updating it and continuously estimating its motion in it, constitutes one of the central research problems in mobile robotics and is referred to as Simultaneous Localization And Mapping (SLAM) in the literature. Relying solely on the sensor-suite onboard the robot, SLAM is a core building block in enabling the navigational autonomy necessary to facilitate the general use of mobile robots and has been the subject of booming research interest spanning over three decades. With the largest body of related literature addressing the challenge of single-agent SLAM, it is only very recently, with the relative maturity of this field that approaches tackling collaborative SLAM with multiple agents have started appearing. The potential of collaborative multi-agent SLAM is great; not only promising to boost the efficiency of robotic missions by splitting the task at hand to more agents but also to improve the overall robustness and accuracy by boosting the amount of data that each agent's estimation process has access to.

While SLAM can be performed using a variety of different sensors, this thesis is focused on the fusion of visual and inertial cues, as one of the most common combinations of sensing modalities in robotics today. The information richness captured by cameras, along with the high-frequency and metric information provided by Inertial Measurement Units (IMUs) in combination with the low weight and power consumption offered by a visual-inertial sensor suite render this setup ideal for a wide variety of applications and robotic platforms, in particular to resource-constrained platforms such as Unmanned Aerial Vehicles (UAVs). The majority of the state-of-the-art visual-inertial estimators are designed as odometry algorithms, providing only estimates consistent within a limited time-horizon. This lack in global consistency of estimates, however, poses a major hurdle in an effective fusion of data from multiple agents and the practical definition of a common reference frame, which is imperative before collaborative effort can be coordinated. In this spirit, this thesis investigates the potential of global optimization, based on a central access point (server) as a first approach, demonstrating global consistency using only monocular-inertial data. Fusing data from multiple agents, not only consistency can be maintained, but also the accuracy is shown to improve at times, revealing the great potential of collaborative SLAM. Aiming at improving the computational efficiency, in a second approach a more efficient system architecture is employed, allowing a more suitable distribution of the computational load amongst the agents and the server. Furthermore, the architecture implements a two-way communication enabling a tighter collaboration between the agents as they become capable of re-using information captured by other agents through communication with the server, enabling improvements of their onboard pose tracking online, during the mission.

In addition to general collaborative SLAM without specific assumptions on the agents' relative pose configuration, we investigate the potential of a configuration with two agents, carrying one camera each with overlapping fields of view, essentially forming a virtual stereo camera. With the ability of each robotic agent to move independently, the potential to control the stereo

baseline according to the scene depth is very promising, for example at high altitudes where all scene points are far away and, therefore, only provide weak constraints on the metric scale in a standard single-agent system. To this end, an approach to estimate the time-varying stereo transformation formed between two agents is proposed, by fusing the egomotion estimates of the individual agents along with the image measurements extracted from the view-overlap in a tightly coupled fashion. Taking this virtual stereo camera idea a step further, a novel collaboration framework is presented, utilizing the view-overlap along with relative distance measurements across the two agents (e.g. obtained via Ultra-Wide Band (UWB) modules), in order to successfully perform state estimation at high altitudes where state-of-the-art single-agent methods fail. In the interest of low-latency pose estimation, each agent holds its own estimate of the map, while consistency between the agents is achieved using a novel consensus-based sliding window bundle adjustment. Despite that in this work, experiments are shown in a two-agent setup, the proposed distributed bundle adjustment scheme holds great potential for scaling up to larger problems with multiple agents, due to the asynchronicity of the proposed estimation process and the high level of parallelism it permits.

The majority of the developed approaches in this thesis rely on sparse feature maps in order to allow for efficient and timely pose estimation, however, this translates to reduced awareness of the spatial structure of a robot's workspace, which can be insufficient for tasks requiring careful scene interaction and manipulation of objects. Equipping a typical visual-inertial sensor suite with an RGB-D camera, an add-on framework is presented that enables the efficient fusion of naturally noisy depth information into an accurate, local, dense map of the scene, providing sufficient information for an agent to plan contact with a surface.

With the focus on collaborative SLAM using visual-inertial data, the approaches and systems presented in this thesis contribute towards achieving collaborative Visual-Inertial SLAM (VI-SLAM) deployable in challenging real-world scenarios, where the participating agents' experiences get fused and processed at a central access point. On the other side, it is shown that taking advantage of specific configurations can push the collaboration amongst the agents towards achieving greater general robustness and accuracy of scene and egomotion estimates in scenarios, where state-of-the-art single-agent systems are otherwise unsuccessful, paving the way towards intelligent robot collaboration.

Zusammenfassung

Die Fähigkeit eines Roboters eine Umgebungskarte zu erstellen, diese kontinuierlich zu aktualisieren und seine Eigenbewegung innerhalb dieser Karte abzuschätzen bildet eine der zentralen Forschungsfragen in der mobilen Robotik und wird als Simultaneous Localization And Mapping (SLAM) bezeichnet. Da hierzu lediglich die im Roboter intergrierten Sensoren benötigt werden, stellt SLAM ein Kernbaustein zur Ermöglichung autonomer Navigation, und somit dem praktischen Einsatz von mobilen Robotern, dar und ist eines der florierenden Forschungsinteressen der letzten drei Dekaden. Während sich der Grossteil der Literatur mit SLAM für einzelne Agenten befasst, wurde erst vor kurzem damit begonnen die Zusammenarbeit mehrerer Agenten für SLAM zu untersuchen. Das Potential für kollaborativen Multi-Agenten SLAM ist gross, einerseits ermöglicht das Aufteilen einer Aufgabe auf mehrere Agenten eine Mission effizienter zu erledigen, andererseits kann aufgrund der umfassenderen Datenmenge auf welche jeder Agent Zugang hat die Robustheit und Genauigkeit gefördert werden.
Während SLAM prinizipiell mit einer Vielzahl von Sensoren möglich ist, lag der Fokus dieser Dissertation auf der Kombination von visuellen und inertialen Sensoren. Die Informationsreichhaltigkeit von Kameras zusammen mit der hochfrequenten, metrischen Information von Beschleunigungssensoren in Kombination mit dem tiefen Gewicht und dem geringen Energieverbrauch des Set-ups sind ideale Vorraussetzungen um dieses auf einer vielzahl von Robotern einzusetzen, im Speziellen auf Plattformen mit eingeschränkten Ressourcen wie zum Beispiel Drohnen. Ein Grossteil der visuell-inertialen Algorithmen kann lediglich lokal konsistente Ergebnisse erzielen. Die fehlende globale Kohärenz der Schätzung, erweist sich jedoch als problematisch bei der Definition eines globalen Bezugsystems, welches allerdings unerlässlich für die Koordination von mehreren Agenten ist. Aus diesem Grund untersucht diese Dissertation als einen ersten Ansatz das Potential von globaler Optimierung, basierend auf einem zentralen Zugangspunkt (Server) und zeigt mögliche Ansätze zur global konsistenten Trajektorienschätzung mittels visuell-inertialen Daten auf. Das Potential von kollaborativem-SLAM wurde mittels Zusammenführung der Daten mehrerer Agenten aufgezeigt. Die Robustheit der Schätzung wurde hierbei erhalten, während die Genauigkeit verbessert werden konnte. In einem zweiten Ansatz wurde eine effizientere Server Architektur gewählt, welche eine verbesserte Balance der erforderlichen Rechenarbeit zwischen Server und Agenten ermöglicht. Nebst einer Effizienzsteigerung können aufgrund der implementierten wechselseitigen Kommunikation ebenfalls Daten anderer Agenten zur verbesserten Schätzung verwendet werden, was zur erhöhten Genauigkeit der Positionsschätzungen der einzelnen Agenten im Verlauf einer Mission beiträgt. Zusätzlich zur generischen Betrachtung des kollaborativen-SLAM Problems, welches keine Annahmen bezüglich der relativen Positionierung der Agenten macht, wurde spezifisch die Konfiguration zweier Agenten, jeweils ausgestattet mit einer monokularen Kamera, betrachtet, bei welchen eine Ansichtsüberlappung existiert. Durch die Fähigkeit der Agenten sich unabhängig voneinander zu positionieren, kann die Stereo-Basisweite zwischen den beiden Agen-

ten dynamisch an die Szenentiefe angepasst werden. Dieser Ansatz erweist sich von grossem Nutzen in Flughöhen, in welchen Systeme, bestehend aus einzelnen Agenten, die metrische Skalierung der Trajektorie nur schlecht abschätzen können. Als erster Ansatz zur Schätzung der zeitlich variierenden Stereo-Transformation zwischen den zwei Agenten, wurden die individuellen Schätzungen der Eigenbewegungen unter Einbezug der Ansichstüberlappung zusammengeführt. Um diese Idee einer virtuellen Stereokamera noch einen Schritt weiter zu führen wurde ein neuartiges System vorgestellt, welches zusätzlich zu den Bildbeobachtungen aus den Ansichtsüberlappungen ebenfalls relative Entfernungsmessungen zwischen den Agenten in die Zustandsschätzung miteinbezieht. Hierdurch werden erfolgreiche Schätzungen in Flughöhen möglich, in welchen, nach dem aktuellen Stand der Technik, konventionelle Systeme mit einzelnen Agenten scheitern. Um eine geringe Latenz für die Positionsschätzung zu erzielen, errechnen beide Agenten ihre eigene Schätzungen der Umgebungskarte, während die Kohärenz der Schätzungen zwischen den beiden Agenten durch ein neuartiges konsensus-basiertes Bundle Adjustment erreicht wird. Aufgrund des hohen Grades an möglicher Parallelisierung und der asynchronen Natur des Algorithmus, ist das vorgeschlagene konsensus-basierte Bundle Adjustment in Bezug auf die Skalierbarkeit hinzu grösseren System äusserst vielversprechend.
Die Mehrheit der vorgestellten Algorithmen befasst sich mit der zentralen Frage einer effizienten Positionsschätzung anhand von spärlichen 3D Merkmalen. Für manche Tätigkeiten, wie beispielsweise das Interagieren mit der Umgebung, oder das Manipulieren von Objekten, erweist sich die Abwesenheit an Information zwischen den spärlichen 3D Merkmalen als problematisch. Um dies anzugehen, wurde ein Erweiterungsframework präsentiert, welches durch die Augmentierung eines typischen visuell-inertialen Systems durch eine RGB-D Kamera eine akkurate, lokale, dicht besetzte Umgebungskarte erstellen kann. Mit Hilfe dieser Erweiterung wird das Planen von Interaktionen mit der Umgebung ermöglicht.
In dieser Dissertation wurden verschiedene mögliche Ansätze zur zentralen Informationsverarbeitung von mehreren Agenten aufgezeigt. Der Fokus lag hierbei auf der Betrachtung von visuell-inertialen Systemen und hatte zum Ziel kollaborativen SLAM einen Schritt näher an realistische Anwendungsfälle heranzubringen. Andererseits wurde auch demonstriert wie die Nutzung spezifischer Konfiguration zur verbesserten Zusammenarbeit zwischen einzelnen Agenten beitragen kann. Dies führt unmittelbar dazu, dass robuste und akkurate Positionsschätzungen in Situation erzielt werden können, wo dies bis anhin mit vergleichbaren Algorithmen für einzelne Agenten nicht möglich war.

Acknowledgements

This doctoral thesis would not have been possible without the help and support of numerous people. First and foremost I would like to express my gratitude to Prof. Dr. Margarita Chli, director of the Vision for Robotics Lab at ETH Zurich, who granted me the opportunity to conduct my doctoral studies within her lab. Thank you for continuous support, the trust in my work and for sparking my inspiration. I am especially grateful for the freedom to bring in my own ideas in the process and for your honest and constructive guidance throughout the years. I would also like to thank Prof. Dr. Giorgio Grisetti and Prof. Dr. Marc Pollefeys for putting in their time and effort to co-examine this doctoral thesis. I highly appreciate the detailed feedback and the insightful discussions regarding my work.

I also want to thank the whole team at the Vision for Robotics Lab, who contributed to this thesis with many fruitful discussions as well as the help to conduct experiments numerous times. It was a great pleasure to work with such an open-minded and kind group of people Besides the remarkable friendly and pleasant work environment, I highly appreciate the good times we had together outside of work. I want to express special thanks to Dr. Lucas Teixeira for helping me out countless times with both hardware and software, to Ignacio Alzugaray for the frequent and open-minded spontaneous discussions over the years. Also special thank goes to Dr. Patrik Schmuck for the close collaboration and discussion on multi-agent research as well as the collaboration in preparing and carrying out various live demos of our systems. Furthermore, I would like to thank the team of the Autonomous Systems Lab as well as the Robotic Systems Lab at ETH Zurich. Thank you for your support and the valuable discussions over the years.

Lastly, I would like to express my deepest gratitude to my family and friends. I want to especially thank my mother, Brigitta, and my father, Philippe, for their unconditional love and for the continuous support they provided to me throughout all the years. Also thanks to my brother, David, for the good times and the fun we had together. A big thanks also goes to all my friends outside of academia for helping me to clear my head and keep a reasonable work-life balance. Special thank goes to Peter and Lukas with whom I could talk freely about anything and have a careless time. Further, I would like to thank my dear friend Seline for her continuous motivation and encouragement, and for transforming commuting to work into a pleasant experience.

September 10, 2020 *Marco Karrer*

Contents

Preface

Chapter 1 of this thesis introduces the central problems tackled in this work. As a cumulative doctoral thesis, Chapter 2 summarizes the context and the contributions of each paper comprised in this thesis. The interrelations between the papers are also detailed. Chapter 3 presents the overall achievements and suggestions of new ways to improve this work. All relevant papers forming the contributions of this thesis are attached at the end of the thesis.

List of Acronyms

ADMM Alternating Direction Method of Multipliers

AR Augmented Reality

CPU Central Process Unit

DoF Degrees of Freedom

EKF Extended Kalman Filter

FoV Field of View

GPS Global Positioning System

GPU Graphics Processing Unit

IMU Inertial Measurement Unit

KF Keyframe

LIDAR LIght Detection And Ranging

MP Map Point

MPC Model Predictive Control

MSCKF Multi-State Constraint Kalman Filter

RMSE Root Mean Squared Error

SaR Search and Rescue

SDF Signed Distance Function

SfM Structure from Motion

SLAM Simultaneous Localization And Mapping

ToF Time of Flight

UAV Unmanned Aerial Vehicle

UWB Ultra-Wide Band

VR Virtual Reality

VI Visual-Inertial

VI-SLAM Visual-Inertial SLAM

VIO Visual Inertial Odometry

Chapter 1

Introduction

The great potential of mobile robots to assist humans in a variety of tasks from healthcare to infrastructure inspection and maintenance, and post-disaster damage assessment has been a key driving force for research into the automation of robot missions over the past couple of decades. While tele-operated robots, for example remotely controlled small Unmanned Aerial Vehicles (UAVs), are already being deployed in applications such as the visual inspection of bridges or power lines, the use of tele-operation severely limits the applicability and the scalability of such approaches as the constant attention of expert operators is imperative.
In order to achieve the level of navigation autonomy that will enable the wider deployments of robots without expert surpervision, a robot needs to be able to constantly keep estimating its motion and its workspace in order to plan the actions required to complete the task at hand while avoiding any collisions. The use of Global Positioning System (GPS) measurements, which provide position estimates in a global reference frame offers tremendous help to robot position estimation and today, is widely employed, for example in commercial drones for photography. However, the quality of the GPS measurements drastically decays close to large structures such as tall buildings, and reception is completely unavailable indoors, which is the intended use-case for many robotic applications. Therefore, in order to enable reliable autonomous navigation of robots across different environments, it is key to equip them with the capability of estimating the map of their surroundings while estimating their ego-motion within this map in real time. This is commonly referred to as Simultaneous Localization And Mapping (SLAM) and it constitutes a core research problem in mobile robotics.
While early approaches to SLAM mainly utilized Sonar sensors [78, 79, 113], contemporary SLAM algorithms are dominated to a large extent by LIght Detection And Ranging (LIDAR) and vision-based techniques. As LIDAR sensors are capable to directly measure the scene depth of the sample points to distances of up to 300m, they are able to provide rich and accurate 3D information of the environment almost instantaneously. While LIDAR sensors available in the market have been getting less bulky than they used to be, their weight, size, cost and power consumption are still prohibiting their deployment on resource-constrained platforms such as

small UAVs, smartphones or Virtual Reality (VR) headsets. On the other hand, cameras are lightweight and low-power while providing rich information about the scene and due to their inexpensive nature, they are ubiquitous. The popularity of cameras has triggered big research interest in camera-based SLAM giving rise to a wealth of methods capable of accurately estimating the motion of a single camera in real-time [38, 44, 96]. However, due to the high information content of images and the resulting data volume as well as the physical limitations of cameras (e.g. its exposure depending on the illumination of the scene), the frame-rate at which vision data can be processed in real-time is limited, rendering vision-only SLAM effectively blind in between two camera frames. Incorporating Inertial Measurement Unit (IMU) measurements in addition to the visual data enables inference of the motion of the sensor-suite (and thus, the moving body carrying it) during these blind intervals between camera images. As a result, the fusion of visual and inertial cues improves the robustness of the estimation process against sudden motions and crucially, enables the estimation of the metric scale [10] of the trajectory, which is otherwise impossible with purely monocular SLAM, albeit imperative for autonomous navigation. Due to the low weight and power consumption, a sensor-suite combining inertial and visual data has quickly become one of the most popular choices for the motion estimation and control of UAVs. However, in contrast to vision-only approaches, where it is well-established to detect loops in the robot's trajectory enabling corrections for accumulated drift and map re-use ([96]), state of the art visual-inertial methods mostly perform Visual Inertial Odometry (VIO). So instead of running full SLAM, detecting and correcting for such loop closures, VIO perform mostly exploratory estimation essentially forgetting faster past experiences, so they inevitably result in unbounded drift. Even though in [98] the concept of map re-use and the associated benefits such as bounding of the estimation drift could be demonstrated, the introduced dependency on time caused by the IMU measurement results in the system's complexity to scale with time rather than with the spatial extent as it is the case in the visual-only case. With VI-ORB-SLAM [98] being the first and so far, the only real-time, complete Visual-Inertial SLAM (VI-SLAM) system in the literature (while it is also closed-source to date, restricting any possibility for testing on other data and benchmarking), effective visual-inertial fusion in a global map remains an open problem subject to research.

With increasing robustness and maturity of single-robot perception, extensions to multi-robot (multi-agent) systems have been increasingly attracting research attention as they promise a variety of different advantages over single-agent systems, such as the tolerance to robot failures and potential reduction of the execution time for a given mission. However, despite the growing interest in multi-agent applications, the level maturity of multi-agent vision-based SLAM systems is significantly lower than for the single-agent case. In multi-agent SLAM, often unrealistic assumptions are made, such as the instant availability of data from different agents, unrestricted communication bandwidth or known initial configurations. These severely limit the practical deployment of such systems. The potential, however, of performing SLAM with multiple agents and fusing their experiences live promises a multitude of potential advantages over single-agent SLAM. The most obvious advantage is the capability to cover larger areas of interest within a given time-frame as the space can be divided amongst the agents, which is especially valuable in time-critical applications such as in Search and Rescue (SaR), where a faster response can make the difference between life and death. Besides this promise for faster exploration and coverage, collaborative SLAM also holds the potential to improve the consis-

tency and accuracy of the scene and egomotion estimates not only by combining the estimates at the end of the mission but even more so during the mission itself. For example, considering loop-closures detected across the trajectories of different agents operating in the same area clearly reduces the overall drift in the estimation processes in comparison to considering the single agents' estimates alone. While this advantage is present also in multi-session or methods performing post-processing, a tight collaboration between the agents, where map-information from different agents is shared and re-used live, offers the possibility to improve the estimation quality and robustness of the individual agents online during a mission. Furthermore, when the agents are operating simultaneously additional measurements across the agents, such as visual observations of other agents during rendezvous or distance measurements, can be included adding further constraints on the SLAM graph and therefore boosting the accuracy and reliability of the estimation.

1.1 Motivation

The work conducted over the course of this thesis is motivated by a variety of different fields, but in particular by three main applications, namely aerial manipulation, robotic collaboration and Augmented Reality (AR)/VR. While these applications require a wide range of different capabilities in order to realize them, the core capability of scene perception and accurate localization within this scene is a common, key requirement across all of them. Additionally, these tasks require a timely and low-latency estimation on platforms with potentially limited computational resources, hence the algorithms should reflect this. In the remaining part of this section, the three applications will be introduced with their corresponding challenges and the potential impact.

Aerial Manipulation

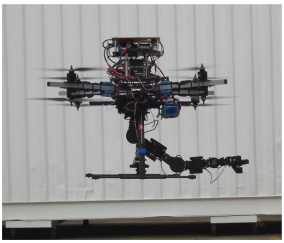


Figure 1.1: Robotic arm with a gripper [54]

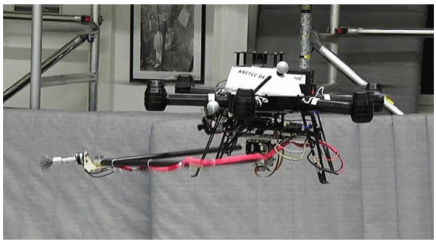


Figure 1.2: Delta manipulator with a grinding tool [61]

In industrial production factories robotic arms and manipulators are widely used to relief factory workers of tedious and dangerous tasks and in order to boost productivity. On the other hand, in other fields, such as inspection and maintenance of power or transportation infrastructure,

most tasks today are performed manually. However, with such environments most often posing limited and dangerous accessibility for humans, prohibitive amounts of security measures need to be undertaken in order to protect the workers from the potential safety hazards. Inspired by the need to eliminate such costly and cumbersome procedures, recent research has been aiming to equip UAVs with manipulators in order to allow the aerial robots to interact with the environment, as for example shown in Fig. 1.1, to perform tasks at places which are hard to reach otherwise. While for some tasks such as the cleaning of a surface as in Fig. 1.2 it is possible to perform the interaction blindly, for example by modeling the contact forces to get an idea of the contact [30] instead of employing vision-based perception, for other tasks such as certain non-destructive measurements with sensitive tools this could lead to the damage of the measurement equipment. In such cases having an accurate 3D model of the environment, where the contact should be performed can provide the necessary information to plan and execute the contact at the desired location and orientation. Furthermore, having a reconstruction of the environment available enables tasks where no direct contact is required but a defined distance and angle to the surface needs to be maintained, for example, optical testing methods [60].

Robotic Collaboration

Tasks requiring large areas to be covered as in applications, such as agriculture and SaR, using multiple robots can bring a significant speedup for doing so. Especially in SaR tasks, the time it takes to explore an unknown area and identify potential threats, such as fires, is crucial and needs to be held as low as possible, rendering multi-robot systems particularly impactful in such scenarios (e.g. [89]). Moreover, collaboration can enable robots to perform tasks, which a single robot is physically unable to achieve, for instance, the transportation of heavy or bulky objects, which cannot be lifted by a single robot [133, 140] as illustrated in Fig. 1.3. Other applications, as illustrated in Fig. 1.4 envision the use of multiple UAVs to quickly build up a mobile communication network that can be used for SaR teams to communicate between different units without the need to set up an infrastructure beforehand. However, at the core of every coordination effort between multiple robots is the requirement to know the relative positions of the other robots or their positions in a common reference frame, respectively. While external measurements, such as GPS sensing, trivially allow the establishment of such a common reference frame, as introduced before, the availability and accuracy of such measurements is not always given and therefore, can compromise the reliability and deployability of such an approach. Aside to overcoming such physical limitations, collaboration among multiple robots also has the potential to enable successful joint estimation in situations, where single-agent estimation fails or is severely degraded.

Figure 1.3: Collaborative transportation of an object [133]

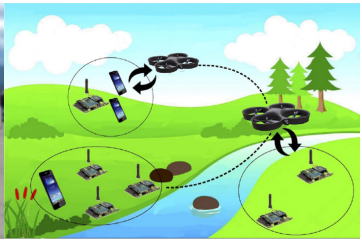


Figure 1.4: Employing UAVs as wireless relays [40]

Augmented and Virtual Reality

Besides applications in mobile robotics VI-SLAM is a driving factor behind the development of VR and AR applications and products. As humans have shown to be extremely sensitive to any inaccuracies in the state-estimation, e.g. resulting in motion sickness, large extents of research work and engineering have been dedicated to the development of smooth and artifact-free state-estimation, specifically to be employed on AR/VR devices. As the maturity of the devices and applications grew with time and first products are already commercially available, the industry's interest in multi-user applications has also been growing. The use-cases for multi-user AR/VR, ranging all the way from multi-player games to VR-aided inspection and virtual lectures on human anatomy, are extremely diverse and promising. However, in order for the user to have a truly immersive experience and to be able to recognize other users or to see the actions they perform on the virtual objects, the AR/VR goggles (and hand controls often coming with the set [2]) must be able to accurately localize themselves within a common map and transmit the information to other participants in a timely manner. One of the first commercially available systems enabling multi-user experiences ([1]), is based on creating spatial anchors, which are user-selected points of interest that other devices are capable to recognize and localize against in order to display content. Although such an approach is sufficient to share 3D content, it does not leverage the full potential of collaborative SLAM in the map-building and co-localization process in real-time. However, with modern AR/VR goggles being typically equipped with onboard cameras, computation units and network communication capabilities, they have all necessary hardware to perform collaborative SLAM, potentially boosting the effectiveness of multi-user applications.

1.2 Background

Global Visual-Inertial Mapping

As introduced before, the combination of visual information from a camera along with the information obtained from an IMU build a very effective and compact sensor setup to perform SLAM. While early approaches were dominated by Extended Kalman Filter (EKF) [110, 130]

and it's variant the Multi-State Constraint Kalman Filter (MSCKF) [85, 95] based approaches, with the development of efficient and real-time capable non-linear solvers [6, 76] along with the on-manifold IMU-preintegration technique [42], keyframe-based sliding window batch approaches to the VI-SLAM problem using non-linear least squares [82, 83, 139] started to become standard. Even though further progress led to improved performance and robustness for these approaches, their inherent design choice of limiting the keyframes entering the optimization at any time using a sliding window to achieve a constant computational complexity, inevitably results in a lack of global consistency of the map and the robot's trajectory. While there exist offline Structure from Motion (SfM)-like frameworks achieving globally consistent maps from visual-inertial data (e.g. [122]), however, due to their exhaustive run-time, these approaches are mainly limited to re-localization tasks [12, 87]. However, the capability of obtaining globally consistent maps within a reasonable time-frame is a crucial building block for collaborative SLAM in order to ensure consistency when fusing different agents' experiences. Due to the incremental nature of SLAM algorithms and the absence of corrections from global measurements, accumulated errors in the SLAM graph can only be corrected by detecting loops in the graph. For vision-based systems, such loops can be detected using the technique of place recognition. With the development of efficient and fast methods for place recognition [25, 26, 51], the incorporation of loop-closure detection in real-time systems has become tangible. Carefully designing a system using state-of-the-art components, the work in [96] was the first to present a complete monocular SLAM system with online global corrections and map-reuse. Using a survival of the fittest approach and removal of redundant keyframes, this work ([96]) was able to achieve a bounded complexity for a given area. With only a handful of works on online global visual-inertial SLAM/mapping available, one approach to this problem is the combination of a VIO with a pose graph back-end for the global optimization as proposed in [67] and later on in [111]. However, while this allows for relatively fast optimization of the poses even for larger problems, the resulting accuracy of the estimate is limited, depending highly on the quality of the VIO, while systematic errors remain largely unaddressed, such as persistent errors in the scale of the estimates. Inspired by these shortcomings and similarly to [98], in *Paper II* (i.e. [64], one of the contributions of this thesis) we explored the potential of global optimization using full bundle adjustment along with on-manifold IMU-preintegration [42]. Additionally, *Paper II* investigated the fusion of visual-inertial data from two agents into a global map in order to evaluate the potential of this method in improving the accuracy and the consistency of the estimates. The obtained results demonstrated that the chosen optimization approach achieves both a high level of accuracy while retaining consistent estimates also in the multi-agent case, where metric constraints are absent across the agents. However, due to the absence of parallelization in the implementation, in the multi-agent case real-time estimation could not be achieved. While some works are aiming at reducing the complexity of the SLAM graph by summarizing nodes [15, 74] or removing redundant nodes [121], these methods are tailored to a pose graph optimization structure and result in a significant amount of bookkeeping when applied to VI-SLAM ([15, 74]), or the resulting runtime prohibits the application for real-time SLAM ([121]). On the other hand, the recent work in [136] achieves a significant reduction of the optimization complexity by introducing a non-linear factor to summarize the IMU measurements in gravity direction and relative pose constraints.

Collaborative SLAM

When it comes to collaborative SLAM, two main architectural paradigms are present in the literature: centralized and distributed. While centralized approaches employ a central entity (i.e. a server) connected to all participants coordinating the data management across all participating agents, in distributed approaches, the agents exchange information directly with their peers (usually their neighbors) and perform all the handling of the information (i.e. exchange and fusion) individually and onboard, eliminating the need for any central entity to orchestrate data sharing and management. As a result, conceptually, distributed systems offer the advantage that data does not need to pass through a single point in the system and that the computational load is distributed across all participants, therefore distributed collaborative architecture is expected to scale better with the number of participants. However, due to the absence of a central coordinator to maintain data consistency, for example by avoiding double-counting of information, is extremely challenging leading to complex strategies for data sharing and fusion [28]. Many works focus on optimizing a specific aspect of the multi-agent system for a distributed setup, such as place recognition [22], robustness [146], or efficient data exchange [21]. Other approaches propose to use distributed optimization to perform pose graph optimization [18] in a collaborative fashion and for example to use semantic information to establish connections between the agents [20].
On the other hand, due to the simpler architecture and data handling, centralized systems are more mature in the literature. Utilizing the central server unit as an additional computation resource, the approaches proposed in [93, 114], reduce the computational load for the agents to a minimum by performing all the computationally expensive tasks, such as mapping, map-merging and global optimization on the server. However, such a strong reliance on the server reduces the applicability of the systems and the autonomy of the agents significantly, as any loss of connections to the server leads to a failure on the agent. The system architecture presented in [119], overcomes this limitation by still performing the most crucial operations on the agents, outsourcing only the heavy computations, such as map-merging and global optimization to the server. Building on top of this architecture, the work in [120] demonstrated collaborative SLAM with multiple agents on live data. However, this approach was designed for collaboration amongst agents running monocular odometry, rendering it prone to limitations, such as the scale ambiguity and the sensitivity to abrupt motions. As with single-agent systems, the inclusion of IMU measurements can be used to overcome these issues. In *Paper III* (i.e. [66]), the global optimization techniques used in *Paper II* were transferred into the server system of [120], while the odometry system on the agents' side was completely re-designed to include IMU-measurements, enabling the successful demonstration of it in real-time performance with up to four agents, using a standard laptop as a server.

Peer to peer collaboration

Observing the same scene from multiple view-points is a very powerful concept as, provided that the transformation between these viewpoints is known, the perception of the depth of the scene becomes possible, in the same way as in multi-view stereo reconstruction [49]. The most basic multi-view configuration is given by two views with overlap, the stereo setup. While the concept of stereo cameras has been used extensively in the literature both for SLAM [39, 97]

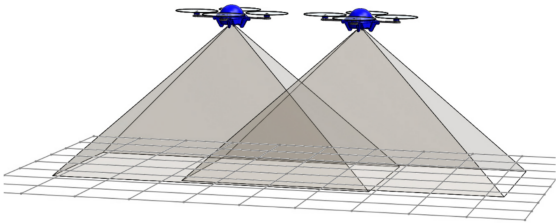


Figure 1.5: Illustration taken from [5], showing the idea of using two UAVs, equipped both with monocular cameras to form a variable baseline stereo camera.

as well as reconstruction [106, 132] and obstacle avoidance [52, 103] , it is also clear that the choice of the baseline of the stereo setup is a crucial parameter and to a large extent, determines the properties of the setup. As analyzed in [50], different baselines and/or image resolutions lead to different uncertainties and possible occlusion of objects. The work in [5], first proposed the idea of using two UAVs with monocular cameras onboard as a virtual stereo camera with a variable baseline as illustrated in Fig. 1.5, estimating the stereo transformation using the IMU measurements and the view-overlap across the monocular images captured by the two UAVs. While the observability of the estimation processes was successfully demonstrated, real applicability was left an open question as all visual measurements had been entirely simulated using ground-truth. Inspired by the potential of such a setup, instead of using the IMU measurements of the agents directly, in *Paper IV* ([63]), we proposed to use each agent's own VIO estimates and fuse them with the image observations obtained through the view overlap across the agents in a tightly coupled approach, which was successfully demonstrated on the EuRoC dataset [11]. More recently, a similar approach in [137] showed the benefits of fusing the data from such a virtual stereo setup in order to increase the accuracy achieved by the individual agents. However, for scenes which mainly contain objects far away from the camera, such as in high altitude UAV flights, these approaches start to become unreliable, due to the fact the single-agent estimation becomes unreliable, thus indirectly, having a big effect on the collaborative estimate as well. One possible fix for this is to use stereo SLAM for each agent's motion estimation while maximizing the stereo baseline. In [57, 58] this was achieved for a fixed-wing airplane by placing the cameras at the tips of the wings estimating the stereo baseline actively to correct for the wing deformations during flight. While this approach is suitable for fixed-wing airplanes, which generally have a large wingspan and therefore, allow large stereo baselines, in other cases, such as small multicopters, this approach is not suited as the maximum stereo baseline is rather limited. In *Paper V* (submitted), this limitation was addressed by building on the variable baseline stereo idea using two UAVs. However, in addition to fusing the agents' visual and IMU data, relative distance measurements, for example as obtained using Ultra-Wide Band (UWB) modules onboard each agent, are fused as well in order to keep the scale well constrained even at high altitudes. Utilizing a distributed optimization scheme, this system was shown to be capable of reliable collaborative SLAM estimation with network delays, without introducing additional latency in the pose estimation.

Scene Estimation

Figure 1.6: Illustration of the same scene using a sparse representation (left) and a dense representation of the environment (right)

Knowing the 3D structure of the environment is crucial before a robot can autonomously navigate through it without collisions or perform any type of autonomous manipulation (e.g. grasping of an object or touching a surface at a certain angle). Accurate localization and pose estimation within a sparse map is well established, however, the lack of information in between the sparse points hinders safe motion or contact planning significantly. In Fig. 1.6, a comparison between a sparse and a dense map of the same scene is illustrated. Furthermore, accurate 3D knowledge of the environment is a crucial element for AR in order to enable realistic blending of the rendered objects with the environment. While dense reconstruction from image data has been subject to research for quite some time and some impressive results [123] already appear in the literature, one of the first real-time capable dense visual SLAM methods was proposed in [100]. However, with the emergence of Time of Flight (ToF) and structured light sensors, offering noisy, but dense scene measurements, a new type of algorithms, labeled RGB-D SLAM started to emerge, essentially starting with the seminal work of KinectFusion [99] demonstrating real-time dense reconstruction of small workspaces for the first time. As KinectFusion was designed for small workspaces, several extensions addressing the spatial scalability were proposed [142, 145] later on, as well as methods for loop-closure detection and corrections for dense frameworks [144]. While the results of these systems are impressive, in order to achieve this performance, heavy Graphics Processing Unit (GPU) computation is required prohibiting the deployment of such approaches on resource-constrained platforms, such as small UAVs. Leaving out the pose tracking, [127] proposed a multi-scale dense mapping approach based on the octrees [91] capable of fusing an RGB-D datastream in real-time on a Central Process Unit (CPU). While [127] used ground-truth poses, in *Paper I* ([65]) the mapping approach was transformed into a local method and evaluated using VIO. Even though, both visual-inertial [11] as well as RGB-D [131] datasets are largely available and common, a combination of both was lacking from the literature at the time, triggering the need for a new dataset, which was presented in *Paper I*, containing both information along with scene- and pose ground-truth for benchmarking, but also allowing the evaluation of the proposed algorithm using real sensor input. Later on, with the development of Voxblox ([104]), a new framework for dense mapping specifically tailored for path-planning was proposed, which was extended to a submap-based approach allowing to include pose corrections after loop-closures [92]. Recently, the submap-

ping approach in [92] was extended using the submaps to create loop-closure constraints in a global pose graph to enable CPU only, global consistent dense mapping [112].

Chapter 2

Contribution

This chapter summarizes the core contributions of the research conducted during this doctoral thesis. In Section 2.1 for every publication, the overall context relating the work to the state of the art in Section 1.2 is provided. For the individual works, a summary of the research contributions as well as the interrelations between the publications are provided. Section 2.2 provides a list of all student projects supervised in the course of the doctoral studies.

2.1 Research Contribution

Paper I: Real-time Dense Surface Reconstruction for Aerial Manipulation

M. Karrer, M. Kamel, R. Siegwart, and M. Chli. Real-time Dense Surface Reconstruction for Aerial Manipulation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1601–1608, 2016

Context

As motivated in Section 1.1, performing manipulation tasks using lightweight manipulators borne by small UAVs offers appealing advantages, both from an economical as well as from a work-safety perspective. Driven by this, research has been conducted to investigate how to control UAVs equipped with manipulators [69, 73], but also how to design manipulators to maximize the accessible space while minimizing the moving mass [61]. While interaction with the environment is possible without structural knowledge of the environment using Model Predictive Control (MPC) and force feedback [30], the availability of accurate 3D information of the environment opens up the possibility to plan and optimize the physical interaction with a structure. Using RGB-D cameras or 2D ToFs sensors, detailed but noisy 3D scene information can be obtained, leading to a variety of dense SLAM and reconstruction methods. Starting from

Figure 2.1: Dense reconstruction using an UAV computed onboard in real-time.

the seminal KinectFusion [99] designed for small workspaces, several variations [37, 143, 145] exist, even systems capable of closing loops in a dense scene representation [144]. However, all of these methods heavily rely on GPU computation to process a large amount of information in real-time, rendering the methods unsuitable to run onboard a UAV with limited computational resources. Using the efficient Octree data structure proposed by [126], the framework in [127] demonstrated that it is possible to fuse the RGB-D data captured from known camera poses in real-time using only CPU computation. Using the framework of [127] as a basis, the work in *Paper I* aims at investigating the reconstruction accuracy that can be achieved using VIO for estimating the camera poses on-the-fly, as opposed to acquiring them from an external tracking system, for example. Closing the gap between RGB-D [131] and high-quality visual-inertial datasets [11] existing at the time, *Paper I* proposed a new dataset containing not only ground-truth camera poses but also accurate scene ground truth enabling quantitative evaluation of the reconstruction accuracy on real-data for the first time.

Contribution

Building on the assumption that VIO typically runs onboard a small UAV for frame-to-frame egomotion estimation and that any interaction of a robotic manipulator with the environment is to be performed within the Field of View (FoV) of the onboard sensors, this paper proposes to utilize a CPU-capable dense reconstruction framework. To this end, besides running the VIO, the UAV is assumed to be equipped with a depth camera (i.e. experiments are presented with both an RGB-D and a ToF camera). As VIO is generally prone to drift, an adjustable time-horizon Δt was introduced, which controls the removal of possibly outdated parts of the reconstruction. This permits the viewpoint of the sensor to fluctuate slightly, e.g. due to wind gusts, without loosing the gathered information about the scene structure, while resetting any volumetric blocks of the dense scene representation when they do not receive an update within Δt in the past. This helps to avoid artifacts arising from drift in the pose estimation when revisiting parts of the scene following longer exploratory motions. In order to allow the incorporation of uncertainty information in the depth measurements, a computationally lightweight weighting scheme was introduced, which resulted in an average decrease of the reconstruction error of 10% on real data using a ToF camera. Due to the lack of real-world datasets containing

high-quality visual-inertial data, depth data as well as pose and scene ground-truth, a dedicated dataset was created and made publicly available to the community [1]. The dataset consists of two different scenes and contains hand-held sequences with a sensor-rig equipped with a Visual-Inertial sensor [101] for high-quality visual-inertial data and either the Intel RealSense R200 or the CamBoard pico flexx from PMD for depth perception (only one depth sensor was mounted at a time to avoid inferences between them). The scene ground truth was captured using a high precision laser measuring system along with the pose ground truth from a motion capturing system, thus enabling quantitative evaluation of the algorithm on real data.

Interrelations

Although the method presented in *Paper I* is developed on a single-agent basis, as the required inputs are the sensor poses in six Degrees of Freedom (DoF) along with a noisy point cloud to represent the scene, the approach can be employed as an add-on to the approaches in *Paper III* and *Paper V*, which produce six-DoF poses at frame-rate. Furthermore, the approaches in *Paper IV* and *Paper V* both estimate a variable-stereo configuration, which could be utilized to compute a disparity image to serve as input for this work.

Paper II: Towards Globally Consistent Visual-Inertial Collaborative SLAM

M. Karrer and M. Chli. Towards Globally Consistent Visual-Inertial Collaborative SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3685–3692, 2018

Context

While visual-inertial methods performing odometry have reached a considerable maturity, full SLAM systems closing loops, performing global corrections and re-using previously mapped information are mainly limited to vision-only systems (either monocular or stereo) [96, 97]. The first approach demonstrating the re-use of mapped information in a monocular-inertial setup was introduced by Mur-Artal et al., dubbed as VI-ORBSLAM [98]. Their work shed light into some of the biggest issues in the full VI-SLAM, namely the proper estimation of accelerometer biases, their initialization and the scalability of the approach. Moreover, while collaborative frameworks performing purely visual SLAM exist [43, 119, 149], at the time there was a complete lack of collaborative approaches fusing visual and inertial measurements in the estimation. Inspired by the potential of such systems, *Paper II* employed a nominal VIO system with minimal modifications to investigate visual-inertial global mapping for one and two agents achieving close to real-time performance.

[1] https://v4rl.ethz.ch/research/datasets-code/V4RL-Dense-Reconstruction-Dataset.html

Contribution

This paper proposes a framework to combine information from potentially multiple agents into a single SLAM map; assuming that each agent runs nominal keyframe-based VIO, keyframe information gets communicated to a central server, where correspondences across keyframes are established and optimization is performed to fuse this data into a global map, contributing towards creating globally consistent SLAM maps from multiple agents. In contrast to [98], the current state of the IMU bias is consulted both for determining a successful initialization as well as for the decision whether a keyframe can be removed in order to increase the computational scalability of the system. While only considering the back-end part of the system, results on the EuRoC benchmarking dataset [11] demonstrates improvements on the trajectory error of at times over 50% over the state of the art, indicating the potential of the approach. Conducting experiments with two agents, in this work, we could demonstrate that even in the absence of inertial constraints between the agents, a consistent estimate of their trajectories and the scale in the estimates can be maintained and at times, even improving the accuracy compared to the single-agent trajectories.

Interrelations

This paper aims at investigating how to obtain a globally consistent trajectory and scene map using data from VIO running onboard one or multiple agents. In the evaluation, considerable improvements over the state of the art could be demonstrated on the simpler datasets, however, the returns were diminishing with more difficult sequences. Furthermore, the sequential nature of the implementation along with the exhaustive local optimization of the incoming keyframes rendered the approach far from real-time capable for more than two agents. In *Paper III*, these shortcomings are addressed, first, by introducing a tailored front-end odometry on each agent eliminating the need for the local optimization to be carried out by the server and second, by implementing parallelized data handling on the server.

Paper III: CVI-SLAM – Collaborative Visual-Inertial SLAM

M. Karrer*, P. Schmuck*, and M. Chli. CVI-SLAM - Collaborative Visual-Inertial SLAM, *IEEE Robotics and Automation Letters*, 3(4):2762-2769, 2018

* indicates equal contribution

Context

While in *Paper II* the centralized, consistent fusion of visual-inertial data obtained from two agents could be demonstrated, the serialized implementation and the exhaustive local optimization prohibits the real-time operation of the approach for more than one agent. Even though this limitation could be relaxed by increasing the level of parallelism in the implementation, as suggested in [43], the lack of information flowing from the server back to the agents limits the level of collaboration of the approach to collaborative mapping. There are a few works effectively addressing the use of two-way communication schemes to allow the agents the re-use of information. For example the RGB-D based system in [114] proposes to completely outsource

all operations except for the tracking to a server, limiting the autonomy of the agents, as any loss of connection inevitably leads to a failure of the system. To the contrary, the work of [33] proposes to run the complete SLAM pipeline on each agent, including loop-closure detection and global optimization, while additionally exchanging data with a central server. While such an approach ensures the autonomy of the agents even with an unreliable connection to the server, it misses out on computational savings of outsourcing heavy, but non-time-critical computations (such as global optimization) to the server. As a result, this system's applicability is limited to agents permitting such heavy computation onboard. An architectural middle-way was proposed in [119, 120], where the agents keep a map of limited horizon providing them with estimation autonomy even in the absence of contact to the server. The server accumulates the data from all agents and fuses them, where possible, via loop closure detection and global optimization. By communicating back to the agents, the server allows the sharing and re-use of information from all participating agents whose maps can be fused together if their trajectories meet. However, the system [120] is designed to work only as a monocular system, therefore suffering from the typical shortcomings such as sensitivity to abrupt motions and the absence of an estimate of the metric scale. As it is well known, the metric scale can be obtained by fusing IMU measurements [10], therefore, transitioning the system from a monocular to a visual-inertial would bring the system closer to real-world robotic applicability. However, such a transition is not trivial and requires significant modifications in both the back-end optimization on the server-side as well as a complete re-design of the odometry front-end.

Contribution

Building on top of the system architecture of [120], both the employed front-end odometry as well as the optimization algorithms were re-designed in order to incorporate the IMU information. In both the agent's front-end odometry as well as for the global collaborative estimate, the well established on-manifold IMU pre-integration scheme [42] is implemented, building a complete collaborative monocular-inertial SLAM system. To the best of the authors' knowledge, CVI-SLAM[2] is the first collaborative SLAM system using visual-inertial data along with two-way communication between the participating agents and the server, bringing collaborative SLAM a step closer to real-world applicability. The experimental evaluation presented in *Paper III* demonstrates that CVI-SLAM performs comparably to state-of-the-art visual-inertial SLAM systems, while outperforming them in a collaborative setup. Additionally, in *Paper III* the effect of the two-way communication is demonstrated as the resulting accuracy of the frame-wise pose tracking could be improved online by using the collaborative scheme to share information. Even though as compared to [120], also IMU data needs to be communicated to the server, the improved robustness of the visual-inertial front-end allows the use of fewer visual features, leading to decreased overall network traffic.
Implementing the whole system from scratch, the contributions claimed in this thesis are mainly focused on the design and the implementation of the optimization related algorithms running on the agents as well as the server, leading to a re-usable and extendable library for SLAM related optimization. Furthermore, besides the incorporation of the IMU information into the optimization, the tight integration into the front-end were implemented as result of this thesis.

[2] Collaborative Visual-Inertial SLAM

Interrelations

Paper III combines the findings from *Paper II* with the architecture of the monocular collaborative SLAM system [120]. Using the efficient communication architecture and the server-sided map reconstruction approach of [120] along with the global optimization presented in *Paper II*, real-time performance with multiple agents is possible. On the agent side, a local map optimization inspired by the local optimization used in *Paper II* is implemented, however, as it is run on each agent, compared to *Paper II* the computational load of this action is distributed, sparing the server from additional computational load. While simple heuristics were employed to remove redundant keyframes from the system in order to improve the system's scalability, the computational bottleneck of the system is the global bundle adjustment. Depending on the available computational architecture, e.g. on a cloud-computer, the scalability of the global bundle adjustment could be boosted by employing the distributed optimization method presented in *Paper V* in order to parallelize the optimization.

Paper IV: Collaborative 6DoF Relative Pose Estimation for Two UAVs with Overlapping Fields of View

M. Karrer, M. Agarwal, M. Kamel, R. Siegwart, and M. Chli. Collaborative 6DoF Relative Pose Estimation for Two UAVs with Overlapping Fields of View. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6687–6693, 2018

Context

By the means of triangulation on pairs of images captured with a stereo camera, the scene depth can be estimated at a particular time step rendering this as a powerful tool for SLAM and scene reconstruction. However, the choice of the stereo baseline is crucial for the effective range of a stereo camera, as too big baselines lead to problems with occlusions or lack of sufficient overlap for triangulation, while too small baselines produce a high uncertainty in the depth estimation. Having the possibility to adjust to a suitable baseline on the fly, based on the depth and accuracy requirements of the task at hand holds a big potential [50]. In a loose sense, such a variable stereo setup can be achieved by using two robots equipped each with a monocular camera, which by arranging themselves in a certain way are able to adjust the stereo baseline. This idea was proposed in [5], where the authors showed that using two UAVs both equipped with cameras and IMUs it is possible to compute the relative transformation between the cameras in metric scale. Furthermore, knowing the relative transformation between two robots allows them to be controlled to perform certain configurations for example in order to transport an object as shown in Fig. 1.3. In *Paper IV*, a lightweight approach to estimate the relative transformation between two cameras by fusing odometry information along with vision measurements is presented.

Contribution

Picking up on the idea in [5], a lightweight estimation framework was developed in *Paper IV*, capable to estimate the relative transformation between two robots based on their odometry

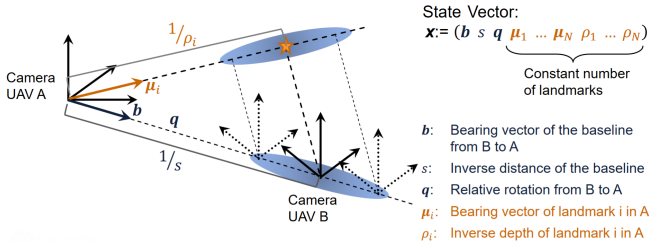


Figure 2.2: The proposed parameterization of the stereo baseline formed by the two UAVs carrying one monocular camera each, defining a state vector containing a bearing vector with inverse distance scaling. This parameterization captures the underlying uncertainty (indicated with the light blue ellipses) of the estimation problem from pairwise image measurements of scene landmarks (here, denoted with a star).

information and view-overlap. In *Paper IV*, an EKF was chosen to fuse the robots' odometry estimates along with their monocular vision measurements. The inclusion of a small number of 3D landmarks in the filter state allowed to employ a guided correspondence search resulting in an average processing frame-rate of 74Hz, while the overall generated network traffic is below 0.5MB/s. In order to capture the underlying uncertainty of the relative transformation between the robots, in *Paper IV* it is proposed to parameterize the transformation as a standard rotation along with a bearing vector with an inverse-depth parameter for the baseline as shown in Fig. 2.2.

Interrelations

Paper IV presents the basic idea that was later on extended in *Paper V* to a fully distributed system. Although the main contribution of *Paper IV*, namely the proposed parameterization of the relative stereo transformation, was not further employed, the realization that a system based on using individual agents' odometries can only be as robust as these odometries considered in isolation, lead to the development of the system in *Paper V*. For example, considering the parametrization illustrated in Fig. 2.2, it becomes clear that the use of a distance measurement between the agents significantly reduces the uncertainty of the system, which is specifically addressed in *Paper V*.

Paper V: Distributed Variable-Baseline Stereo SLAM from two UAVs

M. Karrer and M. Chli. Distributed Variable-Baseline Stereo SLAM from two UAVs, Submitted to *IEEE Transactions on Robotics*, 2020

Context

Nowadays VIO and VI-SLAM approaches show remarkable performance and robustness in various scenarios from hand-held devices, indoors to UAVs employed in large-scale disastrous areas. Furthermore, as shown in *Paper II* and *Paper III*, fusing data captured and processed from multiple robotic agents can improve the overall quality of the estimates when fused in a comprehensive, collaborative manner. However, any failures of the agents' local odometries have a devastating effect and rendering the affected agents unaware of both the other agents and their ego-motion. On the other hand, as demonstrated impressively in [149], for highly dynamic scenes, a tight collaboration between agents can enable to overcome situations in collaboration where a single agent would fail to obtain a reasonable estimate. One of the known failure cases for VI-SLAM, for example, is when all scene points are far away from the agent as it is the case for UAVs flying at high altitudes [57, 58]. Using the idea of a variable baseline stereo setup employing two UAVs, each equipped with a monocular camera to form a stereo camera as in *Paper IV*. However, in contrast to the approach in *Paper IV*, which relies on a functioning odometry for both agents individually, in *Paper V* a joint estimator is proposed, which fed with relative distance measurements between the agents as they can be obtained via UWB modules, is capable of producing a reliable, metrically scaled pose estimate for both agents.

Contribution

In *Paper V*, a novel sensor fusion framework fusing relative distance information (e.g. as obtained using UWB modules) between two UAVs together with the visual-inertial information from the two UAVs. The framework enables to reliably estimate the ego-motion of both agents at absolute scale even at high altitudes in real-time. The frame-wise tracking performed on each agent utilizes an EKF fusing the inertial data together with the visual observations of the estimated map points, allowing to have a low latency pose-estimation latency. In order to fuse the relative distance measurements, in *Paper V* a sliding-window bundle adjustment with a continuous-time Z-spline based trajectory representation is proposed. To allow each agent to hold its own estimate of the map such that their autonomy of estimates is not compromised by potential network delays while maintaining consistency across the agents, it is proposed to employ an asynchronous, consensus-based optimization scheme on the basis of [107] for the first time to the best of the author's knowledge. Even though in *Paper V*, the proposed optimization scheme is mainly employed to obtain a consistent estimate between the two agents, the approach has significant potential in allowing scaling up global bundle adjustment for a large number of agents as the problem can be approached in a decentralized fashion enabling a high degree of parallelism. Using the estimated poses, a high-level formation controller was developed, allowing to automatically adjust the virtual stereo-baseline between the two agents based on the estimated scene requirements. In the thorough evaluation using photo-realistic simulations, the advantage of the proposed approach over state-of-the-art fixed stereo-inertial estimators at higher altitudes could be demonstrated.

Interrelations

The approach presented in *Paper V* complements the approaches in *Papers II-IV*, as it specifically addresses some shortcomings of the single-agent VIO employed in these papers. The proposed distributed optimization approach in *Paper V*, holds large potential to scale up the bundle adjustment problem to a larger amount of data, as the overall problem can be split into smaller subproblems allowing to increase the parallelization of the computations. Such a parallelization could be used to address the computational bottlenecks in *Paper II* and *Paper III*. The proposed in lightweight localization EKF approach in *Paper V*, can potentially be employed for the frame-wise tracking in *Paper III* for the employment on computationally weaker hardware, such as smartphones.

2.2 List of Supervised Students

Master Theses

Master students, 6-month projects, full-time

- Agarwal, Mohit (Fall 2016): "Collaborative Visual-Inertial based Sensing using Multiple UAVs"
- Lampart, Andrea (Fall 2016): "Monochromatic SLAM augmented with an Event-Based Camera"
- Li, Kailai (Fall 2016): "Towards globally consistent,dense 3D reconstruction onboard a UAV"
- Müller, Fabian (Spring 2017): "Event-based pose tracking using patterns learnt during runtime"
- Bartolomei, Luca (Fall 2017): "3D Radiation Mapping using a small UAV"
- Thurnhofer, Franz (Fall 2017): "Large-scale Dense Scene Reconstruction from Multiple UAV"
- Böröndy, Aron (Spring 2018): "Visual-Inertial SLAM for consumer level sensors"
- Herbst, Constantin (Spring 2018): "Event-Based Visual Odometry Using Line Features"
- Hug, David (Spring 2018): "Event-based SLAM in Continuous Time"
- Russi, Mario (Spring 2018): "Collaborative State Estimation for High Altitudes using two UAVs"
- Schmidig, David (Spring 2018): "Visual SLAM for Dynamic Environments"
- Greter Rafael (Fall 2018): "Visual Tracking for UAVs"

- Strässle, Timo (Fall 2018): "Towards Collaborative SLAM using Smartphones"
- Perraudin, Jonathan (Fall 2019): "Towards Collaborative SLAM using Smartphones"
- Zuidema, Christoph (Fall 2019): "Autonomous Radiation Mapping using multiple UAVs"
- Pensotti, Sara (Spring 2020): "Autonomous Multi-Robot Exploration of Partially Unknown Environments"
- Ziegler, Thomas (Spring 2020): "Distributed UAV Formation Estimation via Pairwise Distance Measurements"

Semester Theses

Master students, 3-4-month projects, part-time

- Ziegler, Andreas (Fall 2016): "Map Fusion for Collaborative UAV SLAM"
- Schönbächler, Lukas (Spring 2017): "ROS Toolbox for visual sensor calibration"
- Ziegler, Thomas (Spring 2018): "High Accuracy Visual Inertial SLAM for Autonomous Navigation of small UAVs"
- Schaller, Sebastian (Fall 2018): "Feature Tracking for Event-based Cameras"
- Ginting, Fadhil (Spring 2019): "UAVs Formation Estimation using UWB Measurements"
- Ritz, Kamil (Spring 2019): "Rolling Shutter and Time Offset Compensation for Visual-Inertial-Odometry"
- Roth, Timo (Spring 2019): "Active Vision for Relative Pose Estimation"
- Sempertegui, Emilk (Spring 2019): "Optimization-based Path Planning for Autonomous Robot Navigation"
- Tearle, Ben (Spring 2019): "Collaborative UAV SLAM"
- Oberson, Lucien (Fall 2019): "Spline-Based Direct IMU Integration for SLAM"
- Schönbächler, Sacha (Fall 2019): "Efficient Map Re-Use for Collaborative SLAM"
- Li, Cliff (Fall 2019): "Towards Collaborative SLAM – Robustifying Visual-Inertial Odometry"
- Davide, Mambelli (Spring 2020): "Multi-agent path planning for active sensings"
- Yunfan, Gao (Spring 2020): "Multi-Sensor Fusion in Continuous Time"
- Bayerle, Michael (Spring 2020): "IMU-Handling for Smartphone-based Visual-Inertial Odometry"

Bachelor Theses and Studies on Mechatronics

Bachelor students, 3-month projects, full-time

- Scherrer, Beat (Spring 2018): "Monocular Simultaneous Localization and Mapping (SLAM) for a drone without IMU"
- Briner, Caspar (Fall 2018): "Autonomous Object Grasping"
- Brunner, Maurice (Spring 2020): "Vision and LIDAR based SLAM for mobile robots"

Chapter 3

Conclusion and Outlook

Driven by the potential impact of collaboration for state estimation and scene perception, in this doctoral thesis a set of contributions addressing various aspects of collaborative perception are presented. The focus was set on visual-inertial approaches, as this setup is ubiquitous and can be deployed on almost any platform. The work in this thesis investigated dense scene reconstruction (*Paper I*), global visual-inertial mapping and collaborative VI-SLAM (*Paper II*, *Paper III*) as well as peer-to-peer visual-inertial collaboration (*Paper IV*, *Paper V*). Even though the conducted research introduced novel concepts and pushed the state of the art in various ways, there are several open questions remaining that need to be addressed before the developed concepts can be widely employed in practice. In this chapter, a brief summary of the conducted research, recapitulating the main insights and contributions resulting from this thesis as well as a discussion of the challenges and potential future work in this field are provided.

With the goal of achieving global consistent maps and trajectories from monocular-inertial data, a central data fusion backend was developed in *Paper II*. In order to do so, the off-the-shelf keyframe based VIO method [83] was slightly adapted in order to enable the communication of keyframe data to re-use it in the backend. Our first attempts using vision-based bundle adjustment in combination with relative pose constraints across subsequent keyframes, aiming to maintain the metric scale, lead to poor scale estimates, which were traced back to the large local scale fluctuation of the VIO. Instead, with the combination of optimization of a local window of keyframes along with global full bundle adjustment, both incorporating the IMU information utilizing on-manifold pre-integration [42], we demonstrated that high accuracy can be achieved both on the trajectory as well as on the metric scale. Considering also the use of keyframe data from a second agent operating in the same area in this framework, despite the absence of IMU-constraints across the agents, the consistency of the joint estimate was shown to be maintained while the accuracy of the joint trajectory at times improved compared to the single-agent estimates when considered in isolation from each other. However, the sequential implementation of the approach as well as the exhaustive local optimization implies that the algorithm presented

in *Paper II* could not achieve real-time performance with more than one agent by design.
While the local optimization in *Paper II* is necessary for creating a global map using the nominal VIO employed in that work, an alternative frontend that is better tailored to potentially collaborating agents promises to reduce the required operations. Addressing this, the server-agent architecture proposed in [119, 120] was utilized in CVI-SLAM (*Paper III*) targeting to improve the scalability over the system in *Paper II*. In the CVI-SLAM paradigm, the global graph is constructed on the server using the communicated keyframe and map point data from each participating agent, avoiding local optimization. This is made possible as the VIO frontend is performing the local optimization onboard the agent, therefore, distributing the computational load. While the overall system architecture of CVI-SLAM, in comparison to the approach in *Paper II*, scales better with the number of agents, even though a heuristic-based redundancy detection was incorporated, ultimately the complexity of both methods grows unbounded over time. In contrast to purely vision-based approaches, the detection and removal of redundant keyframes in the global map is limited by time constraints, related to the IMU-measurements in order to retain a well-defined scale estimate, thus the SLAM graph is bound to grow continuously. A promising approach, significantly reducing the number of states in the global optimization, thus improving the scalability, was recently presented in [136], proposing to recover a non-linear factor from the VIO frontend, essentially building up constraints on the keyframes' roll- and pitch angles as well as relative constraints across the temporal neighbors, allowing to drop the IMU-related states (linear velocity, IMU-biases) in the global optimization. With the IMU information summarized in a relative constraint only acting across poses, dropping keyframes as done for example in [97] or [15] could be a valid option to bound the complexity of the problem.
The agent-server architecture in CVI-SLAM permits a two-way communication enabling re-using data from the server on each agent and vice-versa. In the experimental evaluation, this was demonstrated to improve the accuracy of the VIO on the agent during runtime. To the best of the author's knowledge, this is the first visual-inertial system in the literature exhibiting such a high degree of collaboration among the participating agents at runtime. While the data re-use from other agents reduces the overall error, the occurrence of small fluctuations of the estimate may increase, which is especially problematic when deploying the system in a closed control-loop, e.g. to control a UAV. Hence, further investigation of the method employed for data re-use in the agent's estimator holds a great potential to boost the applicability of CVI-SLAM. A possible direction could be the maintenance of an estimate of the state uncertainty on the agent, allowing to perform a model-based update with the information from the server, which was demonstrated in [87] to improve the smoothness of the pose tracking significantly.

Stereo cameras have become a widespread and common tool for both scene reconstruction as well as for SLAM, for example for aiding autonomous navigation for autonomous car and in mobile robotics in general. While being handy for lots of tasks, the useful range for stereo cameras is essentially determined by their baseline. Building upon the idea of using two UAVs equipped with a monocular camera each to form a variable-baseline stereo camera [5], in *Paper IV*, a new take on the problem setup was proposed. In contrast to [5], where IMU measurements along with a simulated vision pipeline were used to perform the experiments, in *Paper IV*, the onboard VIOs were fused with the vision measurements from the view-overlap in a tightly coupled fashion, allowing the system to be evaluated on the EuRoC dataset. Using the standard

6DoF transformation parameterization of the baseline consisting of a rotation and a Euclidean 3D vector, a tendency towards underestimating the baseline's covariance could be observed. The proposed parameterization of the baseline as a bearing vector with an inverse-depth parameter allowed to capture the underlying uncertainty of the problem more naturally as illustrated in Fig. 2.2, improving the consistency of the estimate. However, in order for the metrically scaled baseline to be estimated correctly, the VIO system used onboard each agent must be able to properly estimate the metric scale itself, which is not necessarily true, as the case for example at scenes with large depth (e.g. high-altitude flights with a downward-looking camera).
Filling this gap, in *Paper V* a novel, tightly collaborating estimation scheme was proposed, which besides fusing visual and inertial data captured by two different agents (e.g. UAVs) also fuses relative distance measurements across the agents (e.g. obtained via UWB modules). By employing a distributed approach, the system is designed such that both agents hold their own local estimate of the system's state, thus network delay does not result in additional latency in the pose estimation. Consistency across the two agents' estimates of the trajectories is achieved using an asynchronous consensus-based distributed optimization scheme, which, to the best of the author's knowledge is employed for the first time to solve sliding-window bundle adjustment. While in *Paper V*, the main purpose of the distributed optimization scheme is to achieve consistency across the agents, the potential of the approach lies especially in larger problems as these can be split into overlapping subproblems. Introducing constraints across these subproblems allows them to be solved in parallel while utilizing the consensus protocol still arriving at a global solution, similar to [147], however, without the need for a central coordination unit. The carefully designed system could be demonstrated to outperform state-of-the-art stereo-inertial algorithms with a stereo baseline of 0.22m at high altitudes of 20m upwards and maintain a high-quality metric scale even at 150m height in simulation. While real-world experiments achieving the desired parameters are extremely challenging, some first experiments inside a flying room have been performed successfully and the resulting estimate was utilized to stabilize the UAVs. These tests revealed that the biggest difference between the simulation and the real world is the quality and quantity of the data association across the employed BRISK [81] features. Nevertheless, a thorough analysis of the performance in real settings is subject to future work and in particular, the data association needs to be put to the test, for example investigating different suitable feature types or consider some form of image normalization to improve the matching across the two agents. In a next big step, a generalization of the approach towards a larger fleet and loosening the stereo requirement to be used only if available, for example during rendezvous, but not as a requirement for the system to be operative would significantly boost the applicability of the approach. In combination with a corresponding path planning algorithm, this could be employed to actively improve the quality of the estimation through collaboration.

In this thesis, dense scene reconstruction was investigated as an add-on task to the actual visual-inertial robot localization and its egomotion estimation. The developed approach in *Paper I* is based on the CPU-only method presented in [127], which was extended with a decay window in order to perform only a local reconstruction avoiding inconsistencies in the model due to drift in the VIO's estimate. Furthermore, a weighting scheme was introduced in order to reduce the trust in uncertain depth measurements and vice versa. As at the time of publication, there was a lack of suitable RGB-D datasets in the literature containing also high-quality visual-inertial along with a high-quality pose and scene ground truth data to evaluate the accuracy of

the developed approach, a corresponding dataset was created and made publicly available. Despite the achievable accuracy of the approach with measured errors on the reconstruction of below 10mm, the approach itself is rather limited, as it either requires ground-truth poses or the extent of the local reconstruction to be approximately restricted to lie within to the FoV of the depth sensor. While other methods such as for example [104], suffer from the same problem, an extension of [104] is able to correct a map globally by creating submaps anchored to poses in a pose graph [92]. The recent work [112], building upon [92] even utilizes the submaps to establish loop-closure constraints. In combination with, for example, CVI-SLAM and a suitable way of communicating submaps, the approach presented in [112] could build a powerful basis enabling collaborative, global dense mapping. With such a perception framework, the level of autonomy achievable with multi-robot systems could reach a whole new level.

Throughout all the work conducted in this thesis, the sensor setup used or simulated has always been based on a high-quality global-shutter camera along with a time-synchronized IMU, such as the setup presented in [101]. While in robotics research these assumptions are met to a large extent, for consumer-level devices as, for example, smartphones this is usually not the case. Most such devices contain rolling-shutter cameras and IMUs, which are unsynchronized with the camera and, thus usually are subject to a time offset between the two devices. In recent investigations into the use of CVI-SLAM with smartphones, first results indicate that in general, the estimation process can run smoothly, however, the quality of the estimates is significantly lower compared to the high-quality setup and, therefore, these issues need to be addressed for example by online time-offset estimation and correction as well rolling-shutter compensation. Furthermore, as the camera chips and the lenses on smartphones are generally smaller, the images are more susceptible to motion blur, leading to a devastating drop in the robustness of the system for faster motions. Hence, the robustness of the frontend data association with respect to motion blur needs to be improved, for example by employing a direct image alignment method, as these approaches are known to improve the robustness with respect to motion blur [8].

Overall, with both practical and theoretical research work in this thesis, useful insights emerge on visual-inertial SLAM and state estimation for single- and multiple agents. Focusing on visual-inertial data to perform collaborative SLAM, the presented results push the perception capabilities of multi-robot teams closer towards their deployment in real missions. The advantage of collaboration for performing SLAM could be demonstrated, opening up not only the benefit of using shared experiences with other participating agents at runtime, but also boosting the robot's perception capabilities in collaboration beyond the individual agents' capabilities. While the optimization process in the SLAM backend is key to the performance of collaborative SLAM, it also constitutes the main bottleneck for the scalability of the approaches. Applying the principle of collaboration to the SLAM backend as proposed in this thesis demonstrated that the potential benefits of robotic collaboration are vast. While several open questions remain, this work paves the way towards the deployment of larger and more collaborative robotic teams.

Paper

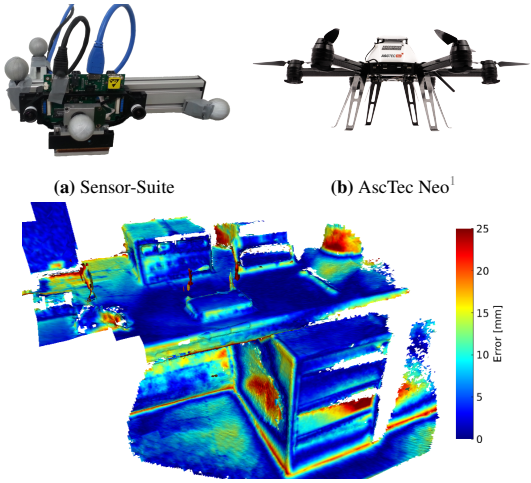
Real-time Dense Surface Reconstruction for Aerial Manipulation

Marco Karrer, Mina Kamel, Roland Siegwart and Margarita Chli

Abstract

With robotic systems reaching considerable maturity in basic self-localization and environment mapping, new research avenues open up pushing for interaction of a robot with its surroundings for added autonomy. However, the transition from traditionally sparse feature-based maps to dense and accurate scene-estimation imperative for realistic manipulation is not straightforward. Moreover, achieving this level of scene perception in real-time from a computationally constrained and highly shaky and agile platform, such as a small an Unmanned Aerial Vehicle (UAV) is perhaps the most challenging scenario for perception for manipulation. Drawing inspiration from otherwise computationally constraining Computer Vision techniques, we present a system combining visual, inertial and depth information to achieve dense, local scene reconstruction of high precision in real-time. Our evaluation testbed is formed using ground-truth not only in the pose of the sensor-suite, but also the scene reconstruction using a highly accurate laser scanner, offering unprecedented comparisons of scene estimation to ground-truth using real sensor data. Given the lack of any real, ground-truth datasets for environment reconstruction, our V4RL Dense Surface Reconstruction dataset is publicly available.

Published in:
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016
DOI: 10.1109/IROS.2016.7759259



(a) Sensor-Suite **(b)** AscTec Neo[1]

(c) Local scene reconstruction with SLAM in the loop

Figure 4.1: (a) The Sensor-Suite used for capturing visual, intertial and depth cues as well as ground truth poses. (b) UAV able to carry these sensors as well as processing all data in real-time. (c) An example local scene reconstruction obtained by the proposed method using a time-of-flight camera and color-coded according to the reconstruction error (mean error of $8mm$ in this frame, processed at $5.2ms$ per frame).

1 Introduction

For a robot to be able to interact with its environment, awareness of both its ego-motion as well as its workspace are necessary. With Simultaneous Localization And Mapping (SLAM) techniques opening up new horizons in robotic autonomy, we have witnessed a series of impressive breakthroughs to motion and environment estimation all the way from systems using range sensors on ground robots [80], [17] to real-time SLAM from a single camera [31]. It is due to this progress that vision-based flights with all processing and sensing onboard a small Unmanned Aerial Vehicle (UAV) were made possible [141]. Combining visual and inertial cues is now accepted as a powerful setup suitable for UAV navigation offering complementary sensor information at relatively low weight, power and computation resources, which are particularly limited onboard UAVs. Aiming for more robust and scalable solutions, Visual-Inertial (VI) SLAM has come a long way with systems such as [9] and [82] able to perform with unprecedented robustness, albeit still prone to drift and erroneous estimates due to common challenges, such as lighting changes and fast camera motion. With the increasing availability and afford-

ability of UAVs as well as the acquired knowledge on the controls, active interaction of a UAV with its environment is increasingly attracting the interest of the community. With robot's interaction with its environment spanning a great spectrum from guiding physical contact across a set of pre-defined waypoints on a surface [30], [88] to grasping objects during flight [109], [69], the focus in such works is on the control aspect to ensure the integrity and stability of the vehicle during such tasks. However, such works typically rely on knowing the robot's pose and its workspace, for example using an external tracking system to provide the UAV's pose and attempt to interact only with objects of simple, predefined shapes, sometimes compensating for small errors using tactile feedback [30]. In this way, the big challenge of effective and timely robotic perception of its environment are circumvented, albeit limiting the applicability of these works to real scenarios outside the controlled laboratory environment, such as the industrial manipulation tasks envisioned in [4].

Following the demand for more realistic frameworks enabling robot autonomy, recent Robotics research has been turning towards denser scene representations than traditional feature-based SLAM, borrowing ideas from Computer Vision and Photogrammetry. A significant milestone in this direction was the emergence of the Microsoft Kinect camera, which paved the way to a variety affordable depth sensors able to provide dense depth images at high frame rates. Of particular interest are also the new generations of more compact and cheaper Time of Flight (ToF) cameras.

KinectFusion [99] pioneered real-time dense scene reconstruction using a Kinect camera proposing to maintain a discretized Signed Distance Function (SDF) to fit a surface to the scene. Aiming to address the lack of scalability of [99], [115] proposed a movable reconstruction volume, while [145] proposed the use of a discretized Octree scene representation. Putting SLAM in the loop of dense scene estimation, [126] proposed an RGB-D SLAM system built on dense image alignment and an Octree-based mapping of the underlying SDF. However, despite the visually appealing scene reconstructions that they produce, all aforementioned works make use of power-hungry GPUs to compensate for the otherwise unaffordable cost of computation. As the use of such computational power is prohibitive onboard low power and low payload platforms such as UAVs, most recent research focuses on bringing such techniques on the basis of affordable CPU processing. In this direction, [127] demonstrated real-time CPU-only capability based their earlier work [126], while single-camera CPU-only reconstructions have also made their debut [38], albeit with significantly less accurate and less robust frameworks.

In realistic manipulation tasks, for example to clean a surface from oxidation, it is imperative to have a timely, dense and accurate estimation of the robot's workspace. This is especially the case in aerial manipulation, where the base of the manipulator, instead of firmly mounted on a rigid structure, is attached on a highly agile and shaky UAV, highlighting the need for real-time and precise dense scene estimation before any manipulation task can be carried out successfully. With this challenge in mind, in this paper we propose a novel system, which estimates the pose of the sensor-suite in real-time using monocular-inertial SLAM and produces a dense, local scene reconstruction based on [127] using sensing cues from a depth sensor (Fig. 4.1). We evaluate our system on a variety of challenging surfaces and camera motions with respect to scene ground truth obtained by millimetre-accurate laser scans from a Leica MS50[2] station.

[1] http://www.asctec.de

[2] http://leica-geosystems.com

	VI sensor	RS sensor	ToF sensor
TECHNOLOGY	Monochrome global-shutter CMOS, IMU	RGB imaging, IR depth imaging	Laser (VCSEL) depth imaging
WEIGHT	$130g$	$35g$	$18g$
RANGE	-	$0.5 - 5m$ indoors, varies outdoors	$0.1 - 4m$
POWER	$5W$	$1 - 1.6W$	$300mW$
IMAGING RESOLUTION	480×752	RGB: 1920×1080, IR: 640×480	172×224

Table 4.1: Specifications of the sensors used to produce the local scene reconstruction. While the VI is sufficient for pose estimation, we discuss the use of either the RS or the ToF for dense scene estimation. Note that the resolutions correspond to the maximal supported values.

While several datasets exist in the literature, recording depth values from RGB-D sensors (e.g. [131]), to the best of our knowledge, there are no datasets offering ground truth for evaluating the scene reconstruction at this level of accuracy. The dataset used in this paper, consisting of data from our hand-held multi-sensor setup (Fig. 4.1a), as well as millimetre-precise ground truth for both the pose of the sensor-suite using a Vicon[3] Tracking system and the scene using the Leica laser scanner is available online[4]. Accompanying this paper, a video is available summarizing our approach.

2 Method

2.1 Sensor Setup

Since our framework is intended for the use with an aerial robot, besides accuracy, the weight and power consumption of the sensors are also important specifications. In this paper the Intel RealSense R200[5] (RS), and the novel ToF camera CamBoard pico flexx[6] from PMD are used for the depth perception. For the onboard pose estimation, the VI sensor [101] which consist of a stereo camera pair and a time-synchronized Inertial Measurement Unit (IMU) is used. In Table 4.1 specifications for the sensors used are shown. We use Vicon, an external visual 6 degree of freedom measurement system consisting of a constellation of multiple infrared cameras, tracking markers such as the ones in Fig. 4.1a at 100 Hz at millimeter-precision. This is used to provide ground truth for the poses of the sensor-suite. The setup with the ToF mounted and the markers for the external tracker is shown in Fig. 4.1a.
For clarity, we refer to the visual image captured by the RS as the “RS image", and equivalently for the ToF we refer to the amplitude image as the “ToF image". Note that both of these sensors also provide a corresponding “depth image".

[3] http://www.vicon.com
[4] http://www.v4rl.ethz.ch/research/datasets-code.html
[5] https://software.intel.com/en-us/realsense/r200camera
[6] http://pmdtec.com/picoflexx/

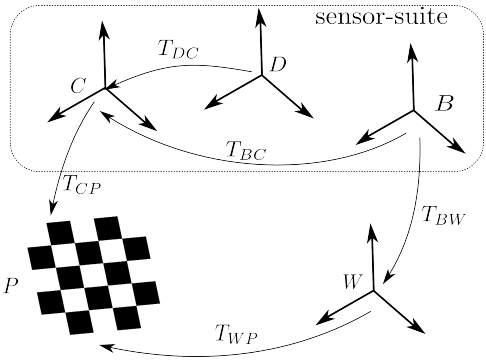


Figure 4.2: In the calibration procedure, we aim to estimate the following rigid body transformations: $\boldsymbol{T}_{CP}$ between the VI-camera (C) and the calibration pattern (P), $\boldsymbol{T}_{DC}$ between the depth sensor (D) and camera, and $\boldsymbol{T}_{BC}$ between the tracker body and camera. The pose of the tracker body in the world frame W ($\boldsymbol{T}_{BW}$) is measured given by the Vicon system. Note that during the calibration the transformation between the world frame and the calibration pattern does not need to be estimated.

2.2 Calibration

In a first step, the camera intrinsics as well as the rigid body transformation between the IMU and the camera are calibrated using the framework of [90], [46] and [45]. The intrinsic parameters of the depth sensors are used as provided by the manufacturer, which were verified using the camera calibration application from MATLAB's Computer Vision Toolbox.

The remaining calibration parameters, namely the rigid body transformations from the left VI-camera (C) to depth sensor (D), which can be either the ToF or the RS, and from the external tracker-body (B) to the camera are depicted in Fig. 4.2. To compute these transformations, a set of images $j = 1, \ldots, J$ of a checkerboard calibration pattern (P) are taken, while simultaneously capturing the pose of the tracker-body ($\boldsymbol{T}_{BW}$) from the Vicon, along with the amplitude image for the ToF or the RGB image of the RS. Instead of just using the visual images of the calibration pattern, for every frame of the depth sensor (i.e. the amplitude image of the ToF or the RGB image of the RS) we also take into account its corresponding depth image. This enables the calibration procedure to account for systematic depth errors, for example coming from deviations of the actual emitted signal and the correlation function used to capture the depth image with the ToF [77]. For the depth correction of the ToF-camera, the parametric depth correction model presented in [118] was used. The model consists of a third order polynomial in the depth as well as two first order terms to correct for possible tilt of the sensing chip, expressed:

$$\lambda^* = a_0 + (1 + a_1) \cdot \lambda + a_2 \cdot x + a_3 \cdot y + a_4 \cdot \lambda^2 + a_5 \cdot \lambda^3, \tag{4.1}$$

where λ corresponds to the measured ray-length, λ^* to the corrected ray length and $a_0, a_1, \ldots, a_5$ are the correction parameters, which have to be calibrated. The variables x and y correspond to the image coordinates. For the RS, we use a simplified model which only takes the constant offset and the linear term of Equation (4.1) into account, since it does not suffer from the effects caused by the oscillating signal typical in ToF depth imaging.
To estimate the calibration parameters of our system, a joint optimization problem is formulated by parameterizing the rigid body transform with respect to the variables $\boldsymbol{v}$. In order to have a minimal representation of the transformations, these are represented as elements of the Lie-algebra [35], for which

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} = \exp_{\mathfrak{se}(3)}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Re^6 \tag{4.2}$$

holds. Therefore, the optimization vector variables $\boldsymbol{v}$ is composed of

$$\boldsymbol{v} = \left(\boldsymbol{T}_{CP}^1, \ldots, \boldsymbol{T}_{CP}^j, \ldots, \boldsymbol{T}_{CP}^J, \boldsymbol{T}_{CD}, \boldsymbol{T}_{CB}, a_0, \ldots, a_5\right). \tag{4.3}$$

For clarity we denote the Lie-parametrization as the actual transformation matrices. The notation $\boldsymbol{T}_{CP}^j$ corresponds to the transformation $\boldsymbol{T}_{CP}$ for the j^{th} frame. We pose the optimization problem to include four different error terms, namely the reprojection errors in both the VI-images and the ToF or the RS images, the depth errors, and the pose errors of the tracker-body and the camera. For the first error term we detect the corner points of the calibration patter in each VI-image, as well as the ToF or RS images. The corresponding error term is composed as the error between the detected corner point $\boldsymbol{y}_D$ for the ToF or the RS image, $\boldsymbol{y}_C$ for the VI-image and the projection of the corresponding 3D point $\boldsymbol{y}_p$ expressed in the coordinate frame of P.

$$\boldsymbol{e}_{y_C}^j \coloneqq \boldsymbol{y}_C^j - h_C\left(\boldsymbol{T}_{PC}^j \boldsymbol{y}_p\right) \tag{4.4}$$

$$\boldsymbol{e}_{y_D}^j \coloneqq \boldsymbol{y}_D^j - h_D\left(\boldsymbol{T}_{CD}\boldsymbol{T}_{PC}^j \boldsymbol{y}_p\right) \tag{4.5}$$

The camera projection models $h_C(\cdot)$ and $h_D(\cdot)$ correspond to the VI-image and the ToF or the RS image respectively. The second error term included is built by the difference of the predicted and the corrected ray-length measured, coming from the depth image, as

$$e_\lambda^{(u,v),j} \coloneqq \lambda^{(u,v),j} - \lambda^{*(u,v),j} \tag{4.6}$$

where $\lambda^{(u,v),j}$ is the predicted ray-length computed by intersecting the ray formed by the pixel coordinates (u, v) and the plane of the calibration pattern. Furthermore, $\lambda^{*(u,v),j}$ corresponds to the corrected measurement by applying Equation (4.1). For a more detailed view of the prediction, we refer the reader to [118]. The last error term included accounts for the error between the predicted camera pose ($\boldsymbol{T}_{CP}$) and the measured pose of the tracker-body ($\boldsymbol{T}_{BW}$). In order to eliminate the transformation between the calibration pattern and the Vicon origin ($\boldsymbol{T}_{PW}$), motion is necessary corresponding to recording two poses per error term [29]. Then

the error term is expressed as an element of the Lie-group, as

$$\boldsymbol{e}_{\xi}^{j} \coloneqq \log_{SE(3)} \left(\boldsymbol{A}\boldsymbol{T}_{BC} \left(\boldsymbol{T}_{BC}\boldsymbol{B} \right)^{-1} \right) \tag{4.7}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined as

$$\boldsymbol{A} \coloneqq \left(\boldsymbol{T}_{CP}^{j} \right)^{-1} \boldsymbol{T}_{CP}^{j-1}, \quad \boldsymbol{B} \coloneqq \boldsymbol{T}_{BW}^{j} \left(\boldsymbol{T}_{BW}^{j-1} \right)^{-1} \tag{4.8}$$

The components of the objective function are defined as

$$G_{y_C} \coloneqq \sum_{j=1}^{J} \boldsymbol{e}_{y_C}^{j^T} \boldsymbol{W}_{y_C}^{-1} \boldsymbol{e}_{y_C}^{j} \tag{4.9}$$

$$G_{y_D} \coloneqq \sum_{j=1}^{J} \boldsymbol{e}_{y_D}^{j^T} \boldsymbol{W}_{y_D}^{-1} \boldsymbol{e}_{y_D}^{j} \tag{4.10}$$

$$G_{\lambda} \coloneqq \sum_{j=1}^{J} \sum_{(u,v)\in A} \frac{1}{w_{\lambda}} e_{\lambda}^{2} \tag{4.11}$$

$$G_{\xi} \coloneqq \sum_{j=2}^{J} \boldsymbol{e}_{\xi}^{j^T} \boldsymbol{W}_{\xi}^{-1} \boldsymbol{e}_{\xi}^{j} \tag{4.12}$$

where A denotes the pixels of the depth image corresponding to the area covered by the calibration pattern. The weight-matrices $\boldsymbol{W}_{y_C}, \boldsymbol{W}_{y_D}$ and $\boldsymbol{W}_{\xi}$ were chosen to be diagonal with the corresponding variance expected considering the calibration of the individual sensors or the manufacturer's specification on the uncertainty. The scalar weight w_{λ} corresponds to the noise variance of the depth sensor. The objective function is composed as:

$$G = G_{y_C} + G_{y_D} + G_{\lambda} + G_{\xi} \tag{4.13}$$

For the initialization of the camera poses the linear solution to the extrinsics problem is used. The transformation between the depth sensor and the camera is initialized using a least squares solution to the reprojection error on a subset of the calibration images. The initial guess for the transformation between the tracker-body and the camera is obtained using the algorithm of [29]. For the optimization of the objective function we use the Levenberg-Marquardt algorithm.

2.3 Surface Reconstruction

We employ the framework of [127] to estimate the surface and extend this to account for any priors on the uncertainty of the depth measurements as well as with an efficient method for performing a local reconstruction as elaborated in Section 2.4. At the core of the approach of [127] lies the Octree structure, which at its leafs, called "bricks" (b), stores the value of the SDF

in a 8^3 voxel volume. With the arrival of a new depth image, the corresponding brick for every pixel is looked up and put into a queue. The queue is then iterated over and for every voxel contained inside of the brick, its position $\boldsymbol{p}$ in the world frame transformed into the frame of the depth sensor $\boldsymbol{p}_D$. The measured point is computed using the linear inverse projection model $\tilde{h}_D^{-1}(\cdot)$ on the undistorted depth image and the depth measurement $Z(u,v)$ at pixel coordinates (u,v), as

$$\boldsymbol{p}_{obs} = \tilde{h}_D^{-1}\left(u, v, Z(u,v)\right) \tag{4.14}$$

Due to the use of a truncated version of the SDF, defined as

$$\Delta_D = \max\left\{\min\left\{\Phi, |\boldsymbol{p}_D - \boldsymbol{p}_{obs}|\right\}, -\Phi\right\} \tag{4.15}$$

where Φ is the truncation threshold and the multiscale approach, the number of bricks queued per frame are limited. The update of the stored distance values $D(\boldsymbol{p},t)$ at the voxel position $\boldsymbol{p}$ at time t is performed as a weighted running average

$$D(\boldsymbol{p},t) = \frac{D(\boldsymbol{p},t-1)W(\boldsymbol{p},t-1) + \Delta_D w(\Delta_D)}{w(\Delta_D) + W(\boldsymbol{p},t-1)} \tag{4.16}$$

Where $W(\boldsymbol{p},t)$ corresponds to the accumulated weight at position $\boldsymbol{p}$ and time t. For the weight increment $w(\Delta_D)$ different weighting schemes can be used as presented in [13]. In [127] $w(\Delta_D)$ is constant for areas in front of the surface and decreases linearly until zero behind the surface, i.e.

$$w(\Delta_D) = \begin{cases} 1 & \text{, if} \quad \Delta_D < \delta \\ \frac{\Phi - \Delta_D}{\Phi - \delta} & \text{, if} \quad \Delta_D \geq \delta \quad \text{and} \quad \Delta_D \leq \Phi \\ 0 & \text{, if} \quad \Delta_D > \Phi \end{cases} \tag{4.17}$$

where δ can be seen as an allowed penetration depth of the measurement. This weighting scheme solely relies on the geometric distance to the measurement, but does not take any additional information into account. A simple noise estimate for example for a ToF camera can be obtained as described by [84]. Since the ToF camera used provides information about the noise level of each pixels, we propose a weighting scheme to incorporate these in the weighting function using it as a scaling factor. Assuming a noise value $\sigma(u,v)$ at pixel coordinates (u,v), we compute a scaling factor according to

$$f_\sigma = \begin{cases} 1 & \text{, if} \quad \sigma(u,v) \leq \sigma_{min} \\ \frac{\sigma_{min}}{\sigma(u,v)} & \text{, if} \quad \sigma_{min} < \sigma(u,v), \end{cases} \tag{4.18}$$

where σ_{min} is a parameter that can be chosen according to the expected noise of the sensor. In this way, we aim to take the sensor's estimated uncertainty into account when weighting the different incoming votes of the voxels. This results to more informed estimation of the surface, providing robustness to common bottlenecks, for example too oblique angles of incidence. The adapted weighting scheme is defined as:

$$\tilde{w}(\Delta_D) = f_\sigma \cdot w(\Delta_D) \tag{4.19}$$

This simple scheme allows to incorporate sensor-specific uncertainty measurement, which in case of our ToF camera is available anyway, without degrading the computational efficiency of the estimation. For the RS, we use the uniform weighting scheme according to Equation (4.17). The SDF representation has a large memory footprint, which e.g. limits cooperative interactions between multiple agents due to limited bandwidth. Therefore, and also for visualization purposes, the algorithm of [127] keeps track of the updated bricks and performs a re-meshing on those areas using an adapted version of a marching cubes algorithm in order to account for the multiscale approach. For a detailed explanation of the meshing algorithm, we refer to [127].

2.4 Surface Reconstruction using SLAM Poses

In order to use the reconstruction algorithm with SLAM poses, typically subject to drift, we implemented a scheme which is able to maintain a locally accurate scene estimate. We implement this by introducing a visibility constraint in the sense that we only keep the portion of the reconstruction, which received measurements within the time horizon t_h and discard the rest. This is done by keeping track of the time-stamp of updated bricks and maintaining the local scene reconstruction within t_h with respect to the current camera pose. We denote $\mathcal{B}_s$ as the set of bricks used for the surface estimation at the current time and $\mathcal{B}_o$ as the bricks to be updated. The notation $t(b_i)$ corresponds to the latest time-stamp that brick b_i was updated. The actual voxels grouped inside the brick b_i are indicated by $\boldsymbol{p}_j$, which corresponds to the voxel center coordinates. The update procedure is shown in Algorithm 1. The time horizon t_h influences the behavior of the algorithm. For larger t_h during exploratory motion, drift of the SLAM system result in larger reconstruction errors and higher computational cost as more bricks need to be tracked, i.e. the size of $\mathcal{B}_s$ increases. On the other hand, if the chosen t_h is too small, even quick deviations from the current viewpoint, e.g. by wind disturbances, can result in loss of the previous reconstruction.

The reconstruction algorithm itself is agnostic to the SLAM technique used. For the pose estimation in our system, we use the framework proposed by [83], which uses the sensor readings of both the IMU as well as the camera input streaming from the VI-sensor. The system is based on sparse features on a set of keyframes as well as the incorporation of IMU measurements into a local graph. The window of keyframes is kept bounded by marginalizing out old keyframes. This allows the algorithm to run in real time on a CPU, while maintaining an accurate pose estimate. However, due to the local keyframe approach the system is still prone to global drift. The framework can be run using multiple cameras or as a monocular system. Motivated by the lower computational complexity, while maintaining a comparable performance, here we use the monocular version of the algorithm for state estimation.

3 Experimental Results

3.1 V4RL Dense Surface Reconstruction Dataset

In order to evaluate the performance of the system in terms of accuracy, a dataset consisting of the VI data, as well as the data of the depth sensor (RS or ToF) and the pose of the tracker-body

Algorithm 1 Time Window for Local Reconstruction

$\mathcal{B}_o :=$ all bricks observed from the current pose
$t \leftarrow$ current time-stamp
for all $b_i \in \mathcal{B}_o$ **do**
  if $b_i \notin \mathcal{B}_s$ **then**
    add b_i to $\mathcal{B}_s$
  end if
end for
for all $b_i \in \mathcal{B}_s$ **do**
  if $b_i \in \mathcal{B}_o$ **then**
    $t(b_i) \leftarrow t$
    for all $\boldsymbol{p}_j \in b_i$ **do**
      update $W(\boldsymbol{p}_j, t)$
      update $D(\boldsymbol{p}_j, t)$
    end for
  else if $(t - t(b_i)) > t_h$ **then**
    remove b_i from $\mathcal{B}_s$
  end if
end for
$\mathcal{B}_o \leftarrow \emptyset$

from the Vicon system were recorded. Complementary, the observed scene was scanned using a Leica MS50 laser scanner from multiple viewpoints in order to obtain ground truth for the scene, against which we can compare the estimated reconstruction. This dataset, which includes scene and poses' ground truth for the first time, is publicly available[4].

We choose two different scenes to evaluate the system; one generic desk sequence containing objects of different texture and material (e.g. affecting their reflectance properties) as shown in Fig. 4.3, as well as a more industrial object consisting of a Pipe structure typical in an industrial inspection scenario (ground truth reconstruction in Fig. 4.4). The recordings were made using a hand-held setup in order to ease testing different types of motions (i.e. of different speed, shaky, different viewpoints) as well as to avoid changes of the scene setup due to the airstream of a UAV.

The scene ground truth was captured by 8 separate scans for the Desk scene with a total number of approximately 10M points and 6 scans for the Pipe structure with 2.5M points in total. The scans were pre-aligned by localizing the Leica station with respect to a set of known markers in the room. The final alignment was performed using the Iterative Closest Point (ICP) algorithm on adjacent scans, incrementally building the ground truth point cloud. Outlier filtering using radius search was applied on the obtained point clouds, while points with low neighborhood support were also removed. The final ground truth point clouds for both scenes are shown in Fig. 4.4.

The properties of the sequences used are listed in Table 4.2. Note that the trajectories using the ToF and the RS are similar in each case but cannot be identical due to the hand-held setup.

Figure 4.3: A view of the Desk scene used for the evaluation together with the Leica MS50 station used to obtain the ground truth scene reconstruction.

Figure 4.4: The registered laser scans of mm-precision used as scene ground truth, left: Pipe structure, right: Desk scene

Scene Type	Name		Average linear velocity $[m/s]$	Average rotation velocity $[^\circ/s]$	Trajectory length $[m]$
Desk	d1	ToF	0.2	13.6	25
	d2		0.3	29.6	40
	d3		0.2	12.9	17
	d4		0.3	25.1	24
	d1	RS	0.2	13.2	25
	d2		0.3	29.5	43
	d3		0.2	12.7	14
	d4		0.3	27.5	22
Pipes	p1	ToF	0.3	17.1	22
	p2		0.3	24.2	20
	p3		0.1	6.4	6
	p4		0.2	25.8	11
	p1	RS	0.3	16.2	24
	p2		0.3	26.3	20
	p3		0.1	7.7	5
	p4		0.3	23.9	10

Table 4.2: Properties and naming convention of the V4RL Dense Surface Reconstruction dataset.

3.2 Reconstruction Accuracy (Global)

To quantitatively evaluate the reconstruction accuracy isolating it from any pose errors, the scenes are reconstructed using the ground truth poses from the Vicon system. The reconstruction is performed by fusing the full data sequence in the system for every trajectory. The voxel resolution at the finest level is chosen to have a side length of $6mm$. The transformation between the world frame (of the Vicon system) and the origin of the scene ground truth is only known approximately, since the world frame origin of the Vicon system can only be selected manually. Therefore, for every comparison we perform an ICP alignment of the estimated reconstruction to the ground truth. Although the obtained reconstruction is represented as a mesh, we consider its vertex points for the alignment. For the evaluation process, we compute the distance of every vertex of the reconstruction to its Nearest Neigbhor (NN) in the ground truth point cloud. In order to correct for regions, where no ground truth data is available (due to occlusions, shiny surfaces, etc.), we do not consider vertices whose NN is further away than a certain maximal distance d_{max}. The distance d_{max} was chosen to be 50mm for the Desk scene and 30mm for the Pipe structure, respectively. In Fig. 4.5, an example of a reconstruction color-coded with respect to the NN-distance is shown. The NN-based errors are recorded for every sequence of the two scenes, using both the RS and the ToF. For the ToF camera, both the standard weighting scheme of [127] as well as our proposed weighting system (according to Section 2.3) are evaluated. The reconstruction accuracies achieved in each case are summarized in Fig. 4.6.

On both the Desk scene as well as the Pipe structure, the RS achieves higher accuracy compared to the ToF. This is caused by the higher resolution ($2\times$) of and the higher frame rate ($3\times$) of the RS, which results in a higher information density. For the ToF our proposed weighting scheme improves the accuracy on all sequences on average by 10% up to a maximal improvement of 17%. The error level on the Desk scene is higher which is caused not only by the larger

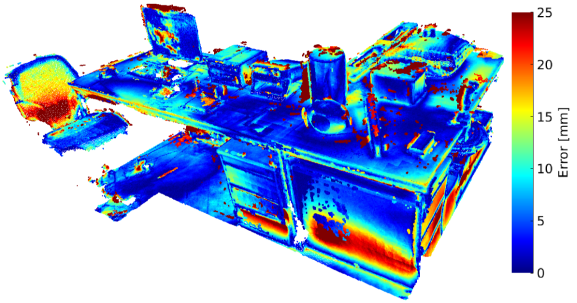


Figure 4.5: Resulting reconstruction error using ground truth poses along with the ToF depth measurements on the Desk scene, while applying the proposed weighting scheme.

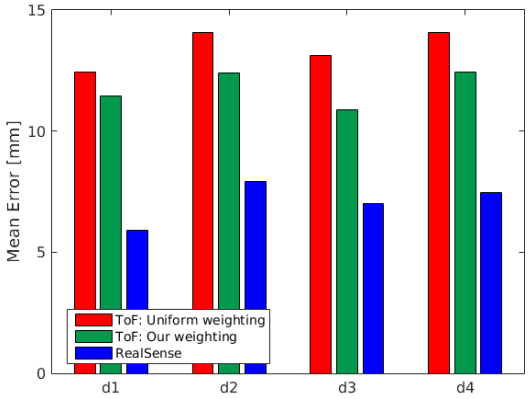
size of the scene, but also due to the different materials and larger variety in the angles of incidence. Furthermore, in the sequences of the Pipe structure all areas of the object are observed from a similar number of views and viewpoints, whereas for the Desk scene some regions are only viewed quickly, while others are observed more thoroughly.

3.3 Local Reconstruction with SLAM in the loop

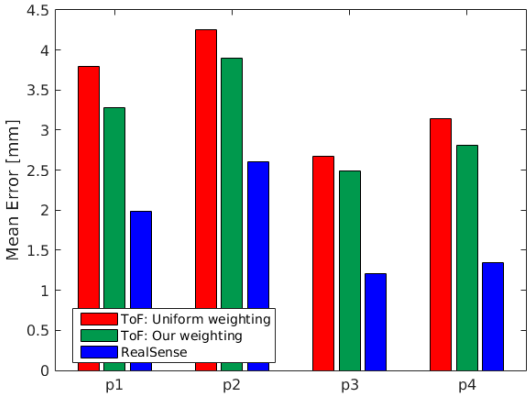
To evaluate the estimated reconstruction when using SLAM for the pose estimation, we applied the time window discussed in Section 2.4 resulting to a local estimation of the scene. The time window t_h for this procedure was set the constant value of $3s$, which gives some tolerance to shaky motions of the camera when observing an area, while still keeping the region of the estimated reconstruction small enough to avoid significant motion drift. Instead of the global model, the most recently estimated local reconstructions were stored at a frequency of about 1 Hz. We aligned each of these local reconstructions to the corresponding scene ground truth using ICP and used the same error metric as for the global reconstruction. The corresponding mean error values are shown in Fig. 4.7.

Surprisingly, the average error for the ToF on the Desk scene is lower than when using the ground truth poses (Section 3.2). This is due to the fact that the impact on the overall error coming from poorly reconstructed areas (e.g. due to oblique angles of incidence, few views) is diminished when considering the average error of local reconstructions. Using the time horizon scheme, there is significant overlap in the regions considered at consecutive reconstructions, therefore these areas are bound to have better accuracy and can overall reduce the average reconstruction error.

When reconstructing the scene locally using RS cues, the average reconstruction error is increased and the accuracy disadvantage of ToF-based global reconstruction is diminished. The explanation for this is multi-fold; firstly, because the variability of viewpoints during local reconstruction is limited, any internal calibration errors of the RS become more evident (i.e. resulting to erroneous depth values at contour edges). Moreover, although the maximum range of



(a) Desk Scene



(b) Pipe Structure

Figure 4.6: Average reconstruction error using ground truth poses. Note: the labels $d1 - d4$ and $p1 - p4$ correspond to the sequences in the dataset defined in Table 4.2

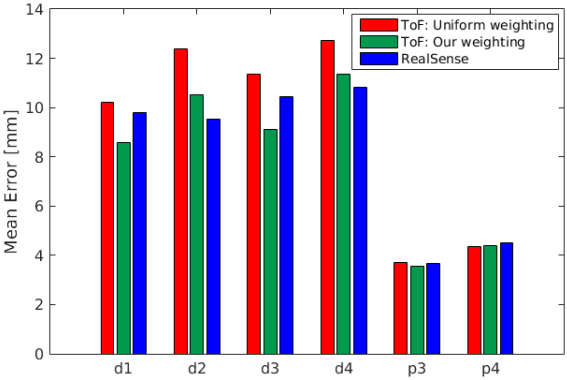


Figure 4.7: Average errors of the local reconstructions using poses from VI-SLAM [83]. Note that for the Pipe structure only sequences $p3$ and $p4$ are used, since for the sequences $p1, p2$ there was a significant yaw drift in the SLAM poses when turning around the structure.

the ToF and the RS (Table 4.1) appear similar in their specification, in practice we observed that the RS usually perceives more distant measurements increasing the sensitivity of the acquired data (and thus, the reconstruction) to angular pose errors.

3.4 Evaluation of Computational Performance

A thorough investigation of the computational performance of the reconstruction and meshing algorithm was performed by [127]. Hence, we focused our investigation on the comparison between the ToF and the RS as well as on the difference between our weighting scheme and the standard implementation. Furthermore, we evaluate the computational overhead induced by applying the time window based Local reconstruction with SLAM in the loop, versus the full Global reconstruction with Vicon poses. We conducted the experiments on the sequences used in Section 3.3, while setting the voxel size at the finest level to $8mm$. The timings were measured on a Intel Core i7-4710MQ CPU running at 2.5 GHz and are presented in Table 4.3. The computational overhead of our proposed weighting scheme is minimal, on average about $0.4ms$, while reducing the reconstruction error on average by 10%. The difference in the update times between the RS and the ToF stems from the higher resolution of the RS compared to the ToF. Using the local reconstruction over the time horizon t_h increases the computational complexity by $\mathcal{O}(n)$, with n being the number of bricks tracked within the time horizon t_h. Table 4.4 records the average number of bricks per frame tracked during the reconstruction as well as the percentage of these which get updated (per frame). Generally, faster motion or finer the voxel resolution (i.e. more detailed reconstruction) results the higher number of tracked bricks, which influences directly the execution time per update step. The RS requires on average 3 times more bricks tracked compared to the ToF due to its higher resolution as well

Sequence	Fusion Time per Frame [ms]					
	ToF Uniform Weighting		ToF Our Weighting		RS	
	Global	Local	Global	Local	Global	Local
d1	4.2	4.6	4.5	4.9	11.0	12.4
d2	4.6	5.1	5.0	5.5	10.8	12.3
d3	4.3	4.7	4.6	5.0	10.0	11.2
d4	4.4	4.9	4.7	5.2	10.0	11.4
p3	2.2	2.3	2.3	2.4	6.0	6.4
p4	2.1	2.2	2.2	2.3	6.6	7.2
Average	3.6	4.0	3.9	4.2	9.1	10.1

Table 4.3: Average time per frame for fusing a new depth image in the model. The columns labeled "Global" correspond to the timings recorded when using ground truth poses with full scene reconstruction, "Local" corresponds to the timings when using SLAM for the pose estimation along with a $3s$ time window for the reconstruction. Despite the slightly increased time necessary for the "Local" reconstruction, our method is always real-time.

Sequence	Number of Bricks Tracked		Updated Bricks [%]	
	ToF	RS	ToF	RS
d1	1544	5182	37.4	17.8
d2	2468	6636	28.2	13.1
d3	1658	4867	37.2	15.9
d4	2106	5920	29.3	13.8
p3	543	1843	45.3	16.0
p4	733	2875	30.6	13.8
Average	1509	4554	34.7	15.1

Table 4.4: Average number of bricks tracked using the time window based Local reconstruction, and the percentage of bricks updated related to the number of tracked bricks.

as its larger measurement range. More execution time is consumed on average to achieve the Local reconstruction, but this is still real-time for the ToF ($< 6ms$) and the RS ($< 13ms$).

In manipulation tasks, we cannot afford to compromise the reconstruction accuracy as this can be catastrophic, especially in the case of UAV manipulation. As a result, in order to ensure sufficient quality of the reconstruction even in the presence of inevitable global drift accumulating in the SLAM pose estimation, a small overhead in execution time is acceptable. In the case of the RS Local reconstruction, this overhead becomes more evident due to a larger number of bricks to track. Overall, the RS outperforms the ToF in terms of accuracy on the global scale (Section 3.2), but in terms of local accuracy (Section 3.3) both the RS and the ToF perform similarly. As a result, considering that ToF offers a significant advantage in execution time, it poses a better choice for a setup with limited computational resources, such as onboard a UAV.

4 Conclusions and Future Work

In this paper we present a system capable of accurately reconstructing a local scene in real-time, suitable for manipulation tasks even from a highly agile platform, such as a UAV. Fusing depth cues at frame rate, while estimating the pose of the sensor-suite using visual-inertial SLAM, our experimental evaluation on a variety of challenging scenarios reveals the high fidelity of the system achieving reconstruction accuracy of the order of $10mm$ on average.

A thorough evaluation of the proposed approach was presented assessing the accuracy of the obtained 3D reconstruction both on a global scale using ground truth poses and ground truth scene reconstruction, as well as for local reconstructions using poses obtained by a nominal visual-inertial SLAM system. As no such testbed (with real sensing data and scene ground truth) exists in the literature, we release our dataset consisting of visual, inertial and depth data from a time-of-flight and an RGBD camera, as well as pose and scene ground truth of millimeter precision. Future work includes employing this reconstruction framework to perform simple manipulation tasks from a UAV, as well as research into UAV path planning for viewpoints promising more accurate reconstructions.

Paper II

Towards Globally Consistent Visual-Inertial Collaborative SLAM

Marco Karrer and Margarita Chli

Abstract

Motivated by the need for globally consistent tracking and mapping before autonomous robot navigation becomes realistically feasible, this paper presents a novel back-end to monocular-inertial odometry. As some of the most challenging platforms for vision-based perception, we evaluate the performance of our system using Unmanned Aerial Vehicles (UAVs). Our experimental validation demonstrates that the proposed approach achieves drift correction and metric scale estimation from a single UAV on benchmarking datasets. Furthermore, the generality of our approach is demonstrated to achieve globally consistent maps built in a collaborative manner from two UAVs, each equipped with a monocular-inertial sensor suite, showing the possible gains opened by collaboration amongst robots to perform SLAM.

Published in:
IEEE International Conference on Robotics and Automation (ICRA), 2018
DOI: 10.1109/ICRA.2018.8461213

1 Introduction

One of the key pre-requisites in the quest of employing mobile robots with navigational autonomy is the development of their ability to perceive their workspace and estimate their egomotion within it, which is commonly referred to as Simultaneous Localization And Mapping (SLAM). While initial attempts to address SLAM have been utilizing range sensors, it was the emergence of monocular and real-time capable SLAM systems, such as [31] and [70] that paved the way towards the use of SLAM onboard small Unmanned Aerial Vehicles (UAVs). The employment of Visual-Inertial (VI) sensing cues and the successful demonstration of vision-controlled flights using onboard sensing only [141], rendered this sensor suite as the standard choice for the control and navigation of small aircrafts.

With increasing maturity and robustness in this field, two state of the art methods for Visual Inertial Odometry (VIO) open-sourced their implementations, namely OKVIS [82] and ROVIO [9]. Such systems permit reliable state estimation even during complicated UAV maneuvers. However, as these algorithms are only local, the current UAV pose that is being estimated is prone to drift over longer trajectories. Aiming to address drift during real-time monocular state estimation, ORB-SLAM [96] pushed the state of the art, tackling large-scale loop correction at an unprecedented robustness and accuracy in monocular systems. Incorporating additional inertial data to the monocular setup, the most recent VI-ORB-SLAM [98] was the first VI-SLAM system capable of correcting drift via loop-closure detection and optimization, while maintaining an estimate of metric scale with high accuracy. Despite constituting a milestone, VI-ORB-SLAM remains closed source and based on the authors' evaluation [98] as the only source of information, its accuracy is reportedly fluctuating across different datasets, highlighting the need for deeper analysis in VI-SLAM.

Moving on from single-robot SLAM systems, the community started making the first steps towards investigating collaborative SLAM in multi-robot scenarios. While [149] for example, leverages the multi-camera setup with view overlap to perform SLAM in challenging dynamic scenes, [43] and [119] explore the advantages of employing multiple UAV equipped with cameras for efficient mapping and collaborative SLAM, respectively. Due to the lack of metric measurements (e.g. inertial data), these systems can only provide estimates up to scale. Instead, the approach in [5] for collaborative stereo from two UAV is capable of estimating the relative pose of two VI systems in simulation, albeit avoiding to address the global consistency of the estimation processes.

While the aforementioned open-sourced VIO systems have been very influential in robot navigation, their inevitable tendency to drift, limits their applicability in real scenarios, where global state estimation is required. In this spirit, we present a carefully designed back-end, which in combination with a nominal VIO system enables the generation of a globally consistent map at comparable accuracy with the state of the art VI-SLAM systems – at times even achieving error reduction of over 50%, solely considering the back-end optimization. Moreover, here we go a step further to illustrate the use of the proposed back-end with two UAVs to achieve collaborative mapping, while correcting for drift upon loop-closures, both within each trajectory as well as across trajectories of different UAV as shown in Fig. 5.1. This paper outlines a new, complete back-end system in enough detail to enable reproducibility of the proposed system, employable in combination with an off-the-shelf VIO system requiring only minimal modification. Furthermore, our evaluation on benchmarking datasets reveals that the proposed

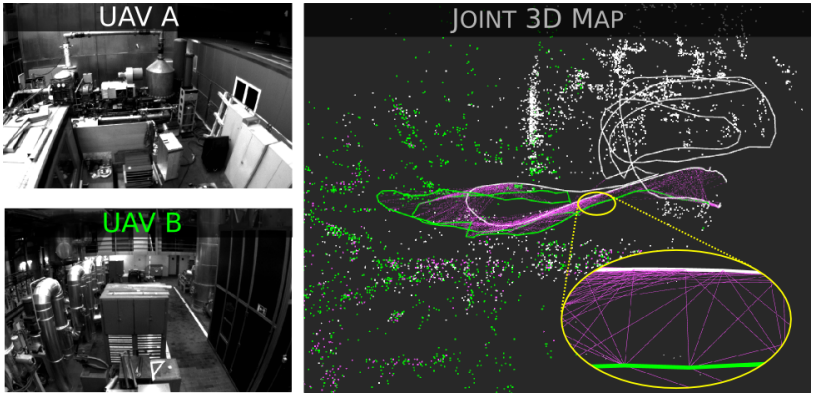


Figure 5.1: A snapshot of the proposed system in a collaborative setup with two UAVs. On the left the viewpoints from each UAV are shown, while on the right is the joint 3D map built collaboratively.Trajectories and landmarks are colored in white and green for UAV A and UAV B, respectively. Landmarks that are shared across both UAVs are indicated in magenta, while covisibility edges, connecting keyframes accross the two UAVs are also in magenta, magnified in the inset for clarity.

framework can achieve significant improvement in accuracy over the state of the art.

2 Preliminaries

2.1 Notation

In this paper, we use bold capital letters for matrices (e.g. $\boldsymbol{A}$), bold small letters for vectors (e.g $\boldsymbol{a}$), and capital letters for coordinate frames (e.g. A), while sets of variables are denoted by calligraphic letters (e.g. $\mathcal{A}$). A rigid body transformation from coordinate frame B to A is denoted by $\boldsymbol{T}_{AB}$, while the rotational part of any transformation $\boldsymbol{T}$ is denoted by $\boldsymbol{R}$ and the translational part by $\boldsymbol{t}$. A vector $\boldsymbol{x}$ expressed in coordinate frame A is written as ${}_A\boldsymbol{x}$. The origin (i.e. the inertial frame) of the system is denoted by W (also referred to as the world frame), the camera coordinate system by C, and the Inertial Measurement Unit (IMU) body frame by S.

2.2 IMU Model and State Representaion

It is well known that readings from MEMS-IMUs do not capture the true acceleration and rotational velocity, but rather a biased version of them. While some errors, such as cross-couplings and scaling errors are constant and can be compensated for via factory calibration, other influences are time-variant and need to be estimated online. In order to model the IMU

measurements, we use the standard measurement model, assuming that the accelerometer ${}_S\boldsymbol{a}(t)$ and the gyroscope measurements ${}_S\boldsymbol{\omega}_{WS}(t)$ are both corrupted by additive white noise $\boldsymbol{\eta}$ and have a sensor biases $\boldsymbol{b}$, which are assumed to be varying slowly over time (t), such that:

$$ {}_S\boldsymbol{a}(t) = \boldsymbol{R}_{WS}^{\mathsf{T}}(t)\,({}_W\hat{\boldsymbol{a}}(t) - {}_W\boldsymbol{g}) + \boldsymbol{b}_a(t) + \boldsymbol{\eta}_a(t)\,, \tag{5.1} $$

$$ {}_S\boldsymbol{\omega}_{WS}(t) = {}_S\hat{\boldsymbol{\omega}}_{WS}(t) + \boldsymbol{b}_g(t) + \boldsymbol{\eta}_g(t)\,. \tag{5.2} $$

The notation $\hat{\cdot}$ signifies the true values of the respective variables, while ${}_W\boldsymbol{g}$ is the gravity vector in the inertial frame. We differentiate the accelerometer-specific variables from the gyroscopic ones via the subscripts a and g, respectively.

Due to the characteristics of the IMU measurements, the state of the system $\boldsymbol{\Theta}$ includes the poses $\{\boldsymbol{R}_{WS}, \boldsymbol{t}_{WS}\}$ of all the Keyframes (KFs) in the trajectory, the positions ${}_W\boldsymbol{l}$ of all of the landmarks ever experienced, as well as the linear velocities ${}_W\boldsymbol{v}$ and bias terms $\boldsymbol{b}$:

$$ \boldsymbol{\Theta} := \{\underbrace{\boldsymbol{R}_{WS}^k, \boldsymbol{t}_{WS}^k, {}_W\boldsymbol{v}^k, \boldsymbol{b}^k}_{\mathrm{KF}_k}, {}_{S_r}\boldsymbol{l}^i\} \quad \forall k \in \mathcal{V}, \forall i \in \mathcal{L}\,, \tag{5.3} $$

where $\mathcal{V}$ is the set of all keyframes and $\mathcal{L}$ is the set of all landmarks. Instead of expressing the landmarks in the global reference frame (W), we express them in local coordinates of a reference KF S_r as proposed in [7]. In combination with an inverse-depth parametrization [23], this aims at improving the conditioning of the problem during the optimization. However, for the sake of readability, we will treat the landmarks as if they were expressed in Euclidean coordinates. In this paper, we refer to individual state variables as θ_j.

3 Method

We consider the setup of two UAVs equipped with a monocular camera and an inertial sensor each, experiencing the world at the same time, while exhibiting an overlap in their fields of view. Following this paradigm, this section gives an overview of the proposed system to arrive to a joint, globally consistent map of the UAVs' surroundings and their relative poses within it.

3.1 System Overview

The proposed system, illustrated in Fig. 5.2, employs a front-end VIO module onboard each UAV and then processes all information gathered from the UAVs to perform Landmark Matching and Mapping, Loop-Closure Detection, and Local and Global Bundle Adjustment (BA) on all estimates. VIO ensures a stable pose estimation of each UAV in six Degrees of Freedom (DoF) and is expected to drift, but can be used to safely stabilize the UAV. The decoupling of the VIO from the rest, the map management threads that can run on a ground station; VIO communicates KF messages to the back-end. While the absence of feedback from the global map to the VIO prohibits direct corrections of the VIO's state upon map changes, it enables the use of an off-the-shelf VIO with only minimal modifications. Furthermore, as transformation between the global map's and the VIO's coordinate can be easily estimated, the UAV would still be able to make use of corrections, as e.g. presented by [102].

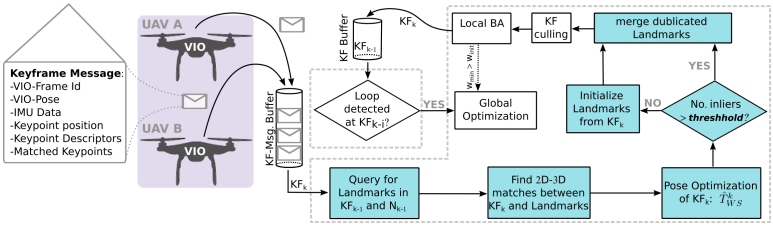


Figure 5.2: A schematic of the proposed pipeline to fuse the experiences of multiple UAVs into a joint, globally consistent map, by reusing information obtained by the VIO running onboard each UAV (in purple). At first, correspondences between keyframes and landmarks are established and new landmarks get initialized (boxes in cyan). The scene structure and UAVs' poses are then optimized frequently on a local scope and upon detection of loop closures, optimization is performed on a global scale.

During Landmark Matching and Mapping (in cyan in Fig. 5.2), past observations get associated with the landmarks in the resulting, joint map from both UAVs, while new landmarks get initialized in this map. The map comprises of a set of 3D landmarks and KFs, where each KF consists of the corresponding UAV pose, a set of 2D observations and the landmarks visible from it. Each landmark in the map stores the KF-IDs that have observed it, an estimate of the local surface normal based on the viewing angles of all corresponding observations, as well as the most representative image descriptor for this landmark across all observations, as proposed in [96] – this aims to increase the re-detectability of the landmarks.

A Covisibility Graph is maintained throughout each session, with nodes corresponding to individual KFs. Two nodes share an edge if the corresponding KFs share a minimum number of landmark observations ($\alpha_{min} = 12$ in our implementation), and each edge is associated with a weight α reflecting the number shared landmark observations. An Essential Graph is also maintained (this notion was first introduced in [96]), which is of similar structure to the Covisibility Graph, only preserving the most essential information, by restricting edges even more (e.g. $\alpha_{min} = 100$). In addition to the purely spatial KF covisibility, we also keep track of the temporal predecessor of each KF, distinguishing the agent (i.e. here the UAV), from which the KF originates, as this necessary for the constraints used to obtain metric scale.

Since Loop-Closure Detection is mostly independent of Mapping and Local BA they run in separate threads. However, in case of a loop-closure, the system waits until the Local BA for the current KF has finished, and then triggers the loop correction, blocking the processing of new KFs until the map is updated with the result of the Global BA. At the core of the proposed system is the optimization of both the KF poses and scene structure (i.e. landmarks) simultaneously, including any IMU readings obtained between consecutive KF poses. This aims at recovering each UAV's trajectory in metric scale. Local BA is performed for more frequent small-scale corrections, while Global BA is used to optimize all poses and landmarks obtained from all participating agents.

3.2 Visual-Inertial-Odometry Input

The proposed system is generally independent of the choice of the VIO pipeline used, with the only requirement of providing metrically scaled current poses and corresponding 2D observations. For the experiments presented in this paper, the publicly available VIO system OKVIS [82] is employed with some adaptations to reuse the matching results. OKVIS performs a joint, non-linear optimization over a constant number of KFs, including inertial measurements.

Packing the relevant information for the current KF as provided by VIO into KF-messages, these then serve as input to the proposed back-end framework. Each KF-message encloses the current KF's pose, the IMU readings since the last KF, the locations of the current KF's kyepoints in the image, and their corresponding descriptors – thus, eliminating the need for sending full images. Each KF-message also includes a list of global identifiers of its associated keypoints, enabling tracing of the KF's keypoints back to older KFs that they were matched from, within the local window (of retained KFs) of VIO. This enables re-use of data associations as discussed in the next section. Note that this latter part of a KF-message is optional, as for example, filter based VIO systems (e.g. [9]) may not have this information available. As the KFs arrive at an unknown rate, we store arriving KF-messages within a first-in-first-out buffer, before processing them sequentially.

3.3 Landmark Matching and Mapping

Landmark matching consists of establishing correspondences of the current frame to the existing landmarks (via 3D-2D matching) and the creation of new correspondences (via 2D-2D matching) across different KFs. In order to establish matches to existing landmarks, every observation in the current KF is checked for correspondences with the past KFs via the global landmark identifiers listed in the KF-message. Before accepting a new correspondence, this is checked for consistency in terms of the reprojection error and the descriptor distance within the map. In order to establish additional 3D-2D correspondences, or in case the VIO system at hand does not provide matching information, the system uses the relative transformation $\boldsymbol{T}_{k-1,k}$ between the current keyframe (KF_k) and the previous KF (KF_{k-1}) stemming from the same agent, as estimated by the VIO. Given an estimate for the pose $\boldsymbol{T}_{WS}^{k-1}$ of KF_{k-1}, the system predicts the current pose ($\hat{\boldsymbol{T}}_{WS}^{k}$) as

$$\hat{\boldsymbol{T}}_{WS}^{k} = \boldsymbol{T}_{WS}^{k-1} \cdot \boldsymbol{T}_{k-1,k} \,. \tag{5.4}$$

As a result, all landmarks predicted to be visible in KF_k from KF_{k-1} and its first-order neighbors (N_{k-1}) in the Covisiblity Graph, are projected in it. Similarly to [96], the search for matching observations is restricted within a radius around the predicted projection of a landmark, while a correspondence is established to the observation with the smallest descriptor distance. In case of multiple landmarks matching to the same 2D observation, only the correspondence to the landmark with the biggest number of observations is established, or the landmark with the smallest descriptor distance, if the first criterion is inconclusive. This process is performed first using a large radius (i.e. for coarse matching) followed by solving the P3P problem as in [71] for a number of RANSAC iterations (40 iterations) for outlier filtering on the initial correspondences. The projection based matching is then repeated, using the pose obtained by the

previous RANSAC step, with a more restrictive radius to find additional matches. Using all the established correspondences, the current KF pose is refined by minimizing the reprojection error of the matched landmarks in the current KF, while keeping the landmark position fixed.

Initialization of new landmarks is only performed when the UAV is in an *exploratory state*, which is determined by a minimum number on the 3D-2D inlier correspondences found (here 60). In order to initialize new landmarks, we first attempt to triangulate the remaining correspondences obtained by the VIO system (using their identifiers), for which no 3D association was found. At a second stage, new matches of unassociated observations are searched for. The candidate frames used for match searching are extracted again as the first-order neighbors of the previous keyframe (KF_{k-1}) in the Covisibility Graph. We only attempt to match observations corresponding to the same visual word as computed by the loop-closure detector, rendering the matching more efficient than brute force. All matches found are checked for consistency before inserting their correspondence as landmarks into the map. For every newly inserted landmark, we set its reference KF to the more recent one used to perform the triangulation. Due to this two-stage correspondence search, our system is capable of running without the need for matches obtained by the VIO system.

In a cleanup step, duplicated landmarks get merged by projecting all landmarks associated in KF_k to the covisible KFs and matches are searched for in the same fashion as for the initial 3D-2D matching. In case different landmarks are associated to one observation, they get merged into one landmark, i.e. the one with most observations associated to it. When there is no landmark associated with an observation, a new correspondence with that landmark is established.

3.4 Factor Graph Formulation

Keyframe-based VI-SLAM can be formulated as a factor graph [75], where the variable nodes θ_j represent the system state, and factor nodes f_i are given by the relation of measurements and the variables (observations). The factor graph defines the factorization of a function $f(\boldsymbol{\Theta})$ as

$$f(\boldsymbol{\Theta}) = \prod_i f_i(\mathcal{A}_i) \,, \tag{5.5}$$

where $\mathcal{A}_i$ represents the set of variable nodes affected by the factor f_i. The goal is to find the values of the variables $\boldsymbol{\Theta}^*$, which maximize the factorization function Equation (5.5). Under the usual assumption that observations are corrupted by zero-mean gaussian noise (gaussian measurement model), the problem can be stated as

$$\begin{aligned}
\boldsymbol{\Theta}^* &= \arg\max_{\boldsymbol{\Theta}} \{f(\boldsymbol{\Theta})\} = \arg\min_{\boldsymbol{\Theta}} \{-\log f(\boldsymbol{\Theta})\} \\
&= \arg\min_{\boldsymbol{\Theta}} \left\{ -\log \prod_i \exp\left(-\frac{1}{2} \|\boldsymbol{z}_i - h_i(\mathcal{A}_i)\|^2_{\boldsymbol{\Sigma}_i} \right) \right\} \\
&= \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_i \|\boldsymbol{z}_i - h_i(\mathcal{A}_i)\|^2_{\boldsymbol{\Sigma}_i} \right\}
\end{aligned} \tag{5.6}$$

$$= \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_i \boldsymbol{e}_i^\intercal \boldsymbol{\Sigma}_i^{-1} \boldsymbol{e}_i \right\} = \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_i \boldsymbol{e}_i^\intercal \boldsymbol{W}_i \boldsymbol{e}_i \right\} ,$$

where $\|\boldsymbol{x}\|^2_{\boldsymbol{\Sigma}} = \boldsymbol{x}^\intercal \boldsymbol{\Sigma}^{-1} \boldsymbol{x}$ denotes the squared Mahalanobis distance, $\boldsymbol{e}_i$ represents the residual error, $\boldsymbol{\Sigma}_i$ the covariance matrix, and $\boldsymbol{W}_i$ the information matrix of the measurement i. In this paper, we use the residual notation to describe the objective function we are looking to minimize. Within VI-SLAM, we essentially use 3 different types of factors, which are introduced below based on the corresponding residual error terms for the factors.

Reprojection Factor. Given the position of a landmark ${}_{S_r}\boldsymbol{l}^j$ expressed in KF_r and the corresponding keypoint observation $\boldsymbol{z}^{k,j}$ in the image coordinates of KF_k, we define the reprojection error as

$$\boldsymbol{e}_r^{k,j} \coloneqq \boldsymbol{z}^{k,j} - h\left(\boldsymbol{K}^k \boldsymbol{T}_{CS} \boldsymbol{T}_{SW}^k \boldsymbol{T}_{WSS_r}^r \boldsymbol{l}^j \right) , \tag{5.7}$$

where $h(\cdot)$ converts homogeneous coordinates into image measurements and $\boldsymbol{K}$ is the camera matrix. Since we use undistorted keypoint coordinates the error function does not contain a distortion model.

IMU pre-integration Factor. Given a set of IMU (accelerometer and gyroscope) readings between two subsequent KFs, we can perform integration of the measurements with an initial estimate of the bias terms as in [42], which later can be optimized without the need to perform numerical re-integration of the raw measuremsents. With a given estimate of the pre-integration, the resulting residuals can be written as

$$\begin{aligned}
\boldsymbol{e}_{\Delta\boldsymbol{R}}^{k-1,k} &= \log\left(\left(\Delta\tilde{\boldsymbol{R}}_{k-1,k}(\bar{\boldsymbol{b}}_g^{k-1}) \exp\left(\frac{\partial \Delta\bar{\boldsymbol{R}}_{k-1,k}}{\partial \boldsymbol{b}_g} \delta\boldsymbol{b}_g \right) \right)^\intercal \boldsymbol{R}_{WS}^{k-1\intercal} \boldsymbol{R}_{WS}^k \right) \\
\boldsymbol{e}_{\Delta\boldsymbol{v}}^{k-1,k} &= \Delta\boldsymbol{R}_{WS}^{k-1\intercal} \left({}_W\boldsymbol{v}^k - {}_W\boldsymbol{v}^{k-1} - {}_W\boldsymbol{g}\Delta t_{k-1,k} \right) \\
&\quad - \left(\Delta\tilde{\boldsymbol{v}}_{k-1,k}(\bar{\boldsymbol{b}}) + \frac{\partial \Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial \boldsymbol{b}_a} \delta\boldsymbol{b}_a + \frac{\partial \Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial \boldsymbol{b}_g} \delta\boldsymbol{b}_g \right) \\
\boldsymbol{e}_{\Delta\boldsymbol{t}}^{k-1,k} &= \boldsymbol{R}_{WS}^{k-1\intercal} \left(\Delta\boldsymbol{t}_{WS}^k - \boldsymbol{t}_{WS}^{k-1} - {}_W\boldsymbol{v}^{k-1}\Delta t_{k-1,k} - \frac{1}{2}\boldsymbol{g}\Delta t_{k-1,k}^2 \right) \\
&\quad - \left(\Delta\tilde{\boldsymbol{t}}_{k-1,k}(\bar{\boldsymbol{b}}^{k-1}) + \frac{\partial \Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial \boldsymbol{b}_a} \delta\boldsymbol{b}_a + \frac{\partial \Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial \boldsymbol{b}_g} \delta\boldsymbol{b}_g \right) ,
\end{aligned} \tag{5.8}$$

where $\tilde{\cdot}$ denotes values obtained by the current estimate of the pre-integration and $\bar{\cdot}$ denotes values obtained with the bias $\bar{\boldsymbol{b}}$ used at the time that the integration was performed. For more detailed explanation of pre-integration the reader is kindly referred to [42]. The scalar $\Delta t_{k-1,k}$ represents the integration time between KF_{k-1} and KF_k. As a result, the residual terms of Equation (5.8) are as follows

$$\boldsymbol{e}_a^{k-1,k} = \left[\boldsymbol{e}_{\Delta\boldsymbol{R}}^{k-1,k\intercal}, \boldsymbol{e}_{\Delta\boldsymbol{v}}^{k-1,k\intercal}, \boldsymbol{e}_{\Delta\boldsymbol{t}}^{k-1,k\intercal} \right]^\intercal . \tag{5.9}$$

Prior Factor. Given prior knowledge of a variable θ at time t_k, the residual for a prior factor

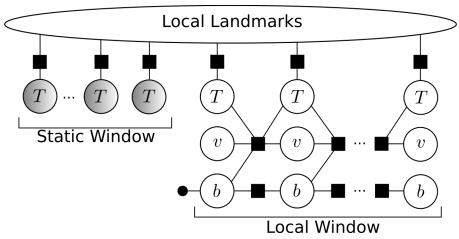


Figure 5.3: Schematic for the local optimization. The state variables participating in the optimization are shown in clear circles, whereas static variables are shaded. The Factors in the optimization (residuals) are shown as squares and a prior on a variable is drawn as a small disc.

is the difference between the prior knowledge $\bar{\theta}^k$ and the estimate θ^k:

$$\boldsymbol{e}_\theta^k = \bar{\theta}^k - \theta^k \ . \tag{5.10}$$

Note that for non-Euclidean variables (i.e. rotations) the minus operation has to be adapted to the commonly used box-minus operator, as shown in [55].

3.5 Local Bundle Adjustment (BA)

Since a full BA quickly becomes computationally infeasible for real-time or close to real-time applications, local optimization is performed frequently as in most KF-based SLAM systems today [96], [128]. While for pure visual SLAM, it is well-established as shown by [96] that selecting the local optimization window based on covisibility is a reasonable choice, the situation in the case of VI-SLAM is different, as a temporal ordering of the keyframes is of crucial importance in order to obtain well defined constraints formed by the IMU cues.

In this work, we employ a strategy similarly to [98], where the the local optimization window of KFs is defined as the set of the N most recent KFs as illustrated in Fig. 5.3. In the multi-agent case, we consider the last N KFs stemming from the same agent as KF_k. In addition to the KFs in the Local Window, KFs that share observations with the Local Landmarks visible in the Local Window are placed as fixed variables in the optimization (Static Window). For landmarks with only two observations, we check the KF within the Local Window whether it is the last one inside the window, in which case the landmark is completely deleted from the map, as it is unlikely to be re-detected. By doing so, landmark culling is performed by design without the need for further bookkeeping of the landmark observations.

Since the optimization of the bias terms is limited to the Local Window, we impose a prior on the N^{th} KF in order to constraint the variation of the bias. Therefore, the objective function for the local BA in terms of residuals can be written as

$$J(\boldsymbol{\Theta}) \coloneqq \boldsymbol{e}_b^{N\intercal} \boldsymbol{W}_b^N \boldsymbol{e}_b^N + \sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{L}(j)} \delta \left(\boldsymbol{e}_r^{k,j\intercal} \boldsymbol{W}_r^{k,j} \boldsymbol{e}_r^{k,j} \right)$$

$$+ \sum_{k-1,k\in\mathcal{V}\setminus\mathcal{V}_s} \boldsymbol{e}_a^{k-1,k\intercal}\boldsymbol{W}_a^{k-1,k}\boldsymbol{e}_a^{k-1,k} \tag{5.11}$$
$$+ \sum_{k-1,k\in\mathcal{V}\setminus\mathcal{V}_s} \boldsymbol{e}_b^{k-1,k\intercal}\boldsymbol{W}_b^{k-1,k}\boldsymbol{e}_b^{k-1,k},$$

where $\delta(\cdot)$ represents a robust cost function – here, the Cauchy loss function. The set of static KFs is denoted as $\mathcal{V}_s$. Optimization is performed using the Levenberg-Marquardt algorithm available in the optimization framework GTSAM[1], while we approximate the information matrix $\boldsymbol{W}_b^{N-1}$ for the bias prior in the next iteration by extracting the diagonal block of the Hessian matrix corresponding to $\boldsymbol{b}^{N-1}$, computed by linearizing Equation (5.11) at the updated state $\boldsymbol{\Theta}$.

3.6 IMU Bias Initialization

While we use the ability of the VIO system to accurately initialize the gravity direction and initial scale, the estimation of the IMU bias terms $\boldsymbol{b}$ is more sensitive, as all axes need to be sufficiently excited and therefore, the initialization is dependent on the movement. While the gyroscope bias usually can be estimated well after a few KFs, the accelerometer bias is more sensitive. When performing Local BA, the bias terms that are outside the Local Window are only re-adjusted following global optimization and therefore, usually at the beginning of a mission they are incorrect. We propose to perform an initial correction using an bundle adjustment triggered based on the uncertainty of the bias estimates. As described in the previous section, we compute the marginal an approximation information matrix of $\boldsymbol{b}^{N-1}$, which gives us an estimate of the uncertainty. As it is safe to assume that the accelerometer bias is problematic, we only look at the part of $\boldsymbol{W}_{\boldsymbol{b}}^{N-1}$ corresponding to $\boldsymbol{b}_a^{N-1}$ and extract

$$w_{min} \coloneqq \sqrt{\min\left(\operatorname{diag}\left\{\boldsymbol{W}_{\boldsymbol{b}_a}^{N-1}\right\}\right)}, \tag{5.12}$$

which approximates the minimal square root information on the accelerometer bias under the assumption that $\boldsymbol{W}_{\boldsymbol{b}}^{N-1}$ is predominant on the diagonal. The global optimization as in Section 3.9 is triggered as soon as w_{min} is above a threshold parameter w_{init}. We do the same procedure for both agents, however, if the second agent only has very few frames in the map, the initialization is postponed until a sufficient number of KFs form this agent are processed in order to avoid unstable results.

3.7 Keyframe Management

While inserting keyframes is a necessity during exploration, insertion of new KFs in a well mapped area is problematic in the sense that the number of error terms in Equation (5.11) grows unbounded causing the optimization to slow down. For purely visual SLAM, it is well established that this can be avoided by dropping KFs (culling) containing predominantely reduntant information [96]. Here we assume a KF to be redundant if more than 90 percent of its landmark

[1] https://research.cc.gatech.edu/borg/gtsam

are observed in at least 3 other KFs as well. When using IMU information, this approach is problematic, as the preintegrated IMU measurements form a weaker constraint the larger the integration time between two consecutive KFs gets. While [98] uses a fixed time-based threshold to limit the integration time between KFs, we propose to utilize the estimated uncertainty of the preintegrated measurement. Since the translational part of the preintegrated measurement is crucial to recover a trajectory of metric scale and is also the most sensitive value due to the double integration of the acceleration, we only use the sub-part of $\mathbf{\Sigma}_a$ corresponding to the translation. We only allow a KF k to be culled, if it is outside the Local Window of both agents and

$$\sigma_{min}^{k-1,k+1} := \sqrt{\min\left(\operatorname{diag}\left\{\mathbf{\Sigma}_a^{k-1,k} + \mathbf{\Sigma}_a^{k,k+1}\right\}\right)} < \sigma_{\text{cull}} \tag{5.13}$$

holds. This results in a more generic threshold as a maximal integration time, since it accounts for the uncertainty of the bias used for the preintegration which changes over the trajectory and furthermore naturally considers the noise of the IMU measurements allowing to use the same threshold for different IMU measurement noise levels (i.e. for different sensors).

3.8 Loop-closure Detection & Frame Localization

In order to be able to correct accumulated drift over larger trajectories when going back to a previously mapped area, the need to recognize visited place arises. As OKVIS is a purely VIO system, it does not have an implementation of loop-closure detection nor correction. As a result, in our implementation we employ the bag of binary words approach [51] together with the appearance and geometric checks used in [96]. In brief, loop-closure candidates are accepted if the similarity score of an older matching keyframe (KF_m) is larger than the minimal similarity of the KFs sharing connections on the Covisibility graph with the current keyframe (KF_k). Once a suitable candidate (KF_l) is found, the KFs are matched via descriptor matching and a projective RANSAC is performed to filter outliers and finally decide upon the inlier observation whether the match is accepted as a loop-closure. In case a match is found, we transform the loop-closuring keyframe (KF_k) and its neighbors into the coordinate frame of the re-detected KF_l and search for additional matches before merging duplicated landmarks as described in Section 3.3. After the merging step, the newly generated correspondences are inserted in the Covisibility Graph and a pose graph optimization followed by Global BA is triggered.

A similar routine is performed to initialize our multi-agent setup. Note that here, we assume that the first agent has already initialized the map and we try to localize any subsequent agent in this map. Since at this stage we do not have any covisibility information from the additional agent, we only start searching based on the descriptor similarity score, which we threshold in order to avoid tedious searching. To verify a candidate for initialization, we solve the P3P problem using [71] together with RANSAC to filter outliers. Once the initialization is performed, we directly associate the observations matched with landmarks and proceed with the normal mapping.

3.9 Global BA

During Global BA, a full optimization of both the structure and the KF states is performed. In our system, this optimization is carried out in three cases; when we detect a loop closure, when we trigger the initialization and also at the end of a mission. In the case of loop-closure, we fist perform an optimization of the Essential Graph as a 6DoF pose graph optimization without optimization of the velocity and bias terms. The second step of the global optimization, the Global BA, is identical for all of the three possible cases.

In the Global BA, we perform a full BA including the estimation of velocity and bias terms for all KFs. In contrast to the Local BA, all states variables are included in the optimization and we do not impose any prior on the bias terms. Therefore, the objective function to be minimize is expressed as

$$\begin{aligned} J(\boldsymbol{\Theta}) \coloneqq & \boldsymbol{e}_p^{0\intercal} \boldsymbol{W}_p^0 \boldsymbol{e}_p^0 + \sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{L}(k)} \delta \left(\boldsymbol{e}_r^{k,j\intercal} \boldsymbol{W}_r^{k,j} \boldsymbol{e}_r^{k,j} \right) \\ & + \sum_{k-1,k \in \mathcal{V}} \boldsymbol{e}_a^{k-1,k\intercal} \boldsymbol{W}_a^{k-1,k} \boldsymbol{e}_a^{k-1,k} \\ & + \sum_{k-1,k \in \mathcal{V}} \boldsymbol{e}_b^{k-1,k\intercal} \boldsymbol{W}_b^{k-1,k} \boldsymbol{e}_b^{k-1,k} \,, \end{aligned} \tag{5.14}$$

where the first term of Equation (5.14) is a prior on the root KF, in order to remove the ambiguity arising by the choice of the reference coordinate system. Again, a Cauchy loss function is used on the reprojection terms. Since Global BA is only performed after a local optimization, we can expect only a very limited number of outliers, therefore we carry out the full optimization using the Levenberg-Marquardt algorithm and only perform an outlier removal afterwards.

4 Experimental Results

4.1 Experimental Setup

In order to evaluate the proposed system, we perform experiments on the publicly available EuRoC dataset [11], consisting of different sequences recorded from a UAV flying different trajectories both in a smaller room (Vicon Room) as well as in a larger industrial environment (Machine-Hall), where we put our focus on the Machine-Hall sequences. This dataset is specifically selected to enable a direct and fair comparison of the proposed pipeline to the most relevant state of the art system in VI-SLAM, namely VI-ORB-SLAM [98], as it was evaluated on this dataset and it is closed source. In order to conduct experiments in a multi-UAV setup, we run two different sequences from this dataset simultaneously, while treating each sequence as coming from a separate UAV.

Since the proposed system aims to achieve a globally consistent map and we only optimize KFs, we choose the Absolute Trajectory Error (ATE) as our evaluation metric for comparison. Assuming an estimated trajectory of n KF poses $\boldsymbol{T}_{WS}^{1:n}$ and the corresponding trajectory in the ground truth $\boldsymbol{T}_{GS}^{1:n}$, where G is the origin of the ground truth poses, we can compute $\boldsymbol{T}_{GW}$ to transform the estimated trajectory into the origin of the ground truth, e.g. by using the method

	VI-ORB-SLAM			Proposed			
	RMSE [m]	Scale Err. [%]	RMSE* [m]	RMSE [m]	Scale Err. [%]	RMSE* [m]	KF-rate [Hz]
V1_01_easy	**0.023**	0.8	0.016	0.044	1.5	0.034	7.4
V1_02_medium	0.027	1.0	0.019	**0.021**	1.0	0.012	9.7
V1_03_difficult	X	X	X	**0.046**	2.0	0.034	10.0
MH_01_easy	0.068	0.3	0.068	**0.018**	0.2	0.015	6.5
MH_02_easy	0.073	0.4	0.072	**0.027**	0.4	0.020	6.5
MH_03_medium	0.071	0.1	0.071	**0.031**	0.2	0.030	6.3
MH_04_difficult	**0.087**	0.9	0.066	0.089	0.1	0.089	8.4
MH_05_difficult	**0.060**	0.2	0.060	0.070	0.5	0.054	8.1

Table 5.1: The scale and RMSE errors of VI-ORB-SLAM [98] and the proposed monocular-inertial pipeline evaluated on the EuRoC dataset (averaged over 3 runs). The best RMSE performance in each sequence is indicated in bold. RMSE* records the error when performing the alignment to the ground-truth trajectory using a 7DoF transformation, indicating the error that would be achieved with perfect scale estimation.

of Horn [59]. The error is computed as the Root Mean Squared Error (RMSE) of the translation ($trans(\cdot)$) for all poses as

$$RMSE(\boldsymbol{T}_{WS}^{1:n}) \coloneqq \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|trans\left((\boldsymbol{T}_{GS}^{i})^{-1}\boldsymbol{T}_{GW}\boldsymbol{T}_{WS}^{i}\right)\|^2}\,. \tag{5.15}$$

The evaluation of our system was performed on an Intel Core i7-4710MQ running at 2.5 GHz with 16GB RAM.

4.2 Results

We first evaluate the system in a single UAV configuration and compare against VI-ORB-SLAM as shown in Table 5.1. Note that the values for VI-ORB-SLAM are copied from [98] for reference, as there is no open-source implementation of this method. For our approach we report the mean value over three runs. In the smaller Vicon-Room sequences, the proposed system generally has a higher error level compared to VI-ORB-SLAM, although for the sequence *V1_02_medium* we perform slightly better. We attain this to the low-textured scene of this sequence, in which a tight coupling between front-end and back-end as employed by VI-ORB-SLAM is advantageous, enabling reaction to a low number of matches, e.g. triggering the detection of additional, weaker keypoints.

On the MH sequences, we are able to achieve over 50% reduction on the trajectory error compared to VI-ORB-SLAM for the well-textured sequences *MH_01- MH_03*. On *MH_04* and *MH_05*, which exhibit partially very bad illumination, we perform comparably to VI-ORB-SLAM with marginally bigger errors. Evidently, the proposed system achieves higher accuracy in feature rich sequences (i.e. well-textured scenes with sufficient illumination), which we attain to the following points. Compared to [98], we generally create fewer landmarks, allowing the

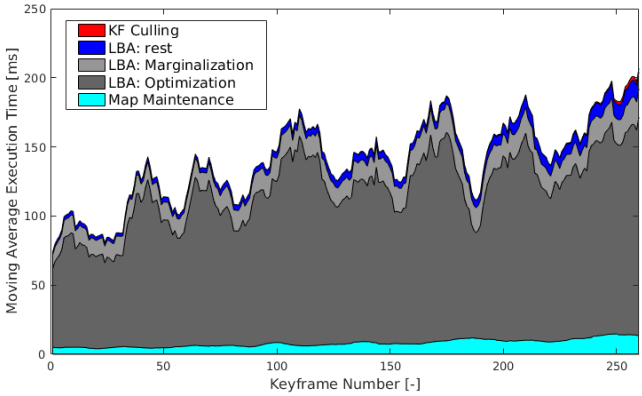


Figure 5.4: Breakdown of the computation time of the proposed pipeline performed for every KF for the sequence *MH_03_medium*. To filter fluctuations between KFs,the timings presented are computed using a moving average filter over 10 KFs. The average incoming KF-rate for this sequence is approximately 4Hz (250 ms).

inclusion of more KFs in our Local Window, therefore increasing the scope of the Local BA. Furthermore, the use of the inverse distance parametrization together with the local reference keyframe formulation generally results in a better conditioned optimization problem for larger trajectories. Additionally the inclusion of a soft prior in the local optimization results allows to adjust the bias terms more freely in the course of the local optimization, damping the diffusion of initial errors over the whole trajectory.

The fluctuations of the error, however, across the different sequences attest to the fact that the front-end is a crucial component in handling difficult scenarios e.g. with bad illumination and low-textured scenes. So while proposing a powerful back-end is shown to improve significantly the accuracy of the estimation processes, further investigation in interfacing it with the front-end promises to result to even further improvement. This is most evident for visually challenging Vicon-Room sequences, for which reason our evaluation was focused on the Machine Hall.

Evaluation of the complexity of the proposed system and the resulting timings is not straightforward, as real-time performance here depends on the rate at which KFs are processed rather than a fixed frame rate, therefore, we analyze complexity using the average KF-rate, as shown in Table 5.1. Note that the KF-rate is both scene- and motion-dependent, and thus, it varies both across sequences and throughout one sequence. As a result, we consider the system to be real-time capable, if it is able to process the KFs faster than the average KF-rate. The average KF processing rate for each sequence is shown in Table 5.1 with the system achieving real-time capability across all sequences. A detailed breakdown of the execution time for *MH_03_medium* is shown in Fig. 5.4. Since the execution time has relatively large fluctuations between KFs, we process the timings using a moving average filter. As it can be seen, the runtime is slowly

UAV A:	*MH_01*	*MH_03*	*MH_04*
UAV B:	*MH_02*	*MH_02*	*MH_05*
RMSE [m]	0.021	0.026	0.059
Scale Err. [%]	0.3	0.05	0.1
RMSE* [m]	0.015	0.026	0.59

Table 5.2: Average ATE for the proposed pipeline in the two-UAV setup. Different combinations of sequences are used to conduct experiments of different levels of difficulty. In Fig. 5.1, the map as obtained by *MH_02* & *MH_03* is shown

increasing with a growing number of KFs, which is caused by the need for well defined IMU-constraints, prohibiting arbitrary KF culling and therefore, although the number of variables is approximately constant, the number of error terms contributing to the cost function increases (Static Window). The fluctuation within the sequence is attained to the fact that exploration generally tends to be cheaper, as the number of KFs in the Static Window decreases. Although the Global BA is the most expensive part of the system (included in the recorded average KF-processing rate), it is only sporadically triggered and therefore, the bottleneck for real-time operation is, on average, the Local BA including the computation of the prior information for the bias.

Evaluation using two UAVs was performed by combining different MH sequences. The trajectory error was computed by aligning the joint map to the ground-truth in the same fashion as for the single UAV setup. An overview of the results is in Table 5.2, whereas the trajectory and landmarks for the combination *MH_02* & *MH_03* is shown in Fig. 5.1.

Although there are no IMU measurements between KFs from different UAVs to impose further constraints, it can be seen that the overall accuracy is maintained or increases compared to the single UAV case, indicating global consistency of the two trajectories in the common map frame. The advantages of collaborative sensing from two UAVs become evident especially in the difficult sequences. However, at this stage we are only able to process the data close to real-time (factor of ~1.5), due to our sequential setup.

5 Conclusion

This work presents a back-end to monocular-inertial odometry from one or multiple agents, contributing towards achieving globally consistent SLAM, while resolving the scale ambiguity. The system considers the state of the bias estimate in both a local optimization and during the keyframe culling and is real-time capable in the single agent case. An evaluation on the EuRoC benchmarking dataset reveals over 50% improvement in accuracy at times over the state of the art. Finally, this system is demonstrated to achieve globally consistent collaborative VI mapping from two UAVs.

The significant reduction of the trajectory error in some of the test cases reveals the room for improvement still existing on the state of the art. However, the reported fluctuations emphasize the need for a tight integration between front-end and back-end in order to allow appropriate reactions to difficult conditions, such as low-textured scenes or bad illumination. Future work will

aim at addressing this integration into the proposed system. Furthermore, appropriate methods to summarize IMU-constraints in order to expand the horizon for keyframe culling are essential towards the goal of life-long real-time SLAM.

Paper III

CVI-SLAM – Collaborative Visual-Inertial SLAM

Marco Karrer, Patrik Schmuck and Margarita Chli

Abstract

With robotic perception constituting the biggest impediment before robots are ready for employment in real missions, the promise of more efficient and robust robotic perception in multi-agent, collaborative missions can have a great impact many robotic applications. Employing a ubiquitous and well-established visual-inertial setup onboard each agent, in this paper we propose CVI-SLAM, a novel visual-inertial framework for centralized collaborative SLAM. Sharing all information with a central server, each agent outsources computationally expensive tasks, such as global map optimization to relieve onboard resources and passes on measurements to other participating agents, while running visual-inertial odometry onboard to ensure autonomy throughout the mission. Thoroughly analyzing CVI-SLAM, we attest to its accuracy and the improvements arising from collaboration, and evaluate its scalability in the number of participating agents and applicability in terms of network requirements.

Published in:
IEEE Robotics and Automation Letters (RA-L), 2018
DOI: 10.1109/LRA.2018.2837226

1 Introduction

With state-of-the-art Simultaneous Localization And Mapping (SLAM) systems having reached substantial robustness and accuracy in the centimeter range [96] for single-robot applications, multi-robot systems have been gaining growing popularity in numerous scenarios, ranging from search-and-rescue applications to digitization of archeological sites. Increasing the robustness of the system by sharing information amongst the participants, boosting the efficiency of a mission by dividing up a task or enabling tasks otherwise impossible for a single robot are only some of the advantages a team of robots has to offer. At the same time, multi-robot scenarios pose significant challenges, such as dealing with network characteristics (e.g. time delays and bandwidth) and ensuring transparent and consistent information access among all agents. In

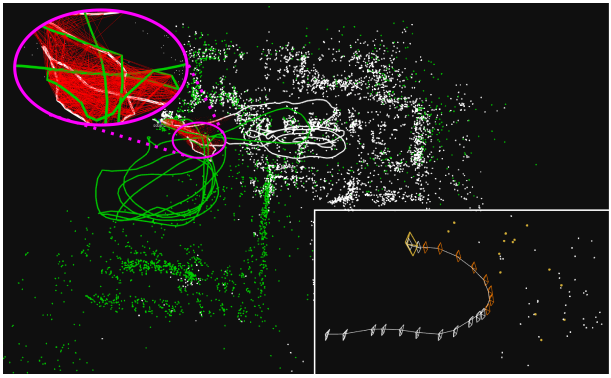


Figure 6.1: A snapshot of CVI-SLAM with two agents collaboratively building the map. Trajectories and map points are colored in white and green for agent 1 and 2, respectively. Covisibility constraints across different agents are indicated in red. The inlet on the bottom right depicts the limited local map of agent 1 with the newest frame and its observed map points colored in yellow, and keyframes which received pose updates from the server in orange.

this spirit, this paper proposes CVI-SLAM, a centralized ***C***ollaborative SLAM system for multiple robotic agents, each equipped with a Visual-Inertial (VI) sensor suite, and a central ground station, the "server". Fig. 6.1 shows a snapshot of our proposed system. Taking the powerful monocular setup of [119] a step further towards real deployment, CVI-SLAM agents employ a visual-inertial sensor suite, enabling metric scale estimation and gravity alignment of the estimated trajectories and maps, which is necessary for autonomous exploration of an environment, guaranteeing higher accuracy and robustness compared to monocular SLAM [83], [98], due to the complimentary nature of the sensors. Similarly to [119], here information is consequently shared amongst all agents. However, incorporating an Inertial Measurement Unit (IMU) increases the complexity of the system, since more data has to be processed, and sharing and removing of data becomes more difficult, since IMU constraints depend on the relative timings

between measurements. Furthermore, camera images and IMU measurements need to be synchronized, varying IMU bias terms need to be constantly estimated, and for accurate results, the manifold structure of the rotation group $SO(3)$ needs to be addressed. Therefore, we use a new visual-inertial odometry system designed for CVI-SLAM, implementing on-manifold pre-integration of IMU measurements [42], having shown its beneficial characteristics for IMU handling already in other SLAM systems [98], [111]. We thoroughly evaluate CVI-SLAM on a public dataset in scenarios with one to four agents on aerial platforms, as some of the most challenging platform for robotic perception (due to their high agility and constrained resources). Our analysis discusses the bandwidth requirements for CVI-SLAM and its scalability to the number of agents. Attesting to the accuracy of CVI-SLAM and improved performance from sharing data, we compare the collaborative trajectory estimates against ground truth, exhibiting equivalent performance to other state-of-the-art SLAM systems in single agent scenarios, while outperforming these system in the multi-agent applications.

2 Related Work

While several works in the literature deal with either collaborative localization [5], [108], [34] or collaborative mapping [138], [53], [14], only few existing works are able to perform collaborative SLAM with multiple agents. Eliminating the need of a pre-computed map (as in collaborative localization) or known robot poses (required in collaborative mapping), collaborative SLAM promises to exploit the full spectrum of possibilities for robot collaboration in far more generic and realistic setups.

A centralized architecture for robotic collaboration is usually employed in the literature when it comes to systems applied to practical scenarios. However, some works tackle collaborative SLAM in a decentralized manner, such as [27], proposing a fully distributed SLAM system evaluated in simulation, emulating a sensor setup with visual, inertial and Global Positioning System (GPS) sensors. An efficient place recognizer distributed amongst all agents with real data, but in simulation was shown in [22]. Most recently, Choudhary et al. [19] showed a decentralized SLAM system where co-localization of the participating robots is based on commonly observed pre-trained objects.

Guaranteeing data consistency and avoiding double-counting are the biggest challenges in a decentralized setup, whereas a centralized system has a clearer allocation of information. Furthermore, a centralized client-server-architecture allows agents to outsource non time-critical, but computationally expensive algorithms, such as global map optimization, to the server, which is potentially much more powerful. This allows an agent to allocate its potentially limited onboard resources to the most critical tasks, such as visual odometry.

Probably the most powerful vision-only collaborative SLAM system is CoSLAM [149], which has the ability to handle dynamic environments, albeit at high computational cost, requiring GPUs, often prohibitive in resource-constraint robots. Receiving image data from multiple monocular cameras as input, CoSLAM groups cameras with scene overlap, relying on the assumptions that all cameras are synchronized and observe the same scene at initialization. Together with the requirement for a GPU, these assumptions render CoSLAM impractical to run online onboard multiple robots.

In an earlier attempt to multi-robot collaboration, [43] extended a structure from motion pipeline

to collaborative SLAM for Unmanned Aerial Vehicle (UAV), with each agent running a keyframe-based visual odometry sending all keyframes to a central server. The server would search for overlap across maps and merge them if necessary. While impressive, this system fell short of sending any feedback from the server back to the agents and therefore, cannot profit from optimization results and data from other agents.
In a system more suitable for multi-robot collaboration, C^2TAM [114] proposed to perform position tracking onboard each agent, while all mapping tasks are run on the server. The server then sends the complete map to each agent for further tracking steps, enabling operation with agents with very limited computational resources. However, C^2TAM's assumption that an agent is always in communication with the server heavily restricts the agent's autonomy, while the assumption of being able to repeatedly send the whole map to an agent further restricts C^2TAM's practicality and generality.
Targeting multi-device mapping applications with hand-held devices, MOARSLAM [94] proposed to employ a server to act as central memory for storing and distributing data amongst agents, with each agent executing all parts of a full SLAM system (i.e. visual odometry, place recognition and global map optimization) onboard. While employing a visual-inertial setup, MOARSLAM only uses the IMU as one of two alternatives for scale disambiguation during pose estimation from visual odometry (with stereo images being the second option). The server back-end presented by Deutsch et al. [33] can be run on a server to combine different SLAM systems in a collaborative framework. Shifting all intelligence and computation to the agents, [94] and [33] do not exploit the centralized architecture to its full potential, restricting its employment with powerful agents. Instead, CVI-SLAM outsources tasks that are computationally too expensive for the resource-limited agents to make full use of the potential of a centralized collaborative architecture, while ensuring that all tasks critical to the autonomy of each agent are still run onboard, as opposed to [114]. In contrast to systems sending no [43] or only partial [33], [93] feedback to the agents, CVI-SLAM consequently promotes full transparency of information.
Proposing collaborative monocular SLAM, [119] showed a proof of concept of a powerful centralized architecture suitable for resource-constraint platforms. Inspired by the extent of collaboration that this architecture can enable, in this work we propose a system to fuse visual and inertial information from each agent to a globally consistent map that can be reused in full or in parts by each agent. Adding to the challenge of consistent sharing of data and efficient handling of multiple agents, CVI-SLAM enables collaborative SLAM estimation with metric scale and high accuracy, and boosts the robustness and scalability of the system employing redundancy detection and removal at global scope.

3 Preliminaries

3.1 Notation

For the denotion of vectors we use bold small letters (e.g. $\boldsymbol{a}$), while matrices are denoted by bold capital letters (e.g. $\boldsymbol{A}$). To distinguish different coordinate frames we use capital letters (e.g. A). As a result, a vector expressed in A is denoted by ${}_A\boldsymbol{x}$. For the coordinate frames, S denotes the IMU body frame, C the camera coordinate system and W for the origin (i.e. the

inertial frame). The summarization of sets of variables is denoted by calligraphic letters (e.g. $\mathcal{A}$). To denote a rigid body transformation from coordinate frame B into A we use the notation $\boldsymbol{T}_{AB}$, where the rotational and the translational part of $\boldsymbol{T}$ are denoted by $\boldsymbol{R}$ and $\boldsymbol{t}$, respectively.

3.2 IMU Model and System States

In this paper, we use the standard IMU measurement model, assuming that measurements from both the accelerometer ${}_S\boldsymbol{a}(t)$ and the gyroscope ${}_S\boldsymbol{\omega}_{WS}(t)$ are corrupted by additive white noise $\boldsymbol{\eta}$ and have an unknown, time varying sensor bias $\boldsymbol{b}$:

$$ {}_S\boldsymbol{a}(t) = \boldsymbol{R}_{WS}^{\mathsf{T}}(t)\left({}_W\hat{\boldsymbol{a}}(t) - {}_W\boldsymbol{g}\right) + \boldsymbol{b}_a(t) + \boldsymbol{\eta}_a(t)\,, \tag{6.1} $$

$$ {}_S\boldsymbol{\omega}_{WS}(t) = {}_S\hat{\boldsymbol{\omega}}_{WS}(t) + \boldsymbol{b}_g(t) + \boldsymbol{\eta}_g(t)\,. \tag{6.2} $$

The true values of the respective variables are indicated by $\hat{\cdot}$, while ${}_W\boldsymbol{g}$ denotes the gravity vector. In order to account for these characteristics of the IMU measurements, the system state, denoted by $\boldsymbol{\Theta}$, besides the Keyframe (KF) poses $\{\boldsymbol{R}_{WS}, \boldsymbol{t}_{WS}\}$ and Map Point (MP) positions ${}_W\boldsymbol{l}$ also includes the linear velocities ${}_W\boldsymbol{v}$ and bias terms $\boldsymbol{b}$:

$$ \boldsymbol{\Theta} \coloneqq \{\underbrace{\boldsymbol{R}_{WS}^k, \boldsymbol{t}_{WS}^k, {}_W\boldsymbol{v}^k, \boldsymbol{b}^k}_{\mathrm{KF}_k}, {}_{S_r}\boldsymbol{l}^i\} \quad \forall k \in \mathcal{V}, \forall i \in \mathcal{L}\,, \tag{6.3} $$

where the set of all KFs and all MPs are denoted by $\mathcal{V}$ and $\mathcal{L}$, respectively. In this paper, we denote individual state variables as θ_j when appropriate. As proposed in [7], instead of expressing the MPs in a global reference frame, we express them in coordinates of a reference KF S_r. While in our system we employ the inverse depth parameterization for the MPs, for the sake of readability in the following we treat the MPs as if they were expressed in Euclidean coordinates.

4 Methodology

4.1 System Architecture

CVI-SLAM uses the basic system architecture shown in Fig. 6.2, which was first introduced in [119]. Here, however, each agent runs the *visual-inertial odometry* (VIO) system discussed in Sec. 4.3 onboard, and the messages and processes are adapted to handling visual and inertial data. VIO estimates the local trajectory of each robot and maintains a *local map* limited to a fixed number of N KFs of the immediate surroundings of the robot. Furthermore, a *communication interface* on each agent module serves as the interface from and to the server. The server can communicate with all agents, coordinating the exchange of information amongst them. The server maintains maps of unbounded size (*server maps*) on a *server map stack*, holding all data that was collected by all agents throughout the mission. Starting with one server map for each agent, the server merges maps throughout the mission to associate the data from the individual agents. The *agent handlers* on the server side, one for each individual agent, distribute the information from their corresponding agent to the correct modules on the server. The server runs

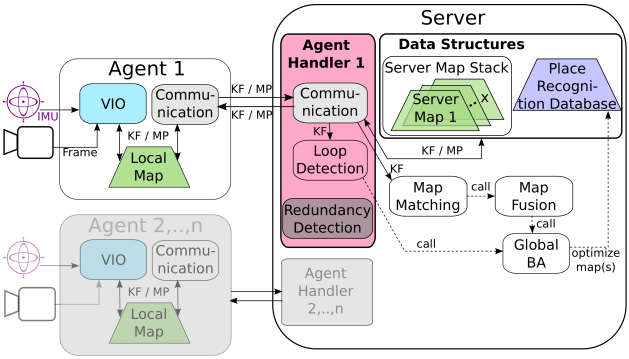


Figure 6.2: Overview of the CVI-SLAM's system architecture. An agent runs real-time VIO maintaining a *local map*. A *communication interface* is used to exchange *keyframes* (KF) and *map points* (MP) between the agent and the server. The server is a powerful computer that runs computationally expensive and non time-critical tasks: *redundancy detection*, *loop detection*, *map fusion*, and *bundle adjustment* (BA).

two place recognition modules, a *loop detection* module to detect loop closure inside one map, and a *map matching* module to detect overlap between distinct maps, allowing to merge these individual maps. A *keyframe database* implements an efficient look-up procedure allowing for a new KF to query other KFs, using an inverted file index. A successful query of the place recognition modules is then followed by a global optimization step (aka *Bundle Adjustment* (BA)). All optimization schemes are implemented using the Ceres[1] solver. An in-depth discussion of the system architecture can be found in [119]. In addition to the handling of IMU data, here we use an extended version of this architecture employing the *redundancy detection* module discussed in Section 4.5.

4.2 Error Residuals Formulation

The problem of KF-based VI-SLAM can be expressed as a nonlinear weighted least-squares optimization, where the state variables θ_j are optimized w.r.t. to some observation measurements $\boldsymbol{z}_i$. With the definition of a residual error as:

$$\boldsymbol{e}_i \coloneqq \boldsymbol{z}_i - h_i(\mathcal{A}_i)\,, \tag{6.4}$$

where $\mathcal{A}_i$ denotes all variables θ_j involved in measurement $\boldsymbol{z}_i$ and $h_i(\cdot)$ is a function that predicts the measurement according to the current state. Using this notation, the objective that

[1] http://ceres-solver.org

is minimized can be written as:

$$\begin{aligned}\boldsymbol{\Theta}^* &= \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_i \| \boldsymbol{z}_i - h_i(\mathcal{A}_i) \|^2_{\boldsymbol{W}_i} \right\} \\ &= \arg\min_{\boldsymbol{\Theta}} \left\{ \sum_i \boldsymbol{e}_i^\intercal \boldsymbol{W}_i \boldsymbol{e}_i \right\} ,\end{aligned} \tag{6.5}$$

where $\|\boldsymbol{x}\|^2_{\boldsymbol{W}} = \boldsymbol{x}^\intercal \boldsymbol{W} \boldsymbol{x}$ denotes the squared Mahalanobis distance in information form. Within our VI-SLAM system, we use essentially three different types of residuals:

Reprojection Residual

With a given MP position ${}_{S_r}\boldsymbol{l}^j$ expressed in the reference KF's IMU frame S_r, the KF poses for both KF_k and KF_r and the keypoint observation $\boldsymbol{z}^{k,j}$ in the image of KF_k, we define the reprojection residual as:

$$\boldsymbol{e}_r^{k,j} := \boldsymbol{z}^{k,j} - g\left(\boldsymbol{K}^k \boldsymbol{T}_{CS} \boldsymbol{T}_{SW}^k \boldsymbol{T}_{WS_r}^r {}_{S_r}\boldsymbol{l}^j \right) , \tag{6.6}$$

where $g(\cdot)$ is a function to convert homogeneous into image coordinates and $\boldsymbol{K}^k$ represents the camera intrinsics matrix. Note that in this paper we use undistorted keypoints, hence, Equation (6.6) does not contain a distortion model.

IMU pre-integration Residual

With a given sequence of IMU readings between two consecutive KFs, both from the accelerometer and the gyroscope, integration of the measurements can be performed in order to obtain a relative constraint between the KFs. In order to avoid re-integration of the measurements upon changes in the bias terms, [42] presented a method allowing to perform the integration only once and apply linearized corrections considering changes of the biases after the integration. With a given pre-integration, the resulting residuals can be written as

$$\begin{aligned}\boldsymbol{e}_{\Delta \boldsymbol{R}}^{k-1,k} &= \log\left(\left(\Delta\tilde{\boldsymbol{R}}_{k-1,k}(\bar{\boldsymbol{b}}_g^{k-1}) \exp\left(\frac{\partial \Delta\bar{\boldsymbol{R}}_{k-1,k}}{\partial \boldsymbol{b}_g} \delta\boldsymbol{b}_g \right) \right)^\intercal \boldsymbol{R}_{WS}^{k-1\intercal} \boldsymbol{R}_{WS}^k \right) \\ \boldsymbol{e}_{\Delta \boldsymbol{v}}^{k-1,k} &= \Delta\boldsymbol{R}_{WS}^{k-1\intercal} \left({}_W\boldsymbol{v}^k - {}_W\boldsymbol{v}^{k-1} - {}_W\boldsymbol{g}\Delta t_{k-1,k} \right) \\ &\quad - \left(\Delta\tilde{\boldsymbol{v}}_{k-1,k}(\bar{\boldsymbol{b}}) + \frac{\partial \Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial \boldsymbol{b}_a} \delta\boldsymbol{b}_a + \frac{\partial \Delta\bar{\boldsymbol{v}}_{k-1,k}}{\partial \boldsymbol{b}_g} \right) \\ \boldsymbol{e}_{\Delta \boldsymbol{t}}^{k-1,k} &= \boldsymbol{R}_{WS}^{k-1\intercal} \left(\Delta\boldsymbol{t}_{WS}^k - \boldsymbol{t}_{WS}^{k-1} - {}_W\boldsymbol{v}^{k-1}\Delta t_{k-1,k} - \frac{1}{2}\boldsymbol{g}\Delta t_{k-1,k}^2 \right) \\ &\quad - \left(\Delta\tilde{\boldsymbol{t}}_{k-1,k}(\bar{\boldsymbol{b}}^{k-1}) + \frac{\partial \Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial \boldsymbol{b}_a} \delta\boldsymbol{b}_a + \frac{\partial \Delta\bar{\boldsymbol{t}}_{k-1,k}}{\partial \boldsymbol{b}_g} \delta\boldsymbol{b}_g \right) ,\end{aligned} \tag{6.7}$$

where values denoted by $\bar{\cdot}$ are obtained with the bias estimate $\bar{\boldsymbol{b}}$ at the time of the pre-integration

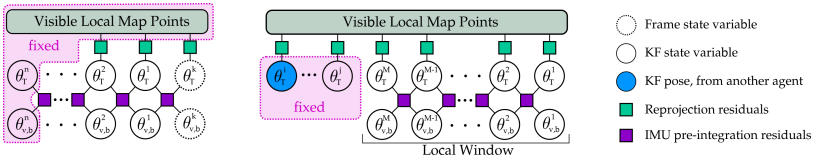


Figure 6.3: Schematics depicting the variables involved in the local optimization during frame tracking (left) and the *Local Bundle Adjustment* (LBA) with a *local window* of M KFs running onboard the agent (right). While the tracking considers only the most recent n KFs stemming from the same agent, the LBA can also include KFs from other agents.

and $\tilde{\cdot}$ denotes values using the current estimate of the state variables. The scalar value $\Delta t_{k-1,k}$ represents the time interval between the two KFs. For a detailed explanation of the used pre-integration method, we kindly refer the reader to [42]. Using the individual parts in Equation (6.7), we define the residual vector for the IMU pre-integration as

$$\boldsymbol{e}_a^{k-1,k} = \left[\boldsymbol{e}_{\Delta \boldsymbol{R}}^{k-1,k\intercal}, \boldsymbol{e}_{\Delta \boldsymbol{v}}^{k-1,k\intercal}, \boldsymbol{e}_{\Delta \boldsymbol{t}}^{k-1,k\intercal}\right]^\intercal . \tag{6.8}$$

Prior Residual

In order to use prior knowledge $\bar{\theta}^k$ about a variable θ at time instance t_k, we define the prior residual as

$$\boldsymbol{e}_\theta^k := \bar{\theta}^k - \theta^k . \tag{6.9}$$

Here, for non-Euclidean variables, such as rotations or bearing vectors, the minus operation needs to be adapted to the widely used box-minus operator.

4.3 Visual-Inertial Odometry

Frame Tracking

For every incoming Frame F, we extract a set of ORB features [116] and perform integration of the IMU readings queued since the last frame. Using the integrated IMU measurements, we perform a prediction of the pose for the current frame to carry out a guided correspondence search by projecting the agent's MPs in the current frame and matching the associated descriptors. Using the established correspondences, we perform an alignment step by minimizing the reprojection error against the observed MPs followed by a second correspondence search. In order to obtain a smooth trajectory and an accurate velocity estimate for the current frame, we execute a motion-only BA in which we optimize the KF poses and the IMU states of the most recent n KFs together with the current frame as shown in Fig. 6.3. After the optimization, we check whether to insert a new KF in the agent's map. A new KF is inserted when one of the three following conditions is met:

(a) more than 20 frames have passed since the last KF was created,

(b) the area where 2D keypoints are found covers less than 40% of the image, or

(c) less than 15 MPs are observed by the current frame.

Criterion (a) enforces temporal constraints between successive KFs, in order to avoid weak IMU constraints due to the accumulated uncertainty from the integration. Criteria (b) and (c) ensure sufficient overlap between the KFs, while adaptively inserting more KFs during challenging motions such as fast rotations.

Local Map

The *local map* holds the KFs and MPs in the surroundings of the current position. It is bounded to the N closest KFs in the vicinity of the agent, with N primarily depending on the available computational onboard resources of the agent. The KFs in the local map are both connected by the IMU constraints between consecutive KFs as well as covisibility constraints derived from common observations of MPs. In CVI-SLAM, two KFs are considered covisible if they share at least 15 MPs. As indicated in Fig. 6.3, we keep a constant number M, smaller than N, of consecutive KFs in a *local window* inside the *local map*, in order to ensure well defined IMU constraints. Note that while the temporal KFs must come from the same agent as the local map, the covisible KFs are purely defined by their covisibility, regardless of which agent created the KF. This allows the local odometry to benefit from experiences of other agents and improve the accuracy of the estimation in collaboration.

Local Mapping

The *local mapping* of our VIO runs in a separate thread and is responsible for maintaining and optimizing the *local map* and is triggered every time the *tracking* inserts a new KF. For map maintenance, we employ a scheme inspired by [96], in which MPs with insufficient observations in the tracking are culled. After culling we triangulate new MPs between the local KFs and merge them with existing MPs considering their vicinity and their associated ORB descriptors. In order to improve the accuracy and consistency of the *local map*, we perform a *Local Bundle Adjustment* (LBA) step. The scope of the LBA is defined by the most recent M KFs in the *local window*, and all MPs observed in those KFs. Additionally we insert all KFs that have common observations with the KFs in the *local window*, as illustrated in Fig. 6.3. These KFs outside the *local window* are inserted with their pose fixed and serve as anchors to stabilize the optimization. Therefore, we can formulate the objective of the LBA as follows:

$$J(\boldsymbol{\Theta}) \coloneqq \sum_{k \in \mathcal{V}} \sum_{j \in \mathcal{L}(k)} \delta\left(\| \boldsymbol{e}_r^{k,j} \|^2_{\boldsymbol{W}_r^{k,j}} \right) + \sum_{k-1,k \in \mathcal{V} \setminus \mathcal{V}_s} \left(\| \boldsymbol{e}_a^{k-1,k} \|^2_{\boldsymbol{W}_a^{k-1,k}} + \| \boldsymbol{e}_b^{k-1,k} \|^2_{\boldsymbol{W}_b^{k-1,k}} \right), \tag{6.10}$$

where $\delta(\cdot)$ is a robust cost function, here the Cauchy loss, aiming to reduce the influence of outliers, $\mathcal{L}(k)$ are the MPs observed by KF k, and $\boldsymbol{e}_b^{k-1,k}$ penalizes changes in the IMU bias

for successive KFs. The set $\mathcal{V}_s$, denoting all KFs inserted with a fixed pose, can include KFs created by other agents.

IMU Initialization

Since the IMU biases, as well as the gravity direction are unknown, a dedicated initialization is necessary in order to perform the tracking as described above. In this work, we employ the strategy proposed in [111], which performs the initialization in three steps. In the initial phase, the vision-only tracking as in [119] is employed, while performing the IMU pre-integration with all bias variables set to zero. To avoid large integration windows between the KFs, we restrict the number of frames between KFs to 3. After reaching 15 KFs, the visual structure is bundle adjusted and the gyroscope bias is initialized in a linear least squares fashion. Afterwards, the scale of the visual structure, the velocities in S and gravity direction are linearly estimated followed by a refinement of the gravity direction. Upon successful computation of all steps, the visual structure is scaled and aligned with the gravity direction, while the body velocities are rotated in the world frame W. If the initialization fails up to a window of 20 KFs, we re-initialize the visual-only tracking and start again.

4.4 Communication

The *communication modules* on the agents' and the server's sides act as the interface between both sides that serializes and de-serializes data that is shared between them. Using this interface, each agent informs the server about changes in its *local map*, i.e. any added or changed KFs and MPs. Constantly sending the whole data structure for KFs and MPs (including e.g. 2D feature keypoints extracted from the image) would result in high network traffic, with messages of around 300 Bytes for MPs and 29 KB for KFs (for our setup with 500 ORB features per KF). Therefore, after sending a KF or a MP once, for all following changes an update message is sent, having only a size of 184 Bytes for KFs and 52 Bytes for MPs. Messages from the server to the agent contain the K KFs with the strongest covisibility to the current position of the agent, to augment the agent's *local map* with past data and KFs and MPs from other agents. Using these messages, we also transmit IMU related information (IMU measurements and current bias and velocity estimates) from the agents to the server. From the KF messages received from the agent, we reconstruct the initial pre-integration factors created by VIO for incorporation in global map optimiation (Section 4.6). From the agent to the server, we do not include the pre-integration factors, since IMU data is not considered for KFs outside the *local window*. However, we include the most current velocity and bias estimates for KFs to update this data on the agent with refined results from BA. In case of a loss of connection to the server, an agent will not be able to exchange data with the server any more, and fall back to a VIO system with limited *local map* size N.

4.5 Redundancy Detection and Removal

When one or more agents visit the same location multiple times during a mission, the map contains several KFs from the same distinct place. Some of these KFs encode almost the same information, if the measurement was taken from a similar viewpoint. Since the size of the map

affects the timings of most processes on the server, such as place recognition, map queries or BA, it is desirable to remove these redundancies to boost the scalability of the system. Our rejection scheme randomly selects a KF from the map, and compares it to its neighbors in the covisibility graph. If the number of commonly observed landmarks with all neighbors is higher than a pre-defined threshold, the KF is considered redundant. Furthermore, for consecutive KFs connected by IMU constraints, we allow a maximum time of 2s between two KFs, to bound inaccuracies from IMU integration for optimization.

4.6 Loop & Map Fusion

To detect repeatedly visited locations inside one *server map* (*loop closure*) and separated maps (*map matching*) on our *server map stack*, we employ a multi-stage place recognition system. Firstly, for a query Keyframe KF_q, we select a subset of possibly matching candidates $\mathcal{C}$ from all map data using a bag-of-words approach [51], followed by a brute force descriptor matching of KF_q against all KFs in $\mathcal{C}$. Upon successful matching of KF_c, we employ a projective RANSAC scheme to compute the initial transformation $\boldsymbol{T}_{cq}$, which is then refined by minimizing the reprojection error. Using the optimized $\boldsymbol{T}_{cq}$, we search for additional matches in the vicinity of KF_q and KF_c by projecting associated MPs from KF_q into KF_c and vice versa. In the case of a *loop closure*, we first perform a pose-graph optimization using the scheme of [119], followed by transforming the MPs using the optimized poses. For the *map matching*, we transform the map of the query KF into the map of the candidate KF using $\boldsymbol{T}_{cq}$. Finally we perform global BA (GBA) using the following objective function:

$$J(\boldsymbol{\Theta}) \coloneqq \|\boldsymbol{e}_p^c\|^2_{\boldsymbol{W}_p^c} + \sum_{k\in\mathcal{V}} \sum_{j\in\mathcal{L}(k)} \delta\left(\|\boldsymbol{e}_r^{k,j}\|^2_{\boldsymbol{W}_r^{k,j}}\right) \tag{6.11}$$
$$+ \sum_{k-1,k\in\mathcal{V}} \left(\|\boldsymbol{e}_a^{k-1,k}\|^2_{\boldsymbol{W}_a^{k-1,k}} + \|\boldsymbol{e}_b^{k-1,k}\|^2_{\boldsymbol{W}_b^{k-1,k}}\right),$$

where the first term is a prior on the pose of KF_c in order to remove the gauge degree of freedom. The optimization is performed in two steps; at first, we only optimize the MP positions and the IMU states, while fixing the KF poses, followed by a full optimization over all states involved in Equation (6.11). Note that the IMU constraints in Equation (6.11) are only inserted between consecutive KFs created by the same agent.

5 Experimental Results

We evaluate CVI-SLAM in a variety of different scenarios to test its performance, applicability, scalability and accuracy. We analyze the network traffic between agents and the server to reveal the bandwidth requirements of CVI-SLAM, as well as the timings of the modules onboard an agent, in single-agent and multi-agent scenarios. Furthermore, we show the importance of the redundancy detection on the server side, and evaluate the accuracy of the system on the five Machine Hall (MH) sequences (MH1 – MH5) of the EuRoC dataset [11], where a small UAV flies several trajectories through an industrial environment, and compare the results

to other state-of-the-art SLAM and VIO systems, namely the visual-inertial version of ORB-SLAM [98] and VINS-mono [111], which both also use an IMU pre-integration scheme. For all experiments, we use the following setup:

- Server: Lenovo Legion Y920 notebook (Core i7-7820HK @ 2.90GHz × 8, 32 GB RAM)
- Agents 1 – 4: Intel NUC 5i7RYH (3.1 GHz × 4, 8 GB RAM), used e.g. on the AscTec Neo UAV

Throughout our experiments, we use a *local window* size of $M = 10$ KFs, *local map* size of $N = 20$ KF, and $K = 10$ KF for messages from the server to an agent. For the analysis on the pre-recorded datasets, the agents and the server are connected via a wireless network, so that real communication between the server and the agents takes place. Then the datasets are processed onboard the agents. This makes our evaluation across different runs more comparable and provides us with ground truth, while still using real network communication as we would during a robotic mission. All values in this section are averaged over 3 runs for each experiment if not stated otherwise.

5.1 Accuracy

In order to obtain a baseline for the accuracy of CVI-SLAM, we first evaluate the absolute error as well as the scale error for the KF trajectory obtained from the server, while running a single agent and compare against state-of-the-art methods. In Table 6.1, we compare the performance to both VI-ORB-SLAM and VINS-mono. With the exception of the sequence MH4, we perform comparably or slightly better than the state-of-the-art methods, indicating the effectiveness of CVI-SLAM's server-agent architecture. We attain the higher error on the MH4 sequence to the fact that CVI-SLAM is able to close the loop approximately half-way through the trajectory, but does not do so at the end and therefore, does not propagate the correction over the whole trajectory. For the evaluation of the map merging capability of our system, we run experiments

Table 6.1: Single-agent RMSE and scale error over the global KF trajectory. The lowest error is indicated in bold.

Dataset	**VI-ORB [98]**		**VINS [111]**		**CVI-SLAM**	
	RMSE	Scale	RMSE	Scale	RMSE	Scale
MH1	**0.075 m**	0.5%	0.177 m	0.359%	0.085 m	1.81%
MH2	0.084 m	0.8%	0.13 m	1.117%	**0.063 m**	1.055%
MH3	0.087 m	1.5%	0.1 m	0.318%	**0.065 m**	0.336%
MH4	0.217 m	3.4%	**0.155 m**	0.947%	0.293 m	3.178%
MH5	0.082 m	0.5%	0.136 m	0.314%	**0.081 m**	0.299%

both in a two-agent as well as in a four-agent setting, while for each agent we use a different sequence of the dataset. As VI-ORB-SLAM is closed source, we are unable to compare against

in this setup. The comparison with the open-sourced VINS-mono is performed using its multi-session capability by sequentially running the different sequences. The values in Table 6.2 report the error of the joint trajectory of all agents at the end of all runs. In all sequences, we consistently perform better than VINS-mono. This can be explained by the fact that we can correct both the KF states as well as the map structure during loop-closures rendering the optimization much more powerful than the pose-graph optimization employed by VINS-mono. Furthermore, as we maintain a covisibility graph and are able to re-use parts of the map, the constraints upon loop-closures are generally stronger. Compared to the single-agent scenario, we get more consistent errors in a similar range, indicating the power of sharing and re-using information between agents in the collaborative setting.

Table 6.2: Multi-agent joint KF-trajectory evaluation with 2 and 4 agents. The lowest error is indicated in bold.

Datasets	**VINS [111]**		**CVI-SLAM**	
	RMSE	Scale	RMSE	Scale
MH1 & MH2	0.158 m	0.150%	**0.050 m**	0.673%
MH2 & MH3	0.197 m	0.408%	**0.073 m**	0.538%
MH4 & MH5	0.198 m	0.575%	**0.115 m**	0.756%
MH1;2;3;5	0.244 m	1.33%	**0.156 m**	0.137%

The last experiment aims at the investigation of the collaborative setting on the tracking that runs onboard an agent. For this purpose, we run the experiments both in a single-agent as well as in a two-agent setting. The error values reported in Table 6.3 are computed by aligning the global trajectory obtained by the tracking with ground truth. For each experiment we align and evaluate only the results of the second sequence in the multi-agent case, while for the single-agent case we only run the second sequence. As it can be seen, the tracking accuracy is consistently improved in the multi-agent setting, even for the sequences MH2 & MH3, which have only little overlap. This highlights the clear benefit of sharing information obtained from multiple agents amongst them in order to improve the accuracy in real-time during collaboration.

Table 6.3: Tracking error on the agent.

Datasets	**CVI-SLAM Single**		**CVI-SLAM Multi**	
	RMSE	Scale	RMSE	Scale
MH1 & MH2	0.224 m	3.693%	**0.139 m**	1.002%
MH2 & MH3	0.295 m	1.444 %	**0.256 m**	0.856%
MH4 & MH5	0.412 m	3.341%	**0.34 m**	1.208%

5.2 Scalability

The efficient architecture of CVI-SLAM boosts the scalability of the system in terms of agents. Fig. 6.4 shows the timings of the onboard processes on the agent (*frame tracking*, *local mapping* of the VIO, *communication*) with the number of participating agents. The number of participants in CVI-SLAM does not influence the timings onboard each agent. With an average tracking time of around 36 ms per frame, the onboard modules can run in real-time with the 20 Hz camera image stream provided by the sensor used for the EuRoC dataset. Also for the

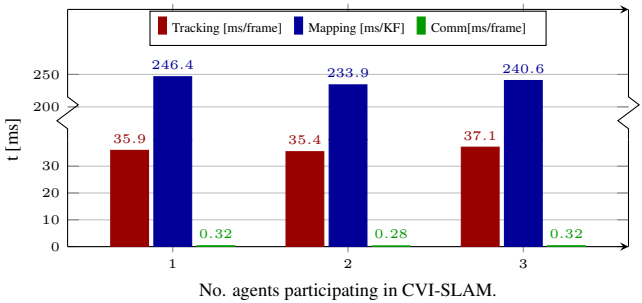


Figure 6.4: Timings of the agent's onboard modules with a growing number of participating agents, measured on Agent 1. All values are averaged over 5 runs using the EuRoC dataset sequences MH1, MH2 and MH3.

network traffic shown in Fig. 6.5, no increase from more participating agents can be observed, implying a linear scalability of the network traffic with the number of agents. The reduced network traffic from server to agent originates from the situation that in a multi-agent scenario, more loop closures occur compared to the single-agent case, resulting in a increased number of optimization steps and therefore, more time periods where a server map is locked and no data is be transmitted to the agents.

On the server's side, more participating agents result in more data to be managed. While place recognition using an inverted file index scales linearly with the number of KFs, the complexity of BA is cubic in the number of KFs and MPs incorporated in the optimization step. Therefore, the detection and removal of redundant information is essential for the scalability of a collaborative SLAM system with the number of agents. Fig. 6.6 shows the resulting number of KFs in the *server map stack* with and without redundancy detection throughout the experiment, evaluated in a 4-agent scenario. The tendency clearly shows a smaller increase in the number of KFs in the system over time, boosting the scalability of CVI-SLAM. The redundancy detection could reduce the number of KFs by 20% from 1460 to 1170, achieving similar accuracy of the resulting estimate.

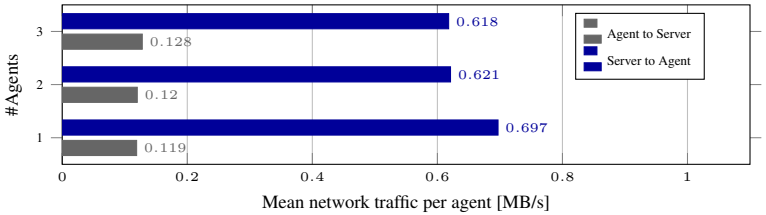


Figure 6.5: Network traffic between agent and server with a growing number of participating agents. All values are averaged over 5 runs for each experiment using the EuRoC dataset sequences MH1, MH2 and MH3.

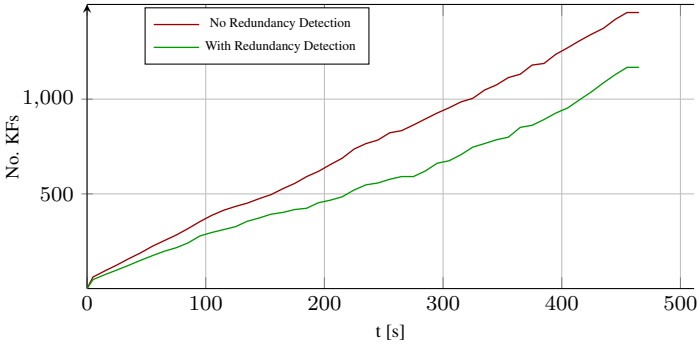


Figure 6.6: Number of KFs in the *server map stack* over runtime in a 4 agent scenario, with and without redundancy detection and removal, using the EuRoC sequences MH1, MH2, MH3 and MH5.

5.3 Network Traffic

Three sequences of the EuRoC dataset are used to analyze the network traffic between the agents and the server. While an agent sends new KFs and MPs to the server when they enter the map, the server sends with every message the $K = 10$ closest KFs to the current position of the agent. The agent can send up to 5 messages per second to the server, while for the server this number is limited to 2, since these messages are usually larger than the ones from the agent. Table 6.4 shows the average traffic between server and agent. On the server, the average traffic is around 0.7 MB/s, which is composed of 2×10 KF messages per second of approximately 30 KB, plus a varying number of observed MPs. By reducing the publishing frequency or the number K of KFs per message, the network traffic can be reduced, though also reducing the information shared with the agent. For the agent, the main influencing factor on the traffic, besides the message limit per second, is the number of KFs inserted in the map. Since the datasets MH3 and MH5 are more challenging than MH1, more KFs are inserted here, resulting in a slightly higher network traffic from the agents to the server for these datasets.

Table 6.4: Mean network traffic between an agent and the server on different trajectories of the EuRoC dataset

Dataset	Agent → Server [MB/s]	Server → Agent [MB/s]
EuRoC MH1	0.119	0.697
EuRoC MH3	0.135	0.707
EuRoC MH5	0.151	0.720

Fig. 6.7 shows the network traffic between the server and the agent over runtime. The communication from agent to server starts with successful IMU initialization, therefore the first message contains the initial map with the size of this depending on the size of the IMU initialization window (15 KFs in our case), causing the spike in the network traffic at the beginning of each experiment.

From server to agent (S→A), the traffic first fluctuates around the mean traffic of approximately 0.7 MB/s, but shows several drops later in the experiment. When the agent returns to previously visited locations, the global BA triggered by the loop closures temporarily suspends the transmission of data to the agents, causing these drops.

6 Conclusion

In this paper, we present CVI-SLAM, an accurate and powerful system for keyframe-based collaborative SLAM. Participating agents are equipped with a visual-inertial sensor suite and constraint onboard calculation power, sharing all information throughout the mission with a more powerful central server. The server merges information from the participating agents and distributes it throughout the system, such that agents can profit from measurements contributed by collaborating agents. Our experiments demonstrate that CVI-SLAM's accuracy is comparable to state-of-the-art visual-inertial SLAM systems, outperforming them in collaborative

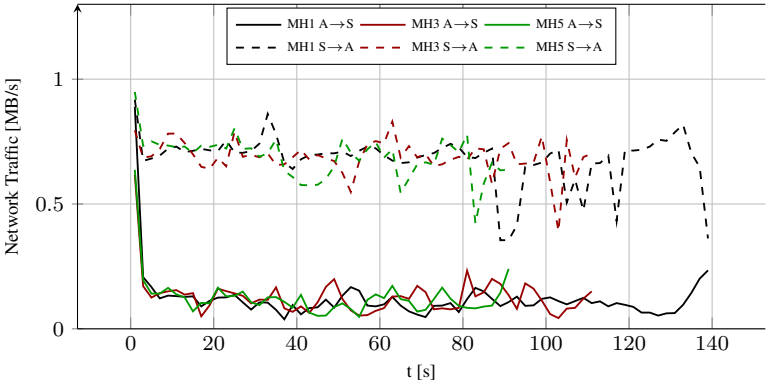


Figure 6.7: Network traffic from agent to server (A→S) and vice-versa (S→A) on several sequences of the EuRoC dataset. Sending the initial map after IMU initialization causes a higher traffic on the agent side at the beginning. Global optimization locking the map for data exchange, resulting from loop closure, causes the drops towards the end of the trajectories on the server side.

scenarios. Furthermore, our evaluation confirms that sharing information amongst participating agents during collaborative SLAM estimation improves the accuracy of pose estimation onboard each agent in real-time compared to single-agent scenarios. A comparison of single- and multi-agent experiments and analysis of the network traffic attests to the scalability and applicability of CVI-SLAM. Compared to existing collaborative SLAM systems, CVI-SLAM combines efficient collaboration in mapping *and* localization, sharing all information amongst all participating agents, with accurate collaborative scene estimation and practical applicability of the system. To the best of our knowledge, CVI-SLAM is the first full visual-inertial collaborative SLAM system implementing two-way communication between agent and server, tested on real data. Future work will focus on further boosting the accuracy improvements achieved from collaborative scene estimation, and increasing the number of participating agents in the system.

Paper IV

Collaborative 6DoF Relative Pose Estimation for Two UAVs with Overlapping Fields of View

Marco Karrer, Mohit Agarwal, Mina Kamel, Roland Siegwart and Margarita Chli

Abstract

Driven by the promise of leveraging the benefits of collaborative robot operation, this paper presents an approach to estimate the relative transformation between two small Unmanned Aerial Vehicles (UAVs), each equipped with a single camera and an inertial sensor, comprising the first step of any meaningful collaboration. Formation flying and collaborative object manipulation are some of the few tasks that the proposed work has direct applications on, while forming a variable-baseline stereo rig using two UAVs carrying a monocular camera each promises unprecedented effectiveness in collaborative scene estimation.

Assuming an overlap in the UAVs' fields of view, in the proposed framework, each UAV runs monocular-inertial odometry onboard, while an Extended Kalman Filter fuses the UAVs' estimates and common image measurements to estimate the metrically scaled relative transformation between them, in real-time. Decoupling the direction of the baseline between the cameras of the two UAVs from its magnitude, this work enables consistent and robust estimation of the uncertainty of the relative pose estimation. Our evaluation on both on simulated data and benchmarking datasets consisting of real aerial data, reveals the power of the proposed methodology in a variety of scenarios.

Published in:
IEEE International Conference on Robotics and Automation (ICRA), 2018
DOI: 10.1109/ICRA.2018.8461143

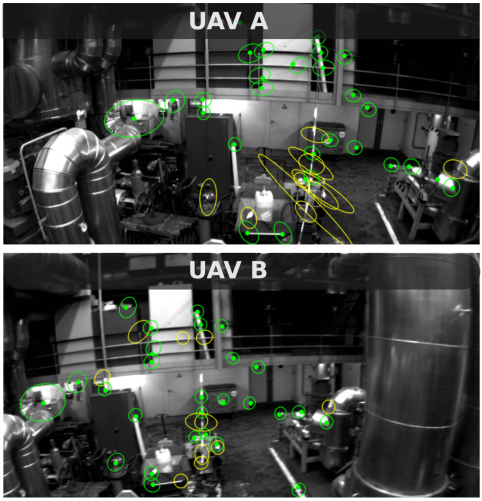


Figure 7.1: A snapshot of the proposed system on the EuRoC [11] dataset, showing the camera views of the two UAVs. Green points correspond to map landmarks successfully detected in image space. The ellipses indicate the projected uncertainty of the predicted landmark positions in the image plane, with yellow corresponding to any unmatched landmarks. Fusing cues from monocular-inertial odometry running onboard each UAV with cross-camera correspondences into an EKF framework, the relative pose between the two UAVs gets estimated in real-time.

1 Introduction

The capability of a robot to estimate its pose in a previously unknown environment, while simultaneously mapping the scene, commonly referred to as Simultaneous Localization And Mapping (SLAM), is a key enabler of autonomous navigation. While initially the problem of SLAM was addressed using range sensors, it was with the emergence of real-time capable SLAM systems using a monocular camera, such as [31] and [70] that SLAM onboard small Unmanned Aerial Vehicle (UAV) started being investigated. Using the combination of both visual measurements as well as readings from an Inertial Measurement Unit (IMU), [141] demonstrated the strength of this combination in a vision controlled flight using only onboard sensing capabilities, rendering Visual-Inertial (VI) sensing as the preferred choice for the control and the navigation of a small UAV. Current state of the art Visual Inertial Odometry (VIO) systems, such as OKVIS [82] and ROVIO [9], have reached a remarkable maturity and robustness and their publicly available implementations had great impact in the community. Most recently, the work of [98] presented a complete VI-SLAM system achieving global consistency of the map at metric scale.

Alongside with the increased maturity of single-robot SLAM systems, the community started to gain interest in collaborative SLAM using multiple robots. The system proposed in [149] for example, leverages view overlap from multiple individual moving cameras to perform SLAM in challenging dynamic scenes. While the system introduces very powerful ideas, the reliance on a GPU and the absence of metric scale limits the applicability of the approach in robotic application, especially on UAVs. The approaches of [43] and [119] demonstrate the use of multiple UAVs, each equipped with a monocular camera to perform efficient mapping and collaborative SLAM, respectively. Whereas these systems use a centralized architecture, ongoing research has been studying the SLAM estimation process in a distributed fashion aiming for scalability of the approach with the number of agents. While [27] presented a method to achieve consistency using only peer-to-peer communication and assuming known correspondences, [34] showed a distributed SLAM pipeline on multiple UAVs utilizing 2D laser scans for mapping.

Consistent mapping from multiple UAVs is necessary for planning the movements of a swarm of robots, e.g. coordinating their trajectories for efficient exploration, however, for a wide variety of tasks, such as collaborative manipulation of objects using multiple UAVs or formation flying, knowing the relative pose between UAVs is most important. The work of [41] aims at estimating the relative pose between a UAV and a ground robot by observing a set of LED markers mounted on the UAV with a monocular camera on the ground robot. While [41] relies on direct measurements between the robots, [5] presented an approach to fuse inertial readings together with scaled relative pose measurements obtained via overlap in the Field of View (FoV) in the images from two UAVs.

In this paper, we present an approach to estimate the relative transformation between two UAVs following the footsteps of [5] by utilizing overlap between the images captured from two UAVs in combination with inertial data. As demonstrated by [129], optimization based approaches to visual odometry allow the estimation over a larger set of state variables more efficiently than filter based methods. However, as shown by [9], including a low number of landmarks in the filter state enables a computationally inexpensive state estimation while exhibiting a considerable robustness. Furthermore, the employment of a filter as opposed to an optimization based method allows the extraction of a probability distribution without the need for additional computations, offering a significant advantage in a robotic scenario, e.g. to define safety margins to avoid collisions in a formation flight. Motivated by this, we base our approach on an Extended Kalman Filter (EKF) to fuse local odometry estimates from two UAVs together with the image measurements to estimate the metrically scaled relative transformation between the two UAVs in real-time. The proposed system is completely expressed in local coordinates, reducing the influence of drift stemming from the local odometry system. Furthermore, in contrast to [5], we demonstrate and evaluate the proposed approach both on simulated data as well as on a benchmarking dataset consisting of real data captured with a UAV.

The main contribution of this work is the presented EKF design for the relative pose estimation, where we express the baseline between the two UAVs using a bearing vector and an inverse distance parametrization for the magnitude of the baseline, allowing to reflect the uncertainty of the estimation problem in a natural and intuitive way. Finally, the presented system design is shown to be capable of running in real-time with two UAVs, only requiring peer-to-peer communication, while sharing some of the computational load.

2 Problem Setup

2.1 Notation

In this paper, small letters are used for scalars (x), capital letters for coordinate frames (X) according to their definition in Fig. 7.2, bold capitals for matrices ($\boldsymbol{X}$), and bold small letters for vectors and unit quaternions ($\boldsymbol{x}$). An arbitrary variable $\boldsymbol{x}_{\text{id}}^{\text{k}}$ can be characterized by its timestamp k and some identifier id, which is used to either label the variable (e.g. $\boldsymbol{x}_1, \boldsymbol{x}_2$,...) or to distinguish the coordinate frame it is expressed in (e.g. $\boldsymbol{x}_{\text{A}}, \boldsymbol{x}_{\text{B}}$,...). In order to differentiate between predicted and updated state variables at each timestamp, we use the following convention:

- $\hat{\boldsymbol{x}}^{k+1}$: the predicted state variable at time $k+1$ given the posterior state and system input at time k.
- $\boldsymbol{x}^{k+1}$: the updated state variable at time $k+1$ given the measurements obtained at time $k+1$.

For the representation of rotations, we use unit quaternions, denoted by $\boldsymbol{q}$. Concatenations of two quaternions are denoted by $\boldsymbol{q}_1 \circ \boldsymbol{q}_2$ and the rotation of a vector $\boldsymbol{v}$ by the quaternion $\boldsymbol{q}$ is denoted by $\boldsymbol{q}(\boldsymbol{v})$.

2.2 System Requirements and Assumptions

The proposed system considers two UAVs, each equipped with one monocular camera onboard and a module producing metrically scaled 6-Degrees of Freedom (DoF) egomotion estimation, such as VIO and the ability to to exchange data amongst each other (e.g. over WiFi). While our system does not impose any general constraints on the movements of the UAVs, we make the following assumptions:

- the cameras experience overlap in their fields of view most of the time,
- the system clocks as well as the cameras of the two UAVs are synchronized, and
- the egomotion estimation for both UAVs is stable at all times.

Although a stable egomotion estimation is assumed, the odometry estimates can be noisy and drift over time and we do not require any prior knowledge about the transformation of their local origins.

3 Relative Pose Filter Setup

3.1 State Representation and Parametrization

As the goal of the filter is to estimate the relative pose of the two UAVs independently of their global position, the reference frame for all state variables is chosen to be the camera

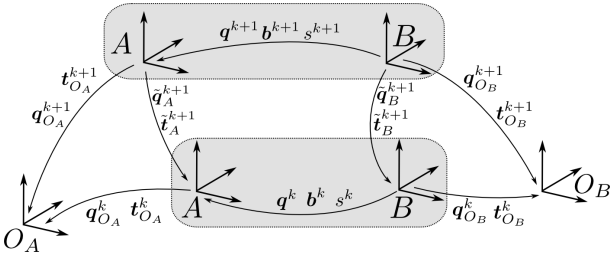


Figure 7.2: The transformations and the coordinate frames used in this paper, with the shaded regions marking the pair of UAVs used as a setup here at two consecutive timestamps (k and $k+1$). The coordinate frames O_A and O_B correspond to the local odometry origins for the UAV A and UAV B, respectively.

coordinate frame of UAV A (A). In this work, we co-estimate a number of landmarks from both UAVs enabling tight coupling between the updates arising from visual measurements and the predictions computed by odometry. The filter state, when there are N visual landmarks tracked, is:

$$\boldsymbol{x} \coloneqq \begin{pmatrix} \boldsymbol{b} & s & \boldsymbol{q} & \boldsymbol{\mu} & \cdots & \boldsymbol{\mu} & \varrho_1 & \cdots & \varrho_N \end{pmatrix}, \tag{7.1}$$

where the following state variables are used:

- $\boldsymbol{b}$: the bearing vector of the baseline from B to A, expressed in A,
- s: the inverse of the metric magnitude of the baseline between the two monocular cameras mounted on the two UAVs,
- $\boldsymbol{q}$: the relative rotation from B to A,
- $\boldsymbol{\mu}_i$: the bearing vector of landmark i in frame A, and
- ϱ_i: the inverse depth of landmark i in frame A.

In order to avoid singularities in the state variables, we express rotations as $\boldsymbol{q} \in SO(3)$ and the bearing vectors as $\boldsymbol{b}, \boldsymbol{\mu} \in S^2$, while both are parameterized as quaternions. To obtain a minimal representation of the state covariance matrix, the lie algebra and the tangent space representations are used for rotations and bearing vectors, respectively. For a detailed explanation and analytical expressions of the used parameterization, we refer the interested reader to [8].

Note that we do not only represent the landmarks using the inverse distance parametrization, but also employ the same parametrization to represent the baseline of the relative transformation. This is especially advantageous at the initialization stage of the filter, as we can get a direct measurement on the direction from only one pair of frames, but not on the metrically scaled distance. The baseline of the relative transformation between UAV A and UAV B is intrinsically expressed by the decoupled nature of the (inverse) distance and the bearing vector formulation.

3.2 State Prediction

For the state prediction, we utilize the poses as obtained by the local odometry systems as

$$\tilde{\boldsymbol{t}}_i^{k+1} = \left(\boldsymbol{q}_{Oi}^k\right)^{-1}\left(\boldsymbol{t}_i^{k+1} - \boldsymbol{t}_i^k\right) + \boldsymbol{\delta}_t, \quad \boldsymbol{\delta}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_t) \tag{7.2}$$

$$\tilde{\boldsymbol{q}}_i^{k+1} = \left(\left(\boldsymbol{q}_{Oi}^k\right)^{-1} \circ \boldsymbol{q}_{Oi}^{k+1}\right) \boxplus \boldsymbol{\delta}_q, \quad \boldsymbol{\delta}_q \sim \mathcal{N}(0, \boldsymbol{\Sigma}_q)\,, \tag{7.3}$$

where the subscript i is used to differentiate the inputs from UAV A and B. The $\boxplus$ operator denotes the generalization of the addition operation for rotations as in [8]. We assume the noise covariances $\boldsymbol{\Sigma}_t$ and $\boldsymbol{\Sigma}_q$ of the computed relative transformations to be diagonal. For simplicity, we summarize the odometry predictions to an input vector given by

$$\boldsymbol{u}^k \coloneqq \begin{pmatrix} \tilde{\boldsymbol{t}}_A^{k+1} & \tilde{\boldsymbol{q}}_A^{k+1} & \tilde{\boldsymbol{t}}_A^{k+1} & \tilde{\boldsymbol{q}}_B^{k+1} \end{pmatrix}\,. \tag{7.4}$$

Using the obtained odometry estimates for the two UAVs, the state prediction is performed by closing the transformation loop:

$$\hat{\boldsymbol{b}}^{k+1} = \left(\tilde{\boldsymbol{q}}_A^{k+1}\right)^{-1}\left(\boldsymbol{q}^k(\tilde{\boldsymbol{t}}_B^{k+1}) + \frac{1}{s^k}\boldsymbol{b}^k - \tilde{\boldsymbol{t}}_A^{k+1}\right)/c_b \tag{7.5}$$

$$\hat{s}^{k+1} = \frac{1}{c_b} \tag{7.6}$$

$$\hat{\boldsymbol{q}}^{k+1} = \left(\tilde{\boldsymbol{q}}_A^{k+1}\right)^{-1} \circ \boldsymbol{q}^k \circ \tilde{\boldsymbol{q}}_B^{k+1} \tag{7.7}$$

$$\hat{\boldsymbol{\mu}}_i^{l+1} = \tilde{\boldsymbol{q}}_A^{k+1}\left(\frac{1}{\varrho_i^k}\boldsymbol{\mu}_i^k - \tilde{\boldsymbol{t}}_A^{k+1}\right)/c_i \tag{7.8}$$

$$\hat{\varrho}_i^{k+1} = \frac{1}{c_i}\,, \tag{7.9}$$

where the intermediate normalization constants c_b, c_i are given by

$$c_b \coloneqq \left\| \boldsymbol{q}^k(\tilde{\boldsymbol{t}}_B^k) + \frac{1}{s^k}\boldsymbol{b}^k - \tilde{\boldsymbol{t}}_A^{k+1} \right\|_2, \quad c_i \coloneqq \left\| \frac{1}{\varrho_i^k}\boldsymbol{\mu}_i^k - \tilde{\boldsymbol{t}}_A^{k+1} \right\|_2 \tag{7.10}$$

Since the used model is already discrete in time, the EKF equations for the covariance prediction can be directly applied:

$$\hat{\boldsymbol{P}}^{k+1} = \boldsymbol{F}^k \boldsymbol{P}^k \, \boldsymbol{F}^{k\,T} + \boldsymbol{G}^k \boldsymbol{Q}^k \, \boldsymbol{G}^{k\,T}\,, \tag{7.11}$$

where $\boldsymbol{Q}^k \in \mathbb{R}^{12\times 12}$ is the covariance matrix of the odometry noise obtained by stacking $\boldsymbol{\Sigma}_q$ and $\boldsymbol{\Sigma}_t$ for both UAVs. The matrices $\boldsymbol{F}^k$ and $\boldsymbol{G}^k$ are the jacobians of the predicted state with

respect to the state variables and the odometry input, respectively:

$$\boldsymbol{F}^k \coloneqq \frac{\partial \hat{\boldsymbol{x}}^{k+1}}{\partial \boldsymbol{x}^k} \in \mathbb{R}^{6+3N\times 6+3N} \tag{7.12}$$

$$\boldsymbol{G}^k \coloneqq \frac{\partial \hat{\boldsymbol{x}}^{k+1}}{\partial \boldsymbol{u}^k} \in \mathbb{R}^{6+3N\times 12}\,, \tag{7.13}$$

3.3 State Update

For every pair of images from the two UAVs, an update of the state is performed. Assuming that the camera intrinsics for both UAVs are known, the mapping between a bearing vector $\boldsymbol{\mu}$ in the camera's coordinate frame and the corresponding pixel coordinates $\boldsymbol{p}$ in the image is given by

$$\boldsymbol{p} = \pi\left(\boldsymbol{\mu}\right)\,. \tag{7.14}$$

Using the reprojection error in the image plane as a measurement, we can formulate the residuals as

$$\boldsymbol{r}_{i,j}^{k+1} = \boldsymbol{z}_{i,j}^{k+1} - h_j(\hat{\boldsymbol{x}}^{k+1})\,, \tag{7.15}$$

where $\boldsymbol{z}_{i,j}^{k+1}$ denotes the measured detection of landmark i in image plane of UAV j and $h_j(\cdot)$ corresponds to the prediction of the measurement, given as

$$h_A(\hat{\boldsymbol{x}}^{k+1}) = \pi_A(\hat{\boldsymbol{\mu}}_i^{k+1}) \tag{7.16}$$

for the prediction in UAV A's image and

$$h_B(\hat{\boldsymbol{x}}^{k+1}) = \pi_B\left(\hat{\boldsymbol{q}}^{k+1}\left(\frac{1}{\hat{\varrho}_i^{k+1}}\hat{\boldsymbol{\mu}}_i^{k+1} - \frac{1}{\hat{s}^{k+1}}\hat{\boldsymbol{b}}^{k+1}\right)^{-1}\right) \tag{7.17}$$

for the measurement prediction in UAV B's image frame. The residual vector $\boldsymbol{r}^{k+1}$ is obtained by stacking all residual terms for both UAVs. Similarly, we obtain the observation matrix $\boldsymbol{H}^{k+1} \in \mathbb{R}^{4N\times 6+3N}$ by stacking the individual jacobians of the predicted measurements given by

$$\boldsymbol{H}_{i,j}^{k+1} = \frac{\partial h_i(\hat{\boldsymbol{x}}^{k+1})}{\partial \hat{\boldsymbol{x}}^{k+1}} \in \mathbb{R}^{2\times 6+3N}\,. \tag{7.18}$$

Together with the predicted covariance, we can compute the residual covariance given by

$$\boldsymbol{S}^{k+1} = \boldsymbol{H}^{k+1}\hat{\boldsymbol{P}}^{k+1}\,\boldsymbol{H}^{k+1\,T} + \boldsymbol{O}^{k+1}\,, \tag{7.19}$$

with $\boldsymbol{O}^{k+1} = \operatorname{diag}\left(\ldots, \sigma_{z_{j,i}}^2, \sigma_{z_{j,i}}^2, \ldots\right) \in \mathbb{R}^{4N\times 4N}$ representing the stacked covariances of the measured landmark detections. At this stage, we perform a Mahalanobis distance based outlier detection, allowing to reject spurious matches by comparing the obtained residuals with the predicted residual covariance. Using the the residual covariance, the Kalman gain $\boldsymbol{K}$ is

computed by

$$\boldsymbol{K}^{k+1} = \hat{\boldsymbol{P}}^{k+1}\ \boldsymbol{H}^{k+1\,T}\ \boldsymbol{S}^{k+1\,-1}\ . \tag{7.20}$$

Given the Kalman gain, the updated state is computed using

$$\boldsymbol{x}^{k+1} = \hat{\boldsymbol{x}}^{k+1} \oplus \boldsymbol{K}^{k+1}\boldsymbol{r}^{k+1}\ , \tag{7.21}$$

where we use the operator $\oplus$ to indicate the use of $+$ for vector and scalar states and the $\boxplus$ operation for rotations and bearing vectors. The updated state covariance is then computed using

$$\boldsymbol{U} = \boldsymbol{I} - \boldsymbol{K}^{k+1}\boldsymbol{H}^{k+1} \tag{7.22}$$

$$\boldsymbol{P}^{k+1} = \boldsymbol{U}\hat{\boldsymbol{P}}^{k+1}\boldsymbol{U}^{T} + \boldsymbol{K}^{k+1}\boldsymbol{O}^{k+1}\ \boldsymbol{K}^{k+1\,T} \tag{7.23}$$

4 System Design

In order to keep the computational complexity of the filter bounded, the proposed approach maintains a constant number of landmarks in the state for the estimation. Therefore, as landmarks can get out of the overlapping field of view during motion, we employ a heuristic strategy to dynamically initialize new and replace any old landmarks that have not been measured consistently during operation. The rest of the section describes these processes in detail.

4.1 Computation Architecture and Keypoint Detection

One of the main motivations of this work is to be able to run an odometry system onboard of each UAV independently, permitting fail-safe control regardless of communication issues of (e.g. network delays). In this way, in the case that collaboration is not possible, the UAVs can still be able to stabilize themselves, while when the UAVs experience sufficient overlap in their fields of view, the proposed filter estimates the relative pose of the two UAVs leading to collaborative mapping. We propose to perform the execution of the filter onboard UAV A, which forms our reference frame. However, keypoint detection can be performed independently on each camera feed and thus, can be run onboard the each UAV independently, distributing some of the computation. In the proposed system, the keypoints are detected using customized Harris corner detection used in [82] and described via an ORB descriptor extraction [116]. Therefore, UAV B forms messages only summarizing its local odometry pose estimate together with its keypoint locations and their descriptors and transmit these to UAV A. As a reference, in our setup each keypoint requires 40 Bytes, 32 for the ORB descriptor and 8 for the keypoint location. With a upper limit on the detected keypoints of 500 and 20Hz framerate, adding up to a bandwidth requirement of 0.4 MB/s, which is easily feasible using standard WiFi modules (e.g. IEEE 802.11g standard).

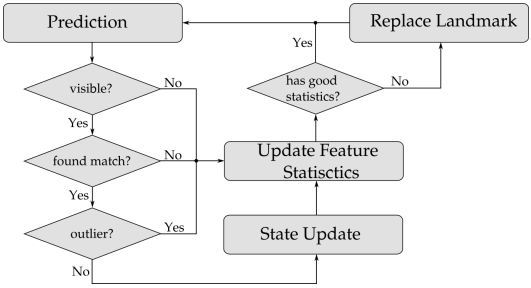


Figure 7.3: Workflow of the dynamic landmark replacement in the filter state. All steps are performed for both UAVs and the decision for landmark replacement is based on the combined statistics.

4.2 State Initialization

For the initialization of the filter, we assume that a rough estimate of the magnitude of the UAVs' baseline is available. As both the bearing vector and the relative orientation can be estimated from a single pair of images, we establish a set of 2D correspondences via descriptor matching between the images of the two UAVs captured at the same time. As described in [82], the descriptor matching of binary descriptors can be performed very fast, permitting brute-force search for correspondences across the two images. Relative RANSAC is then performed using [72] on the matched keypoints to reject outliers and compute the relative transformation between the two UAVs. This transformation is scaled to the magnitude of the initial guess for the baseline. Finally, we attempt to triangulate any remaining matches using the scaled initial pose. If enough matches can be triangulated, we select a random subset for the initialization and set the state variables according to the relative pose and the selected triangulated landmarks, respectively. Initially, the covariance of the metric magnitude of the baseline s is set to a large nominal value, while the covariances of $\boldsymbol{b}$ and $\boldsymbol{q}$ are set to smaller values as in contrast to s, these are measured during the initialization. The initial covariances of the landmarks are obtained by considering the uncertainty of the (relative) pose used for the triangulation. In order to be able to re-detect landmarks, we associate each landmark to both of the keypoint descriptors used to triangulate it.

4.3 Matching and Landmark Management

Since landmark positions are estimated in the state, their predicted positions at the next timestamp are used to obtain new landmark measurements for the state update. This is performed by projecting the landmarks into the corresponding image of each UAV and searching for candidate keypoint matches within a radius, which is chosen reflecting the projected uncertainty of the landmark. During matching, the descriptor used for each landmark is the one obtained during the initialization of this landmark. The candidate with the lowest descriptor distance to

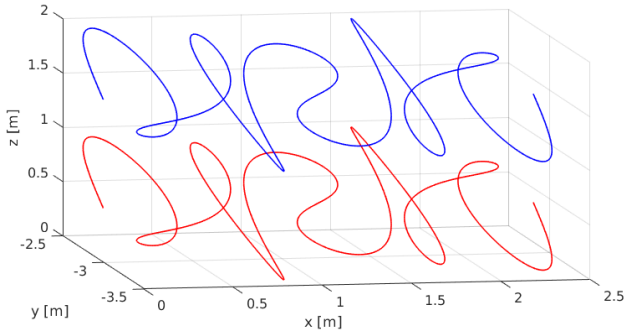


Figure 7.4: Simulated trajectory of the two UAVs for the experiment with a constant relative transformation.

the query landmark that is smaller than a threshold is considered to be a match.

For the decision on when to replace a landmark, we employ a heuristic approach as illustrated in Fig. 7.3 accounting for different failures of the detection of each landmark. In essence, each landmark has two counters associated to it; one for each UAV-agent, which gets incremented by a fixed value in case a test fails and decreases (until the minimum of zero) if a successful update is carried out. The increment for a failure at different cases is increasing, e.g. an outlier is weighted stronger than a landmark that is projected outside of the image. Both counters are independent from each other, however, at the decision whether a landmark needs to be replaced, the counter with the higher value is decisive. Upon the decision to replace landmarks, we perform again brute force matching on the keypoints without a landmark association, followed by checking the distance to the epipolar line in order to reject outliers. The remaining correspondences are used for triangulation, after which we select the landmark(s) used to newly insert in the state, at random. As described in Section 4.2, we associate the two corresponding feature descriptors to the landmark and insert it into the state.

5 Experimental Results

In this section, results using both simulated as well as real data using the EuRoC [11] benchmarking dataset are presented. For both experiments we set the number of landmarks estimated in the filter state to be 40.

5.1 Simulation Setup

In the simulation, we setup an environment consisting of a set of 3D-landmarks randomly distributed in a plane. We generate ground truth trajectories for both UAVs and construct artificial frames with ground truth for both the keypoints' positions obtained by projecting the map land-

marks, as well as for the matching correspondences given by an identifier of the projected landmark. This allows to test the filter performance without any uncertainty introduced in matching. The odometry input is computed by disturbing the relative poses between two subsequent frames with a noise of $\sigma_t = 0.005m$ and $\sigma_q = 0.1°$. Furthermore, the keypoint positions are disturbed by zero-mean gaussian noise with 2 pixels standard deviation. In this setup, we performed two experiments: (i) one maintaining a constant relative transformation between the UAVs, while performing an constant movement along the x-axis and sinusoidal excitation both along the camera axis as well as the z-axis, as shown in Fig. 7.4, and (ii) one, where UAV A performs the same motion as in (i), while UAV B moves in a similar manner, but with an offset resulting in a somewhat oscillating relative transformation.

5.2 Real Data Experimental Setup

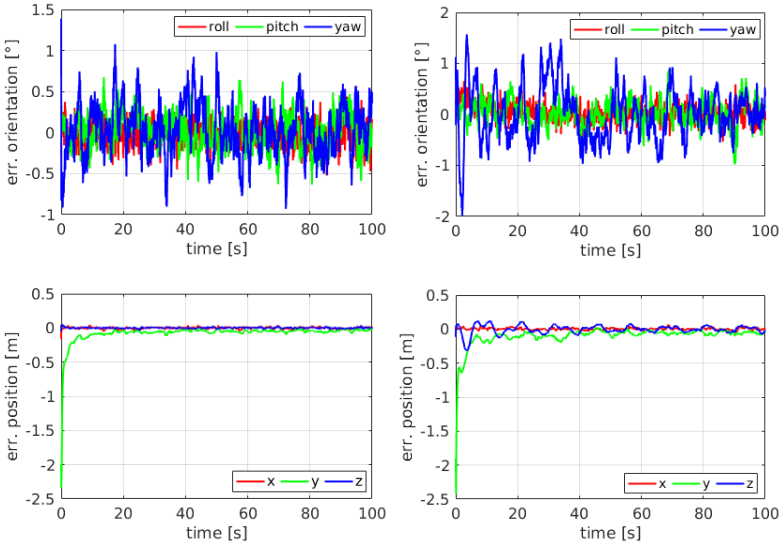
For the experiments using real-data, we utilize the publicly available EuRoC dataset [11], which consists of different sequences recorded from a UAV flying various trajectories in two different environments, both for a room sized scenario (Vicon Room) and in a larger industrial environment (Machine Hall). The onboard sensor suite captures stereo WVGA images at $20Hz$ from a global shutter camera along with inertial readings at $200Hz$ from a hardware-synchronized IMU.

The Vicon Room sequences are not suitable for these experiments, as the UAV is equipped with a forward-looking camera exhibiting predominantly fast rotations rendering it impossible to guarantee overlapping fields of view. Therefore, we evaluate the proposed system on the Machine Hall scenario, namely on the sequences *MH_01_easy* and *MH_02_easy*. As the dataset consists of single UAV trajectories and there is only sporadic view overlap amongst different sequences, we simulate the two UAVs using one EuRoC sequence for one UAV and the same one for the other UAV, but with a time-offset, choosing the left stereo camera of the sensor suite for UAV A and the right one for UAV B. We test with a time-offset of 1s for both sequences, as well as 2s time-offset for *MH_02_easy*, leading to reasonable baselines and sufficient variation, while maintaining sufficient view overlap throughout each test. All experiments are run on a single computer, however, using the same architecture as described in Section 4.1, i.e. data exchange is performed via internal memory access instead of WiFi. For the VIO estimation of both UAVs, we use ROVIO [9]. Note that the approach is agnostic to the used VIO system.

5.3 Results and Discussion

The results for both simulated experiments are shown in Fig. 7.5, while the averaged errors are reported in Table 7.1. Note that for the computation of the Root Mean Squared Error (RMSE), only estimates obtained after the convergence of the filter are considered. As it can be seen, the filter is able to converge in both cases. In contrast to the approach in [5], the proposed system is able to converge even without relative motion. However, as it is evident on the angular error for the full experiment, the relative orientation converges coupled with the position, which is related to the fact that in our system the relative poses of the UAVs are estimated via landmark estimation.

For the real-data experiments, a resulting error plot for the initial part of the sequence *MH_02_easy* with a time offset of $1s$ is shown in Fig. 7.6. After an initial error on the relative distance of

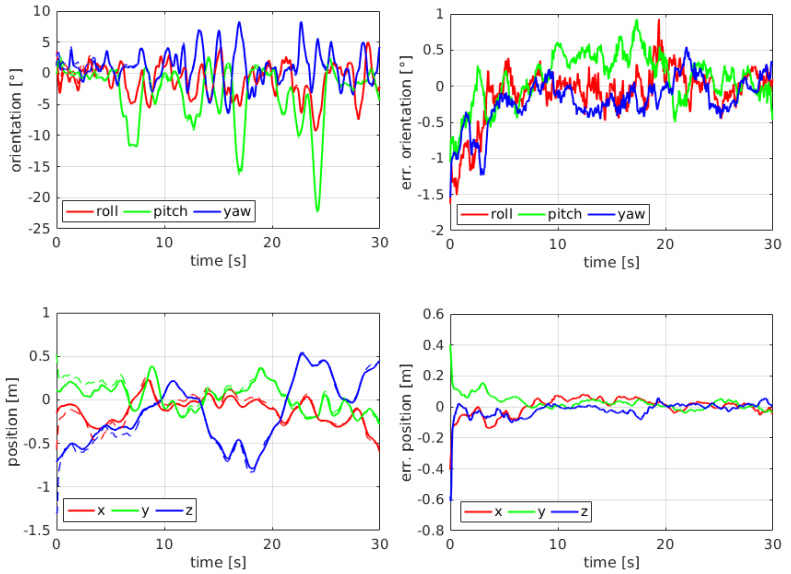


(a) Constant distance **(b)** Full motion

Figure 7.5: The resulting errors converted into roll-pitch-yaw angles and Euclidean coordinates. (a) shows the error for the experiment with a constant relative transformation, while (b) shows the result for the experiment with some oscillating relative motion.

about $1m$, the error, both on translation and rotation decreases initially quickly and reaches a convergent state after approximately $8s$. However, during our experiments we could observe that the convergence rate usually is variable, depending on the initial configuration, but also on the initial matches, which are chosen at random as outlined in Section 4.2. The averaged RMSE for both rotation and translation are reported in Table 7.1. The average errors for the $1s$ offset experiments are approximately $0.5°$ on the rotation and $0.06m$ on the translation. On the sequence *MH_02_easy* with a $2s$ offset, the errors increase compared to the experiments with $1s$ offset, which can be attained to two reasons: (a) the baseline is generally larger indirectly influencing the error, as, for example, any scale error in the estimation results in proportional errors in the absolute distance of the baseline, and (b) the viewpoint changes are larger, resulting to a smaller overlap, which in turn, results in shorter feature tracks (i.e. on average a larger number of landmarks gets replaced per frame pair).

To evaluate the complexity of the proposed algorithm, the runtime was recorded over all real-data sequences. The statistics, broken down into the different parts of the algorithm are shown in Table 7.2. All timings are obtained on an Intel Core i7-4710MQ with 16GB RAM running at 2.5GHz. On average, the total execution time per frame pair is approximately $13.5ms$, resulting



(a) Trajectory **(b)** Error

Figure 7.6: The initial part of the *MH_02_easy* sequence with $1s$ offset.(a) shows the relative transformation over time, where the estimated transformation is plotted with dashed line and the ground-truth with the continuous line, and (b) shows the resulting errors on the relative transformation. Note that the error in orientation is too small for the dashed line to be visible in the top left figure.

in an average frame rate of $74Hz$, whereas the maximum time per frame pair overall is $25ms$. With a frame rate of $20Hz$ our algorithm can therefore, easily run in real-time. The largest variation in timings correspond to the landmark management, since the 2D-2D matching is only executed upon the initialization of new landmarks, as described in Section 4.3. Note that the timings for the keypoint detection and descriptor extraction, which together require $7.5ms$ on average per frame, are not included in Table 7.2, as this process runs in parallel to the actual algorithm.

6 Conclusion

This work presents an EKF design enabling real-time 6-degree-of-freedom relative pose estimation between two UAVs with overlapping fields of view, using local odometry estimates

	RMSE $\boldsymbol{q}$ [°]	RMSE $\boldsymbol{t}$ [m]
Sim. Constant	0.39	0.054
Sim. Dynamic	0.56	0.071
MH_01_easy (1s)	0.63	0.058
MH_02_easy (1s)	0.48	0.063
MH_02_easy (2s)	0.82	0.093

Table 7.1: Average RMS errors of the relative transformations for both the simulation and the real experiments. The rotational errors are obtained by transforming in the roll-pitch-yaw formulation, while the translation errors $\boldsymbol{t}$ are obtained by transforming $\boldsymbol{b}$ and s in Euclidean coordinates. All results are recorded by averaging over 5 runs.

	Prediction	Matching	Update	Landmark Manag.
mean [ms]	1.31	0.15	7.51	4.49
std [ms]	0.29	0.05	0.95	4.41
max [ms]	3.60	1.03	13.76	12.66

Table 7.2: The execution time of the proposed framework, broken down into the different steps. The reported statistics correspond to the real experiments, when using 40 landmarks in the state.

along with monocular vision measurements. Along with the filter design, we propose a new, minimal parametrization of the baseline between the UAVs' cameras as a bearing vector and an inverse-distance, enabling a consistent representation of the uncertainty of the estimation problem. The outlined lightweight system is designed to run onboard two UAVs only using peer-to-peer communication with a bandwidth requirement under 0.5MB/s, while distributing some of the computational load. We demonstrate the capability of the proposed system both on simulated data as well as on real data on the EuRoC benchmarking dataset.

Future work will address the landmark management system in order to improve the performance through longer feature tracks by storing a larger number of landmarks than in the actual filter state (e.g. as in [9]). Furthermore, we believe the robustness of the system could be boosted by incorporating more informed feature selection in the spirit of [16].

Paper V

Distributed Variable-Baseline Stereo SLAM from two UAVs

Marco Karrer and Margarita Chli

Abstract

Visual Inertial Odometry (VIO) has been widely used and researched to control and aid the automation of navigation of robots especially in the absence of absolute position measurements, such as Global Positioning System (GPS). However, when observable landmarks in the scene lie far away from the robot's sensor suite, as it is the case at high altitude flights, the fidelity of estimates and the observability of the metric scale degrades greatly for these methods. Aiming to tackle this issue, in this article, we employ two Unmanned Aerial Vehicles (UAVs) equipped with one monocular camera and one Inertial Measurement Unit (IMU) each, to exploit their view overlap and relative distance measurements between them using Ultra-Wide Band (UWB) modules onboard to enable collaborative VIO. In particular, we propose a novel, distributed fusion scheme enabling the formation of a virtual stereo camera rig with adjustable baseline from the two UAVs. In order to control the UAV agents autonomously, we propose a decentralized collaborative estimation scheme, where each agent hold its own local map, achieving an average pose estimation latency of 11ms, while ensuring consistency of the agents' estimates via consensus based optimization. Following a thorough evaluation on photorealistic simulations, we demonstrate the effectiveness of the approach at high altitude flights of up to 160m, going significantly beyond the capabilities of state-of-the-art VIO methods. Finally, we show the advantage of actively adjusting the baseline on-the-fly over a fixed, target baseline, reducing the error in our experiments by a factor of two.

Published in:
arXiv.org, 2020
http://arxiv.org/abs/2009.04801

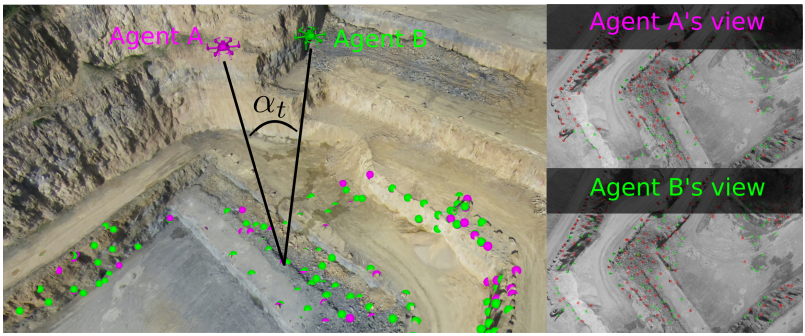


Figure 8.1: Using two UAVs, equipped with one IMU and one monocular camera each, while measuring the relative distance between them using an ultrawideband module, the proposed sensor fusion framework is able to reliably estimate both UAVs' poses in metric scale at the particularly challenging case of high-altitude fights. This is achieved by adjusting the baseline between the UAVs adaptively, to realize a desired triangulation angle α_t. In the distributed implementation, each agent holds its own version of the map, indicated by the green and magenta points in the top image, enabling low latency pose estimation of the UAVs, while consistency across the agents is achieved via a consensus based optimization scheme. The red and blue circles in the bottom two images indicate keypoints with and without 3D point associated with them, respectively.

1 Introduction

Awareness of a robot's pose in previously unseen environments is one of the key elements towards enabling autonomous navigation of robots. While GPS is of tremendous help towards achieving this goal, its availability and quality are still limited, e.g. indoors or close to structure, hence in a general scenario strong reliance on GPS can be a critical limitation. Addressing this issue, research into Simultaneous Localization And Mapping (SLAM) has received quite some attention as it potentially enables ego-motion estimation only using the sensors carried onboard the robot.

Due to the low cost and low power consumption of cameras and their ability to capture rich information about the environment, they have become a popular sensor choice for performing SLAM. With the first systems [31], [36], [70] demonstrating the ability to estimate the ego-motion up to scale using only a single camera, soon more mature and complete systems, such as [96], emerged. While the scale ambiguity inherently present in monocular systems can be resolved, for example, by using a stereo camera rig [97], the lack of information in-between frames renders purely vision-based systems sensitive to fast motions, which are rather common when employed onboard small UAVs. Therefore, the complementary characteristics of IMUs and cameras have been explored extensively to develop robust and accurate state estimation by

the means of Visual Inertial Odometry (VIO) systems, such as [8], [83], [111], which quickly became a standard for autonomous navigation of UAVs [105].
While VIO techniques have been successfully applied in situations where the scene is in the close proximity to the camera (e.g. in autonomous driving, in low-altitude flights or operations indoors), in scenarios where the scene is further away from the camera, such as at high altitude flights, VIO methods experience difficulties to establish similar levels of accuracy. As all VIO methods rely on triangulation of scene landmarks, the scene depth estimates become more uncertain the smaller the triangulation angle gets. For monocular VIO, this angle is determined by the camera motion and the scene depth. As the overall scale of monocular VIO conceptually, is determined via the integration of IMU measurements, with increasing scene depth, a larger range of motion is required in order to obtain a large enough triangulation angle. This leads to the integration of more IMU readings, which at some point results in a numerically weakly constrained scale. The same effect can be observed when using stereo cameras, where the fixed baseline dictates the maximum depth that can be reliably detected [57] and thus, limiting the effectiveness of the imposed constraints on the scale. The choice of the stereo baseline is, thus, a crucial parameter and ideally, one would be able to modify this parameter on-the-fly depending on scene depth or accuracy requirements of the task at hand [50].
Inspired by the idea of using two UAVs as agents equipped with one camera each, to form a virtual stereo camera rig [5], we propose a novel, complete system utilizing view-overlap across two agents together with relative distance measurements obtained via Ultra-Wide Band (UWB) in order to collaboratively estimate the pose of these two cameras. As the two agents are capable of modifying their relative poses, the baseline between their cameras can be adjusted accordingly, e.g. achieving a desired triangulation angle. In order to allow for a low latency pose estimation onboard the agents, independently of network delays, we propose a novel decentralized optimization architecture allowing each agent to hold their own estimate of the map, while ensuring consistency of common scene estimates amongst them. In brief, the main contributions of this work are the following:

- a novel, real-time sensor fusion framework combining monocular visual and inertial data from two UAVs, and relative distance measurements between them (e.g. using ultra-wideband modules on each agent) enabling a reliable relative pose estimation of each other even at high altitudes,

- the adaptation of the framework of [107] for asynchronous multi-agent estimation to enable sliding-window bundle adjustment in a decentralized fashion for the first time to the best of our knowledge,

- a thorough evaluation of the proposed system using photorealistic simulation showing the improvements over state-of-the-art stereo-inertial methods at higher altitudes, and

- demonstration of the advantage of an adjustable baseline in the proposed two-UAV stereo setup for accurate pose estimation by the means of a simple high-level formation controller.

2 Related Work

Alongside the emergence of vision based SLAM for single robots [8, 70, 83, 97, 111], research into multi-robot systems started to attract attention recently. The collaborative framework proposed in [43], demonstrates how Keyframe (KF) data from cameras mounted on different UAVs can be combined into a global map. Designed for mapping visual-inertial data from multiple agents, [64] builds up a global map, showcasing the benefit of collaboration on the overall trajectory error. Utilizing Manhattan-World like structures where available in order to constrain the map optimization, [53] propose a large scale visual-inertial mapping framework and [93] utilize a cloud computer to generate 3D maps, collaboratively. While these approaches are focused on mapping, other systems aim to distribute the processes of the SLAM pipeline between the agents and a server [66, 114, 120], promising to reduce the computational load onboard the agents and to make map data generated from one agent available to all the robotic team. On the other hand, there also exist multi-session frameworks, such as [111, 122] that allow the user to re-localize in previously generated maps. In order to allow these systems to scale up to larger teams of robots, research effort has been aiming to avoid a central server entity and perform all operations in a distributed fashion instead. Many works focus on optimizing a specific aspect of the multi-agent system for a distributed setup, such as place recognition [22], robustness [146], or efficient data exchange [21].

All the aforementioned frameworks make use of collaboration of some form ranging from combining map data into a larger map, to re-using parts of a map created from other agents. Nonetheless, the tracking front-ends of these systems do not require any tight collaboration amongst them and therefore, share the same limitations as their single-agent counterparts, for example, degrading quality of the pose estimate at higher altitudes. A counter example is the collaborative system CoSLAM [149], which addresses the problem of inaccuracies due to dynamic objects in the scene by using multiple freely moving cameras with view-overlap. This approach allows reliable pose estimation of cameras, which only observes the dynamic scene points in collaboration with their neighboring cameras. However, the system requires all image data to be collected at a single access point and makes heavy use of GPU computation, limiting the applicability of the approach to a robotic problem.

The problem of degrading quality of VIO state estimation at high altitudes has recently received some attention in the literature. In wind turbine inspection using UAVs, for example, [134] proposed a framework with two UAVs, one equipped with LED markers and a GPS module and the other one with one camera that can detect the markers and one camera for turbine inspection. For the estimation, the UAV with the cameras observes the other UAV's markers and flies close to the turbine, where the GPS signal is disturbed, while the UAV with the markers flies further away from the turbine to secure more reliable GPS reception, while staying in the field of view of the observing UAV. While this method fits well to inspection of structures, where GPS reception is reliable at least slightly further away from the structure, it is clear that unreliable or imprecise GPS readings have a strong effect on the quality of the pose estimates. Other methods deal with creating a large stereo baseline by placing the two cameras at the tips of a fixed-wing aircraft [57, 58], where the authors model and correct for movements between the cameras due to deformation of the structure essentially continuously estimating the stereo extrinsics. In the work of [5] and [63] the estimation of the relative transformation between two independently moving UAVs equipped with monocular cameras by the means of (rela-

tive) motion and view overlap is investigated. While conceptually, such a configuration can be interpreted as a stereo camera, the absolute scale of the baseline only gets estimated via motion (e.g. IMU measurements), hence suffering from the same limitations as monocular VIO. Nonetheless, taking inspiration from [5], here we propose a framework performing VIO with two agents, each equipped with a monucular camera and an IMU in tight collaboration with each other using relative distance measurements between the agents, e.g. using UWB modules, the scale ambiguity of the variable baseline setup can be addressed, allowing to effectively obtain a virtual stereo camera with a baseline that can be adjusted according to the scene and accuracy requirements. In contrast to other collaborative estimators with tight collaboration on the front-end such as [149], we propose a decentralized architecture allowing for low-latency pose estimation independent of communication delays and fitting well within the bandwidth limitations of a standard WiFi module.

3 Preliminaries

3.1 Notation

Throughout this work, we use small bold letters (e.g. $\mathbf{a}$) to denote vector values, capital bold letters (e.g. $\mathbf{A}$) to denote matrices and plain capitals (e.g. A) denote coordinate frames. To indicate a submatrix formed by the rows r_i to r_j and the columns c_l to c_k of $\mathbf{A}$ we use the notation $\mathbf{A}\left[r_i, r_j; c_l, c_k\right]$, where a 1 based indexing is used. A vector $\mathbf{x}$ expressed in the coordinate frame A, is denoted as ${}_A\mathbf{x}$. Rigid body transformations from coordinate frame B to coordinate frame A are denoted by $\mathbf{T}_{AB}$, comprising the translational part of the transformation $\mathbf{p}_{AB}$ and the rotational part $\mathbf{R}_{AB}$. For notational brevity, at times we use quaternions $\mathbf{q}_{AB}$ interchangeably with such rotation matrices. The concatenation of two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$ is denoted by $\mathbf{q}_1 \circ \mathbf{q}_2$, whereas the rotation of vector $\mathbf{v}$ by a quaternion is denoted by $\mathbf{q}(\mathbf{v})$. Values that correspond to a prediction are indicated with $\hat{\cdot}$, whereas measurements are denoted with $\tilde{\cdot}$. Finally, sets of variables are denoted using capital calligraphic letters (e.g. $\mathcal{A}$).

3.2 Asynchronous-parallel ADMM

The Alternating Direction Method of Multipliers (ADMM) was first introduced by [56] and is an algorithm that aims to solve problems of the following form:

$$\begin{aligned} \underset{\mathbf{x},\mathbf{y}}{\text{minimize}} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{w}, \end{aligned} \tag{8.1}$$

for proper, convex functions $f(\cdot)$ and $g(\cdot)$, by forming the augmented Lagrangian L_γ introducing the dual variables $\mathbf{z}$ and the penalty weight γ as follows:

$$L_\gamma(\mathbf{x}, \mathbf{y}, \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{y}) + \mathbf{z}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{w}) + \frac{\gamma}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{w}\|_2^2 . \tag{8.2}$$

The ADMM algorithm solves the problem in Eq. (8.1) by performing the following update steps iteratively:

$$\mathbf{x}^{k+1} = \quad \arg\min_{\mathbf{x}} L_\gamma(\mathbf{x}, \mathbf{y}^k, \mathbf{z}^k) \tag{8.3}$$

$$\mathbf{y}^{k+1} = \quad \arg\min_{\mathbf{y}} L_\gamma(\mathbf{x}^{k+1}, \mathbf{y}, \mathbf{z}^k) \tag{8.4}$$

$$\mathbf{z}^{k+1} = \quad \mathbf{z}^k + \gamma(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{w}) \,, \tag{8.5}$$

where k is the iteration number. While the problem formulation in Eq. (8.1) corresponds to generic constraint optimization, unconstrained optimization problems with additive cost functions can be brought into an equivalent form, allowing for a distributed implementation over N nodes as used by [147], for example. The form of this distributed problem is given by

$$\begin{aligned} \underset{\mathbf{x_i}}{\text{mininize}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i = \mathbf{S}_i\mathbf{y}, \quad i = 1, \ldots, N \,, \end{aligned} \tag{8.6}$$

where $\mathbf{S}_i$ corresponds to an indicator matrix, selecting the entries in $\mathbf{y}$ corresponding to the variables of $\mathbf{x}_i$. Note that in relation to Eq. (8.1), here $g(\mathbf{y}) = 0$, $\mathbf{w} = \mathbf{0}$, $\mathbf{A}$ is the identity matrix and $\mathbf{B}$ corresponds to the stacked version of the indicator matrices $-\mathbf{S}_i$. As shown in [147], this formulation reduces the update step of $\mathbf{y}$ in Eq. (8.4) to a simple averaging operation (consensus).

While the algorithm proposed by [147] can parallelize the most expensive operation of solving the local optimization problems in Eq. (8.3)-(8.5), in order to compute the consensus terms, all data needs to pass through one central point. So in effect, the synchronized structure of the algorithm dictates the frequency of the ADMM iterations to be equal to the slowest participating node. The algorithmic framework "ARock" introduced in [107], specifically addresses this limitation allowing to arrive to an asynchronous implementation of the ADMM algorithm, which we adopt in this work. A brief overview of the ARock ADMM algorithm presented in [107] is provided below, while the specific details of the algorithm related to the proposed framework are introduced in Section 4.4.

In the general setup, consider a set of nodes $1, \ldots, N$, forming a graph connecting the nodes by a set of edges $\mathcal{E} = \{(i, j) | \text{if node } i \text{ connects to node } j, i < j\}$. A node can be seen as a computational unit holding a partial, local estimate $\mathbf{x}_i$ of the optimization state variable $\mathbf{x}$. For the sake of notational simplicity, in the following, all local variables $\mathbf{x}_i$ are assumed to be realizations of the full state $\mathbf{x}$, i.e. the dimensions of all $\mathbf{x}_i$ are equal to the dimension of $\mathbf{x}$. Additionally, we assume that not every node can communicate with every other node and the edges in $\mathcal{E}$ is given by pairwise set of nodes that can communicate with each other. As shown in Section 4.4, this can be generalized to sharing only a subset of variables between the nodes.

In such a setup, the problem formulated in Eq. (8.6) can be expressed as:

$$\begin{aligned} &\underset{\mathbf{x}_i, \mathbf{y}_{ij}}{\text{minimize}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ &\text{subject to} \quad \mathbf{x}_i = \mathbf{y}_{ij},\ \mathbf{x}_j = \mathbf{y}_{ij} \quad \forall (i,j) \in \mathcal{E}\ . \end{aligned} \tag{8.7}$$

Let $\mathcal{E}_i$ denote the set of edges connected to node i and let $|\mathcal{E}_i|$ denote its cardinality. Furthermore, let $\mathcal{L}_i = \{j | (j,i) \in \mathcal{E}_i, j < i\}$ and $\mathcal{R}_i = \{j | (i,j) \in \mathcal{E}_i, j > i\}$, separating the ordered indices to the left and the right of i, respectively. For every pair of constraints $\mathbf{x}_i = \mathbf{y}_{ij}$ and $\mathbf{x}_j = \mathbf{y}_{ij}$, such that $(i,j) \in \mathcal{E}$ in Eq. (8.7), the dual variables $\mathbf{z}_{ij,i}$ and $\mathbf{z}_{ij,j}$ get associated on node i and j, respectively. Consequently, the ADMM iterations for every node i can be written as

$$\mathbf{x}_i^{k+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + (\mathbf{z}_i^k)^T \mathbf{x}_i + \frac{\gamma}{2} |\mathcal{E}_i| \cdot \|\mathbf{x}_i\|^2 \tag{8.8}$$

$$\mathbf{z}_{li,i}^{k+1} = \mathbf{z}_{li,i}^k - \eta_k \cdot \left((\mathbf{z}_{li,i}^k + \bar{\mathbf{z}}_{li,l})/2 + \gamma \mathbf{x}_i^{k+1} \right), \quad \forall l \in \mathcal{L}_i \tag{8.9}$$

$$\mathbf{z}_{ir,i}^{k+1} = \mathbf{z}_{ir,i}^k - \eta_k \cdot \left((\mathbf{z}_{ir,i}^k + \bar{\mathbf{z}}_{ir,r})/2 + \gamma \mathbf{x}_i^{k+1} \right), \quad \forall r \in \mathcal{R}_i\ , \tag{8.10}$$

where the variables denoted with $\bar{\cdot}$ represent the latest received values from the neighboring nodes and the weight η_k is a factor to account for communication delays and the corresponding potential use of outdated dual variables. Note that $\mathbf{z}_i^k$ corresponds to

$$\mathbf{z}_i^k = \sum_{l \in \mathcal{L}_i} \bar{\mathbf{z}}_{li,l}^k + \sum_{r \in \mathcal{R}_i} \bar{\mathbf{z}}_{ir,r}^k\ . \tag{8.11}$$

Finally, after every iteration of Eq. (8.8)-(8.10) for node i, the updated dual variables $\mathbf{z}_{ji,i}^{k+1}$ get communicated to the nodes $j \in \mathcal{E}_i$ connected to i.

3.3 Z-Spline based 6DoF Pose Interpolation

Representing a trajectory as a continuous-time curve offers several advantages, for example the handling of sensors, which acquire data over a period of time, such as rolling-shutter cameras or LiDARs, as well as simplified fusion of data from multiple, possibly not time-synchronized sensors. One of the most popular representations for continuous-time trajectories are B-splines due to their simple parametric form and their local support properties. In the literature, B-splines have been applied successfully to tackle visual-inertial SLAM using rolling-shutter cameras [47], [86]. As outlined in [86], a standard B-spline curve of degree $k-1$ is defined by

$$\mathbf{x}(t) = \sum_{i=0}^{n} \mathbf{x}_i B_{i,k}(t)\ , \tag{8.12}$$

where $\mathbf{x}_i \in \mathbb{R}^N$ are the control points of the spline at the times t_i with $i \in [0, n]$ and $B_{i,k}(t)$ are the continuous time basis functions. By reorganizing Eq. (8.12), the formulation can be brought into the cumulative form given by

$$\mathbf{x}(t) = \mathbf{x}_0 \bar{B}_{0,k}(t) + \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}_{i-1}) \bar{B}_{i,k}(t) \,, \tag{8.13}$$

where $\bar{B}_{i,k}(t)$ represents the cumulative basis function and the relationship between the cumulative and the standard basis functions is described by

$$\bar{B}_{i,k}(t) = \sum_{j=i}^{n} B_{j,k}(t) \,. \tag{8.14}$$

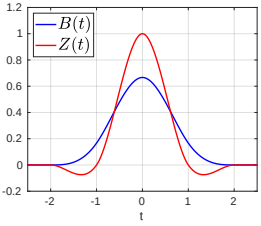
The basis functions for B-splines can be easily computed using De Boor-Cox's recursive formula [24], [32]. In this work in order to represent the continuous time trajectory, we propose to utilize a third order Z-spline representation introduced in [117]. While Z-splines have the same local support properties as B-splines, their basis functions are defined as piece-wise polynomials. In particular, the third order basis function for the Z-spline are defined as

$$Z(s) = \begin{cases} 1 - \frac{5}{2}s^2 + \frac{3}{2}|s|^3 & |s| \leq 1, \\ \frac{1}{2}(2 - |s|)^2(1 - |s|) & 1 < |u| \leq 2, \\ 0 & |u| > 2 \,, \end{cases} \tag{8.15}$$

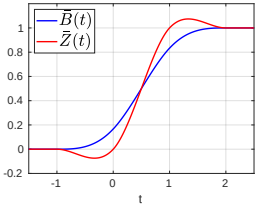
where s denotes a real-value scalar. As in this work we employ cubic splines and utilize control points that are equally spaced in time (i.e. $t_{i+1} - t_i = \Delta t, \forall i \in [0, n-1]$) for any $t \in [t_i, t_{i+1})$, exactly four control points are required, namely the ones at times t_{i-1}, t_i, t_{i+1} and t_{i+2}. Transforming the time t into a local time $u(t) = \frac{t - t_i}{\Delta t}$, the interpolation using the cubic Z-spline can be written as:

$$\mathbf{x}(u) = Z(u+1)\mathbf{x}_{i-1} + Z(u)\mathbf{x}_i + Z(u-1)\mathbf{x}_{i+1} + Z(u-2)\mathbf{x}_{i+2} \,, \tag{8.16}$$

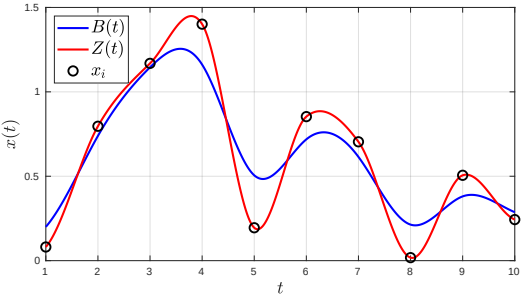
with $u(t)$ denoted by u for brevity. Note that the definition of the cumulative form remains the same as in (8.13). A comparison between the base functions and their cumulative forms is shown in Fig. 8.2. As evident in Fig. 8.2c, in contrast to the B-splines, the Z-spline interpolation passes through the control points exactly. While the use of such a spline formulation on values in $\mathbb{R}^N$ is straightforward, the interpolation of rigid body transformations needs to be considered more carefully. The authors in [86] proposed to interpolate the 6 Degrees of Freedom (DoF) pose using the cumulative formulation on $\mathbb{SE}3$, however, here we perform the interpolation of the rotation on $\mathbb{SO}3$ and the translation parts on $\mathbb{R}^3$ separately. This was already advocated in [68], as such a split representation removes the coupling of the translation and the rotation and avoids artifacts on the translation-interpolation during phases with large rotational velocities.



(a) Standard Basis Function



(b) Cumulative basis Function



(c) Comparison between B-spline and Z-spline based interpolation

Figure 8.2: Comparison between the basis functions of B-splines and Z-splines. The plot in (a) shows the normal basis functions and (b) shows the corresponding cumulative basis function, while (c) shows a comparison of the interpolation result between B-splines and Z-splines using the same control points.

Therefore, the interpolated translation is given by

$$\mathbf{p}(u) = \mathbf{p}_{i-1}\bar{Z}(u+1) + \sum_{j=0}^{2}(\mathbf{p}_{i+j} - \mathbf{p}_{i+j-1})\bar{Z}(u-j)\,. \tag{8.17}$$

Following the approach of [68], the interpolated rotation is computed by

$$\mathbf{q}(u) = \mathbf{q}_{i-1}^{\bar{Z}(u+1)} \prod_{j=0}^{2} \exp\left(\log(\mathbf{q}_{i+j-1}^{-1}\mathbf{q}_{i+j})\bar{Z}(u-j)\right)\,, \tag{8.18}$$

where $\mathbf{q}^{\lambda} = \exp(\lambda \log(\mathbf{q}))$. Essentially, the operations in (8.18) correspond to a mixture of SLeRP interpolations [124] within the local support window of the spline.

4 Method

4.1 System Overview

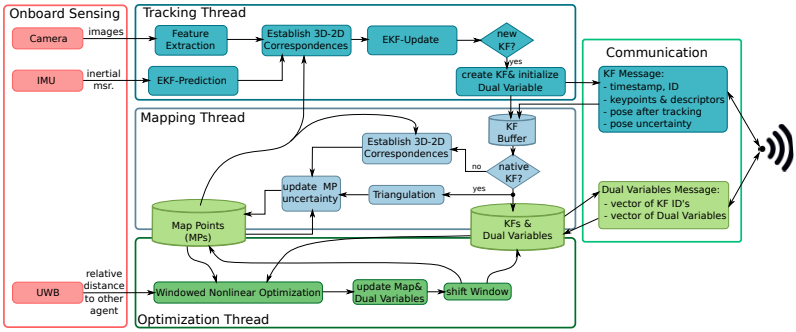


Figure 8.3: Schematic of the processes running on each of the two agent for the proposed system mainly comprising of frame-wise tracking, mapping and non-linear optimization. Consistency between the two agents is achieved by communicating Keyframe (KF) data and Dual Variables associated to the KF poses via wireless communication.

The system performing distributed, collaborative SLAM is designed to be run as two identical instances on two machines, called 'agents', communicating with each other over a wireless connection. An overview of the essential parts of one such instance is shown in Fig. 8.3. The functionality of each instance is partitioned into three main threads: tracking, mapping, and optimization.

In the *tracking thread*, the image observations are fused together with IMU readings in order to localize against the current state of the 3D Map Points (MPs). The fusion is performed by the means of an Extended Kalman Filter (EKF), which allows for a computationally efficient and low-latency frame-wise pose tracking. The tracked frame is further processed to decide whether a KF should be created or not. In the proposed system this decision is purely based on time constraints, i.e. if the time difference between the last KF and the current frame is larger than a threshold, the current frame is marked to become a KF. This scheme is motivated by the simplicity in the book-keeping, but also by the fact that we use the KF poses as base poses in the spline representation of the trajectory in the back-end optimization, hence, using uniform sampling between the KFs reduces the complexity of the interpolation. When a KF is newly created at one agent, the relevant information such as the 2D keypoints, descriptor data as well pose related information gets immediately communicated to the other agent. In the following the KFs

created on its own by an agent are referred to as native KFs, while KFs received via communication but created by the other agent are referred to as non-native KFs. Both, native- as well as non-native KFs are passed to the mapping part of the pipeline. In the *mapping thread*, we try to establish new 3D MPs via triangulation between the newest native KF and a selected subset of established KFs. In order to reduce the required book-keeping for the map maintenance, the system is designed such that each agent can hold its own set of map points, avoiding complex synchronization between the two agents. Besides the creation of new MPs, the mapping thread is also responsible to update the uncertainty estimates of the MPs and establish correspondences between the existing MPs and non-native KFs. In order to avoid extensive locking of data during these operations, the interface between the tracking- and the mapping-thread copies the last state of the MPs to be used for tracking. While the *mapping thread* initializes the MP positions and maintains an estimate of their uncertainty, we utilize nonlinear optimization to update the MP positions and fuse UWB distance measurements between the two agents into the estimation. As we have two instances of the map estimated, one on each agent, in the *optimization thread* we make use of the asynchronous ADMM introduced in Section 3.2 to ensure that both agents converge to a common trajectory estimate by the means of a pose consensus, enforced by exchanging a set of dual variables. In order to keep the computational complexity of the optimization bounded, we only keep a fixed sized window of KFs in the Map, whose size is maintained after every round of optimization.

4.2 EKF-based Pose Tracking

In order to enable both robust and timely tracking of each robot's pose, we designed an EKF fusing the IMU information together with 3D-2D measurements against the current state of the MPs. As the MP positions are conceptually considered as given in the tracking thread, the EKF state $\mathcal{X}_{tr}$ is chosen as follows:

$$\mathcal{X}_{tr} \coloneqq \begin{bmatrix} \mathbf{q}_{WM} & \mathbf{q}_{MS} & \mathbf{p}_{MS} & {}_S\mathbf{v} & \mathbf{b}_a & \mathbf{b}_\omega \end{bmatrix} , \tag{8.19}$$

where $\mathbf{q}_{WM}$ denotes the rotation (in quaternion form) of this agent's map origin into the gravity aligned world frame, $[\mathbf{q}_{MS}, \mathbf{p}_{MS}]$ corresponds to the pose of the IMU frame in the map, ${}_S\mathbf{v}$ denotes the robocentric velocity of the IMU frame, and $\mathbf{b}_a$ and $\mathbf{b}_\omega$ denote the accelerometer- and the gyroscope biases, respectively. The covariance associated to the state $\mathcal{X}_{tr}$ is denoted as $\mathbf{\Sigma}_{tr}$. While rotations are parameterized as quaternions, the gravity rotation ($\mathbf{q}_{WM}$) only has two DoF (roll and pitch), whereas the pose ($\mathbf{q}_{MS}$) has 3DoF, resulting in $\mathbf{\Sigma}_{tr} \in \mathbb{R}^{17\times17}$. As commonly employed in filtering based visual-intertial frameworks, such as in [8], we use the IMU readings to propagate the filter state. The acceleration ${}_S\tilde{\mathbf{a}}_S$ and gyroscope measurements ${}_S\tilde{\boldsymbol{\omega}}_{WS}$ are modeled to be the true acceleration ${}_S\mathbf{a}_S$ and rotational velocity ${}_S\boldsymbol{\omega}_{WS}$ affected by both noise and biases:

$${}_S\mathbf{a}_S = {}_S\tilde{\mathbf{a}}_S - \mathbf{b}_a - \mathbf{w}_a \tag{8.20}$$

$${}_S\boldsymbol{\omega}_{WS} = {}_S\tilde{\boldsymbol{\omega}}_{WS} - \boldsymbol{b}_\omega - \mathbf{w}_\omega , \tag{8.21}$$

where $\mathbf{w}_a, \mathbf{w}_\omega$ are zero-mean Gaussian noise variables acting on the acceleration and the rotational velocity measurements, respectively. Using the IMU readings, the continuous time

behavior of the system can be described by

$$\dot{\mathbf{q}}_{WM} = \mathbf{w}_g \tag{8.22}$$
$$\dot{\mathbf{q}}_{MS} = -\mathbf{q}_{MS}^{-1}(\mathbf{w}_g) + {}_S\boldsymbol{\omega}_{WS} \tag{8.23}$$
$$\dot{\mathbf{p}}_{MS} = -{}_S\boldsymbol{\omega}_{WS}^{\times}\mathbf{p}_{MS} + {}_S\mathbf{v}_S \tag{8.24}$$
$${}_S\dot{\mathbf{v}}_S = -{}_S\boldsymbol{\omega}_{WS}^{\times}\,{}_S\mathbf{v}_S + {}_S\mathbf{a}_S + \mathbf{q}_{MS}^{-1} \circ \mathbf{q}_{WM}^{-1}({}_W\mathbf{g}) \tag{8.25}$$
$$\dot{\mathbf{b}}_a = \mathbf{w}_{b_a} \tag{8.26}$$
$$\dot{\mathbf{b}}_\omega = \mathbf{w}_{b_\omega}\ , \tag{8.27}$$

where $\mathbf{w}_{b_a}$ and $\mathbf{w}_{b_g}$ are white Gaussian noise processes modeling the time variation of the accelerometer and gyroscope biases and $\omega^{\times}$ is denoting the skew symmetric matrix constructed from the tuple $\boldsymbol{\omega}$. The term ${}_W\mathbf{g}$ denotes the gravity vector in the inertial frame and the Gaussian noise process $\mathbf{w}_g$ models a time variation of the rotation of the map with respect to the inertial frame W. The time continuous equations in our implementation are transformed to a set of discrete prediction equations using an Euler forward integration scheme as proposed in [8] resulting in:

$$\hat{\mathbf{q}}_{WM}^{k+1} = \mathbf{q}_{WM}^{t} \tag{8.28}$$
$$\hat{\mathbf{q}}_{MS}^{k+1} = \mathbf{q}_{MS}^{k} \boxplus (\Delta t\ {}_S\boldsymbol{\omega}_{WS}^{k}) \tag{8.29}$$
$$\hat{\mathbf{p}}_{MS}^{k+1} = \mathbf{p}_{MS}^{k} + \Delta t \mathbf{q}_{MS}^{k}({}_S\mathbf{v}_S^{k}) \tag{8.30}$$
$${}_S\hat{\mathbf{v}}_S^{k+1} = {}_S\mathbf{v}_S^{k} + \Delta t \Big((\mathbf{q}_{WM}^{k} \circ \mathbf{q}_{MS}^{k})^{-1}({}_W\mathbf{g}) + {}_S\mathbf{a}_S - ({}_S\boldsymbol{\omega}_{WS}^{k})^{\times}\,{}_S\mathbf{v}_S^{k} \Big) \tag{8.31}$$
$$\hat{\mathbf{b}}_a^{k+1} = \mathbf{b}_a^{k} \tag{8.32}$$
$$\hat{\mathbf{b}}_g^{k+1} = \mathbf{b}_g^{k}\ . \tag{8.33}$$

Note that in Eq. (8.29) we use the boxplus operator ($\boxplus$), which generalizes the functionality of addition for quantities which are not in a vector space [55]. In this case the $\boxplus$ operator is used on rotations and is briefly outlined in Appendix 7.1. Based on the discrete prediction equations (8.28)-(8.33), we propagate the state covariance as follows:

$$\hat{\boldsymbol{\Sigma}}_{tr}^{k+1} = \mathbf{F}\boldsymbol{\Sigma}_{tr}^{k}\mathbf{F}^T + \mathbf{G}\mathbf{W}\mathbf{G}^T\ , \tag{8.34}$$

where $\mathbf{F}$ is the jacobian of the prediction step with respect to the state, $\mathbf{G}$ is the jacobian with respect to the process noise $\mathbf{w}_*$ and $\mathbf{W}$ is the covariance of the process noise in matrix form. The analytical expressions and the form of $\mathbf{G}$, $\mathbf{F}$ and $\mathbf{W}$ are provided in Appendix 7.2.
For updating the EKF-state, we use projective correspondences to the current position estimates of the MPs as measurements. In order to establish these correspondences, we use the predicted

pose $\hat{\mathbf{q}}_{MS}^{k+1}, \hat{\mathbf{p}}_{MS}^{k+1}$ and project all MPs ${}_M\mathbf{m}_i$ into the camera frame:

$$\mathbf{z}_{proj\,i} = \pi(\mathbf{T}_{CS}\hat{\mathbf{T}}_{MS}^{-1}\,{}_M\mathbf{m}_i)\,, \tag{8.35}$$

where the function $\pi(\cdot)$ denotes the projection function (including distortion). The transformation $\mathbf{T}_{CS}$ between the IMU and the camera is considered to be known and can be obtained from calibration. In order to associate the 2D keypoints to the projected MPs, the descriptors in a small radius around the projection $\mathbf{z}_{proj\,i}$ are matched against the MP's descriptors. After a first data association step, we perform an outlier rejection step using 3D-2D RANSAC [71]. Using the remaining inlier correspondences, the reprojection residuals given by

$$\mathbf{y}_{i,j} = \tilde{\mathbf{z}}_j - \mathbf{z}_{proj_i} \tag{8.36}$$

are constructed, where $\tilde{\mathbf{z}}_j$ is the pixel coordinates of the 2D keypoint j in the image space that was associated with MP i. By stacking all reprojection residuals to form the residual vector $\mathbf{y}$, we can formulate the innovation covariance as

$$\mathbf{S} = \mathbf{H}\hat{\mathbf{\Sigma}}_{tr}\mathbf{H}^T + \mathbf{R}\,, \tag{8.37}$$

where $\mathbf{H}$ is the jacobian matrix of the residual with respect to the EKF state and $\mathbf{R}$ is the measurement covariance obtained by stacking the individual measurement covariance $\mathbf{R}_{i,j}$ associated to $\mathbf{y}_{i,j}$. As the MPs are not part of the EKF state, we use the MP uncertainty $\mathbf{\Sigma}_{p_i}$, estimated as described in Section 4.3, in order to inflate the measurement uncertainty by projecting it onto the image plane:

$$\mathbf{R}_{i,j} = \mathbf{H}_{\mathbf{p}_i}\mathbf{\Sigma}_{p_i}\mathbf{H}_{\mathbf{p}_i}^T + \begin{bmatrix} \sigma_{obs_j}^2 & 0 \\ 0 & \sigma_{obs_j}^2 \end{bmatrix}\,, \tag{8.38}$$

where $\mathbf{H}_{\mathbf{p}_i}$ is the jacobian of the projection of MP i into the image and σ_{obs_j} is the keypoint uncertainty, which in our case is only dependent on the octave that this keypoint was detected. Leveraging the computed innovation covariance, we employ a Mahalanobis distance based outlier rejection in order to exclude additional outliers that slipped through the first RANSAC step. Using the remaining correspondences, the Kalman gain can be computed by

$$\mathbf{K} = \mathbf{H}\hat{\mathbf{\Sigma}}_{tr}\mathbf{H}^T\mathbf{S}^{-1}\,. \tag{8.39}$$

The computed gain is utilized to update the state variables and the associated covariance as follows:

$$\mathcal{X}_{tr} = \hat{\mathcal{X}}_{tr} \boxplus (-\mathbf{K}\mathbf{y}) \tag{8.40}$$

$$\mathbf{\Sigma}_{tr} = (\mathbf{I}_{17\times 17} - \mathbf{K}\mathbf{H})\hat{\mathbf{\Sigma}}_{tr}\,, \tag{8.41}$$

where the notation in Eq. (8.40) indicates that the $\boxplus$ operator is applied for the appropriate states.

4.3 Mapping

The *mapping thread* in the proposed pipeline is responsible to generate new MPs and to maintain an estimate of the uncertainty in their positions.

Initialization of new Map Points

The initialization of new MPs is performed by triangulating of 2D correspondences between the most recent native KF (target) and the KFs that are already in the Map (candidates). As an exhaustive correspondence search against all established KFs would be computationally too expensive, instead, we propose to limit the search to a small subset of candidate KFs. In order to select such a subset, we assign a penalty value s_j to every established KF j based on the relative viewpoint with respect to the target KF i as follows:

$$s_j := \beta_{v,j} \cdot w(\alpha_{v,j}, a_v, b_v, c_v) + \beta_{t,j} \cdot w(\alpha_{t,j}, a_t, b_t, c_t) \ , \tag{8.42}$$

where $\beta_{v,j}, \beta_{t,j}$ are weights chosen such that non-native KFs are preferred in order to establish a stronger constraints across the agents, i.e. the weights for native KFs are chosen 10 times larger. The value $\alpha_{v,j}$ is the angle between the camera axes of KF j and KF i, and $\alpha_{t,j}$ is defined as the triangulation angle between the KFs for a given scene depth. The parameters a_i, b_i, c_i with $i \in \{v, t\}$ are internal parameters of the weighting function w. In order to approximate the unknown scene depth, we compute the median distance of the MPs seen in the most recent couple of KFs. The weighting function $w(x, a, b, c)$ is chosen in the form of a tolerant loss function:

$$\begin{aligned} w(x, a, b, c) &= b \cdot \log\left(1 + \exp\left(\frac{(x-c)^2 - a}{b}\right)\right) - c_0 \\ c_0 &= b \cdot \log\left(1.0 + \exp\left(\frac{-a}{b}\right)\right) \ . \end{aligned} \tag{8.43}$$

Using the candidate KFs with the smallest score, we sequentially perform a brute force descriptor matching followed by a 2D-2D RANSAC-based outlier rejection. For the obtained inlier correspondences we perform a SVD-based linear triangulation to obtain the 3D position of the new map points ${}_M\mathbf{m}_{i,j}$ using the KFs pose estimates $\mathbf{T}_{MS_i}, \mathbf{T}_{MS_j}$. Leveraging the estimated covariances of the KFs poses $\mathbf{\Sigma}_{\mathbf{T}_{MS_i}}, \mathbf{\Sigma}_{\mathbf{T}_{MS_j}}$, we estimate the MP uncertainty $\mathbf{\Sigma}_{{}_M\mathbf{m}_{i,j}}$ as:

$$\mathbf{\Sigma}_{{}_M\mathbf{m}_{i,j}} = (\mathbf{J}_{i,j}\mathbf{W}_{i,j}\mathbf{J}_{i,j})^{-1}\,[1,3;1,3] \ , \tag{8.44}$$

where $\mathbf{J}_{i,j}$ is the jacobian of the reprojected MPs and the poses. The form and the analytical expression of $\mathbf{J}_{i,j}$ is provided in Appendix 7.2. The matrix $\mathbf{W}_{i,j}$ is the corresponding

information matrix and is given by

$$\mathbf{W}_{i,j} = \begin{bmatrix} \frac{1}{\sigma_{obs}^2}\mathbf{I}_{2\times 2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}^{-1}_{\mathbf{T}_{MS_i}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}^{-1}_{\mathbf{T}_{MS_j}} \end{bmatrix} . \tag{8.45}$$

Estimation of the Map Points' uncertainty

As introduced in Section 4.2, we utilize the uncertainty of MPs in the update of the pose tracking. In order to obtain an approximation of the uncertainty for each MP with limited computational effort, we employ an independent EKF per MP. For the initialization of the EKFs' state, we use the triangulated positions along with the covariance as computed by Eq. (8.44). As the EKF's state represents the MP position in M, the prediction step from timestep k to $k+1$ is trivial, as the MPs are assumed to be static:

$$_M\hat{\mathbf{m}}_i^{k+1} = {}_M\mathbf{m}_i^k \ , \tag{8.46}$$

while for the covariance we add diagonal noise to account for missing or removed observations due to the sliding window approach:

$$\hat{\mathbf{\Sigma}}_{m_i}^{k+1} = \mathbf{\Sigma}_{m_i}^k + \sigma_m^2 \cdot \mathbf{I}_{3\times 3} \ , \tag{8.47}$$

where σ_m is the noise parameter modeling the MP's changes in position (e.g. during optimization) and in our implementation was chosen to be $0.2m$. In the update step, we use the re-projection error $\mathbf{y}_i$ of the established 2D-3D correspondences, independent of whether the KF used is a native or not. In order to include the uncertainty of the KF pose into the estimation, a similar operation as in Eq. (8.38) is employed:

$$\mathbf{R}_{\mathbf{p}_{y,i}} = \mathbf{H}_{\mathbf{T}_{MS}} \hat{\mathbf{\Sigma}}_{\mathbf{T}_{MS}} \mathbf{H}^T_{\mathbf{T}_{MS}} + \sigma_{obs}^2 \cdot \mathbf{I}_{2\times 2} \ , \tag{8.48}$$

where the Jacobian $\mathbf{H}_{\mathbf{T}_{MS}}$ is given by

$$\mathbf{H}_{\mathbf{T}_{MS}} = \frac{\partial \mathbf{y}_i}{\partial \mathbf{T}_{MS}} \ . \tag{8.49}$$

With the resulting measurement noise from Eq. (8.48), the innovation covariance $\mathbf{S}_{\mathbf{y}_i}$ can be computed as

$$\mathbf{S}_{\mathbf{y}_i} = \mathbf{H}_{\mathbf{p}_i} \hat{\mathbf{\Sigma}}_{\mathbf{p}_i} \mathbf{H}^T_{\mathbf{p}_i} + \mathbf{R}_{\mathbf{p}_{y,i}} \ . \tag{8.50}$$

Using the resulting Kalman Gain $\mathbf{K}_{\mathbf{y}_i} = \mathbf{\Sigma}_{\mathbf{p}_i} \mathbf{H}_{\mathbf{p}_i} \mathbf{S}^{-1}_{\mathbf{y}_i}$, the updated MP position is given by

$$\mathbf{p}_i = \hat{\mathbf{p}}_i - \mathbf{K}_{\mathbf{y}_i} \mathbf{y}_i \ , \tag{8.51}$$

while the corresponding MP's covariance is given by

$$\mathbf{\Sigma}_{\mathbf{p}_i} = \hat{\mathbf{\Sigma}}_{\mathbf{p}_i} - \mathbf{K}_{\mathbf{y}_i}\mathbf{S}_{\mathbf{y}_i}\mathbf{K}_{\mathbf{y}_i}^T \,. \tag{8.52}$$

Note that the described EKF is mainly employed to estimate the uncertainty of the MPs, therefore, the state update in Eq. (8.51) is only performed until the MP has seen the first update from the Nonlinear Optimization as described in Section 4.4.

4.4 Distributed Optimization Back-End

The optimization back-end constitutes the core element of the system and is responsible for the fusion of the UWB distance-measurements together with the visual measurements and maintaining a consistent estimate among the two agents. In this section, we first introduce the optimization objective assuming a centralized system followed by the undertaken steps to optimize the objective in a decentralized fashion.

Centralized Objective Function

The optimization variables in the back-end consist of the KF poses inside a fixed window size of N and the corresponding M MPs visible in these KFs:

$$\mathcal{X} := [\underbrace{\mathbf{T}_{MS_A}^1, \cdots \mathbf{T}_{MS_A}^N}_{\text{agent A's KFs}}, \underbrace{\mathbf{T}_{MS_B}^1, \cdots, \mathbf{T}_{MS_B}^N}_{\text{agent B's KFs}}, \underbrace{{}_M\mathbf{m}^1, \cdots, {}_M\mathbf{m}^M}_{\text{Map Points}}] \,. \tag{8.53}$$

Over these variables, we can define the optimization objective as:

$$f(\mathcal{X}) := \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{M}(i)} \delta_c \left(\mathbf{y}_{proj_{i,j}}^T \mathbf{W}_r^{i,j} \mathbf{y}_{proj_{i,j}}^T \right) + \sum_{u \in \mathcal{D}} \delta_c \left(\frac{1}{\sigma_d^2} e_d^{u\,2} \right) \,, \tag{8.54}$$

where the set $\mathcal{K}$ indicates all KFs i, native and non-native, that are currently inside the sliding window and accordingly, $\mathcal{M}(i)$ indicates all the MPs that are visible in KF i. The function $\delta_c(\cdot)$ denotes a robust loss function, in our case the Cauchy loss function, introduced to reduce the influence of outliers. The terms $\mathbf{y}_{proj_{i,j}}$ are the reprojection residuals as defined in Eq. (8.35) and (8.36) and the corresponding weights are $\mathbf{W}_r^{k,j} = 1/\sigma_{obs}^2 \cdot \mathbf{I}_{2\times 2}$. The set of relative distance measurements with standard deviation σ_d between the two agents is denoted by $\mathcal{D}$, while the corresponding residual terms are given by

$$e_d^u := \| \mathbf{q}_{MS_A}(t)\mathbf{p}_{U_A} + \mathbf{p}_{MS_A}(t) - \mathbf{q}_{MS_B}(t)\mathbf{p}_{U_B} - \mathbf{p}_{MS_B}(t) \| - d_{meas}^u \,, \tag{8.55}$$

where d_{meas}^u corresponds to the distance measurement taken at time t, and $\mathbf{p}_{U_i}, i \in \{A, B\}$ is the UWB antenna offset expressed in the corresponding IMU frame. The interpolated poses $\mathbf{p}_{MS_i}(t), \mathbf{q}_{MS_i}(t), i \in \{A, B\}$ are computed following (8.17) and Eq. (8.18), while the base poses correspond to the KF poses surrounding the timestamp t.

Distributed Optimization

To avoid a complex synchronization effort between the two agents in order to obtain a unique, common map, we allow each agent to hold its own version of the map and, instead, propose to use an ADMM based distributed optimization scheme to obtain a common trajectory estimate across both agents. In order to optimize $f(\mathcal{X})$ in a distributed fashion, the problem in Eq. (8.54) needs to be brought into the form of Eq. (8.7). Based on the need for both agents' trajectories in (8.55) and the absence of a shared map across the agents, we split the problem by their trajectories, resulting in the following distributed state:

$$\mathcal{X}_i := [\underbrace{\mathbf{T}^1_{MS_i}, \cdots \mathbf{T}^N_{MS_i}}_{\text{native KFs}}, \underbrace{\mathbf{T}^1_{MS_{\ulcorner i}}, \cdots, \mathbf{T}^{N-l}_{MS_{\ulcorner i}}}_{\text{non-native KFs}}, \underbrace{{}_M\mathbf{m}^1, \cdots, {}_M\mathbf{m}^{M_i}}_{\text{Map Points}}], i \in \{A, B\}\,, \tag{8.56}$$

where the notation $\ulcorner i$ indicates the opposite index, i.e. $i = A \Rightarrow \ulcorner i = B$. The parameter $l \geq 0$ is used to represent a lag between the creation of a KF on one agent until it is available to the other one, e.g. caused by network delays. To ensure consistency between the trajectories on both agents, the constraint in Eq. (8.7) for the state as in Eq. (8.56) is given by

$$\mathbf{T}^i_{MS_{A,A}} = \mathbf{T}^i_{MS_{A,B}}, \quad \mathbf{T}^i_{MS_{B,A}} = \mathbf{T}^i_{MS_{B,B}}, \quad \forall i \in [1, N-l]\,. \tag{8.57}$$

Using these constraints, the centralized problem in (8.54) can be written in the form of (8.8)

$$\mathcal{X}_i^{k+1} = \arg\min_{\mathcal{X}_i} f(\mathcal{X}_i) + \sum_{j=1}^{N} \|\mathbf{e}_c^{i,j}\|^2 + \sum_{j=1}^{N-l} \|\mathbf{e}_c^{\ulcorner i,j}\|^2\,, \tag{8.58}$$

where $\mathbf{e}_c$ denotes the consensus error term which is responsible to enforce the constraints in Eq. (8.57). As we perform consensus based on variables in $\mathbb{SO}3$, special care needs to be taken while handling the consensus terms in Eq. (8.8). To do so, we propose to perform the consensus on the tangent space of a fixed reference rotation, leading to a modified version of Eq. (8.57) in terms of the rotation:

$$\delta\mathbf{q}^i_{A,A} = \delta\mathbf{q}^i_{A,B}, \quad \delta\mathbf{q}^i_{B,A} = \delta\mathbf{q}^i_{B,B} \quad \forall i \in [1, N-l]\,, \tag{8.59}$$

where for example $\delta\mathbf{q}^i_{A,A}$ is defined via the following relation

$$\mathbf{q}^i_{MS_{A,A}} = \mathbf{q}^i_{A,ref} \boxplus \delta\mathbf{q}^i_{A,A}. \tag{8.60}$$

The reference rotation $\mathbf{q}^i_{A,ref}$ is fixed and is chosen to be the estimated rotation after the pose tracking. Following this definition, we can write down the terms $\mathbf{e}_c^{i,j}$ as

$$\mathbf{e}^{i,j}_{c,q} = \left(\frac{1}{2\gamma_q}\mathbf{z}_{i,q} + \delta\mathbf{q}_{i,j}\right)\sqrt{2\gamma_q}, \text{ and} \tag{8.61}$$

$$\mathbf{e}^{i,j}_{c,p} = \left(\frac{1}{2\gamma_p}\mathbf{z}_{i,p} + \mathbf{p}_{MS_{i,j}}\right)\sqrt{2\gamma_p}\,. \tag{8.62}$$

Note that inserting these into Eq. (8.58) is equivalent to the formulation in Eq. (8.8), however, as in our implementation we use the Ceres [6] library to solve the minimization, by using the rearranged notation the consensus-errors can be directly inserted. The update of the dual variables as in Eq. (8.9), (8.10) for the rotational part of the poses is again performed in the tangent space of the associated reference rotations

$$\mathbf{z}_{\mathbf{q}_{i,j}}^{k+1} = \mathbf{z}_{\mathbf{q}_{i,j}} - \eta \left((\mathbf{z}_{\mathbf{q}_{i,j}}^{k} + \hat{\mathbf{z}}_{\mathbf{q}_{i,j}}) + \gamma \delta \mathbf{q}_i^{k+1} \right) . \tag{8.63}$$

The translational part is update as defined in Eq. (8.9), (8.10). The obtained dual variables $\mathbf{z}_{i,j}^{k}$ are then passed to the communication module Section 4.5, which transmits the information to the other agent. As the ADMM scheme used is iterative by design and is employed in a sliding window fashion, we continuously run the optimization. Similarly as proposed in [147], we only execute a limited number of iterations in the minimization step. At the end of every minimization step of Eq. (8.58), we shift the sliding window and remove the KFs that fall outside the window and remove MPs that have no observations within the shifted window.

4.5 Communication

The communication module of the proposed pipeline is responsible to communicate newly created data (i.e. KFs) as well as for exchanging updates on the dual variables over a wireless network. In the proposed system we employ ROS [125] to serialize/de-serialize data and communicate it between the agents. Every time a new KF gets inserted in the window, this KF is passed to the Communication module. As sending the full data of this KF, e.g. image information, would lead to tremendous network traffic, we summarize the necessary information into a message containing the following information:

- KF timestamp and KF ID information
- 2D keypoint locations and extraction octave information
- Keypoint descriptors
- Pose $\mathbf{T}_{MS}$ after tracking
- Covariance $\mathbf{\Sigma}_{\mathbf{T}_{MS}}$ of the tracked pose.

Note that the sent pose is not only used as an initial guess, but it also defines the reference rotation used in the pose consensus.
In order to update the consensus error terms, the dual variables associated to the KFs within the optimization window get exchanged after every update step in Eq. (8.9), (8.10). As the dual variables are uniquely assigned to a KF pose, we use the identifier of the associated KF in order to correctly assign the dual variables. In summary, for every performed update on the dual variables, we stack the following information into a message and transmit it to the neighboring agent:

- Identifier of the sender (i.e. the agent ID)

- Vector of dual variables (in 6D)
- Vector of associated KF IDs

Note that the latest received dual variables constitute the $\hat{\mathbf{z}}$ in Eq. (8.8)-(8.10).

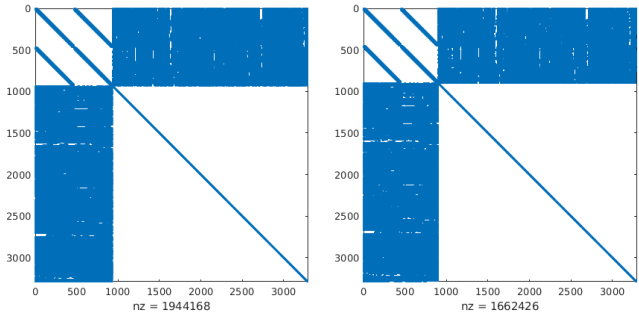
4.6 Initialization

While the previous sections describe the normal mode of operation for the system, an initialization procedure is required in order to build up the initial conditions. As this initialization phase is only active for a limited time, we perform all the necessary computations on a single agent in order to increase the ease the implementation.
To bootstrap the map, based on the timestamp we select the two frames from the agents, which are closest together in time and perform brute force descriptor matching to find 2D-2D correspondences. After a 2D-2D RANSAC outlier rejection, the inlier correspondences are used to compute the relative pose up to scale between the two frames. In order to remove the scale ambiguity, we scale the obtained baseline between the initial frames using the closest available UWB measurement and attempt to triangulate the inlier correspondences to generate MPs. Every subsequent frame (both native and non-native), gets matched and aligned against the existing MPs using a zero-velocity motion model. After the alignment, we attempt the triangulation of new MPs using the newest frame-pair and vision-based bundle adjustment is performed optimizing the frame poses and the MPs. In the case, where we are unable to align a frame, i.e. due to lack of sufficient correspondences, the initial Map is reset and we start with another initial pair again. This process is repeated until the number of frames is sufficient to create at least 4 KFs.
Once the minimal number of KFs is reached, we aim to initialize the IMU related variables for the tracking, namely $\mathbf{q}_{WM}$, ${}_S\mathbf{v}$ and $\mathbf{b}_\omega$. For this, we utilize use a method inspired by the IMU-initialization proposed by [111], which first estimates the gyroscope biases $\mathbf{b}_\omega$ and then solves for velocity states and gravity direction in a least-squares fashion. As we perform the initialization for two trajectories, we jointly solve for the velocities of both agents, but only a single, shared gravity direction. Here, in contrast to [111], as we utilize the UWB measurements, we are able to avoid including the scale as an estimation parameter. After successful initialization, the agent responsible for the computations sends the all of its native KFs along with the MPs and the IMU states to the other agent, and from then on the system enters the normal mode of operation as outlined in the previous sections.

5 Experimental Evaluation

5.1 Run-time efficiency of combined Z-spline interpolation

In this section, we investigate the influence of the trajectory representation proposed here using a combination of KF poses and a Z-spline based interpolation along these poses on the run-time efficiency. As, computationally, the most expensive part in the proposed system is the minimization of Eq. (8.58), reducing the execution time of this minimization permits the execution of more distributed optimization iterations, which in return potentially boosts the accuracy of the estimates.



(a) B-spline **(b)** Combined Z-spline

Figure 8.4: Comparison between the resulting Hessian matrix for the same problem when using standard B-spline based interpolation (a) versus the usage of the proposed combined Z-spline based KF-interpolation (b). While the structure is very similar in both cases, the non-zero elements ("nz") are reduced by 15% in (b)

As the minimization in Eq. (8.58) is performed using a second order method, the iteration time is, to a large extent, determined by the time required to solve the resulting normal equations, which in turn is highly depending on the sparsity of the resulting Hessian matrix. As a result, we investigate the influence of the proposed trajectory representation on the sparsity of the resulting Hessian matrix. For the relative distance error terms in Eq. (8.55) there is no structural difference between the combined Z-spline based interpolation and the standard B-spline interpolation. On the other hand, for the reprojection errors, the Jacobian structure changes significantly. While computing the error using the traditional B-spline based interpolation four base poses have a non-zero contribution in the Jacobian, utilizing the KFs as base poses for a Z-spline based interpolation, one can directly compute the reprojection error using the KF-pose. As a result, the number of non-zero elements in the underlying Jacobian matrix is roughly $1/4$ (as the number of residuals is dominated by the reprojection errors), compared to the standard B-spline interpolation.
Using a sample over a trajectory length of 8 seconds with a KF interval of 0.15 seconds and roughly 2000 MPs we build up the normal equation for the same problem, both using B-spline interpolation as well as the proposed interpolation scheme and extract the resulting Hessian matrices. In Fig. 8.4 the corresponding Hessians with the number of non-zero elements are shown. While the structure itself is similar in both cases, one can see that the proposed Z-spline based formulation improves the sparsity of the resulting Hessian matrix by around 15%. The sparsity remains after performing factorization, indicating that the proposed interpolation scheme indeed reduces the computational complexity in the solving step.

		mean $\pm 2\sigma$ [ms]	max [ms]	rate [Hz]
Tracking	BRISK Extraction	7.9 ± 3.4	20.8	20
	Matching	0.5 ± 0.4	8.7	
	Tracking EKF	1.4 ± 1.3	12.0	
	Total Latency	10.3 ± 4.2	25.0	
Mapping	Uncertainty EKF	0.2 ± 0.1	1.4	6.7
	BF-Matching	8.3 ± 8.5	26.1	1.6
	Triangulation	0.1 ± 0.5	1.8	1.6

Table 8.1: Runtime evaluation of the different parts of the tracking and mapping, recording the mean $\pm$ twice the standard deviation of the respective measurement, and the rates of the corresponding operations performed on average.

5.2 Runtime and Bandwidth Evaluation

As real-time capabilities and data sharing rates are crucial elements for the applicability of a distributed system, in this section we analyze the run-time of the main elements of the proposed system and investigate the bandwidth requirements for the necessary data exchange. In order to evaluate the system under realistic conditions, all the experiments are performed using two Intel Core i7-8550U computers communicating with each other via a wireless TP-Link AC1750 router configured to use the 2.4GHz interface.

Runtime Evaluation

As the proposed system runs multiple threads, the decisive element for the real-time capability of the overall system is the tracking-thread, which needs to process every frame. Nonetheless, the timings of the other parts, i.e. mapping and optimization, are important for the overall performance of the system and thus are measured. For example, a slow optimization step will result in fewer ADMM-iterations, possibly leading to bigger inconsistencies and inaccuracies. In Table 8.1, the timings for the main tracking- and mapping processes are reported. On the tracking side, the most expensive operation is the extraction of the BRISK features with approximately 8ms per frame, while the matching-step and EKF estimation roughly take up to 2ms per frame. The total latency describes the effective time it takes from the reception of the image until the pose estimate is computed, which takes on average 10ms. Even in the worst case recorded, the latency stays within a 25ms budget, which corresponds to twice real-time. The timings for mapping are dominated by the correspondence search, while the time for uncertainty maintenance and triangulation is almost neglectable. Note that the EKF based uncertainty maintenance of the MPs gets called for every KF, which in our case corresponds to every third frame, whereas the matching and triangulation processes only get called if new MPs need to be inserted. The optimization loop, which includes communicating and updating the dual variables, takes approximately 77ms on average with a standard deviation of 11.5ms. With a KF-interval of 150ms, this means on average per KF, two ADMM-iterations are performed.

Bandwidth Usage

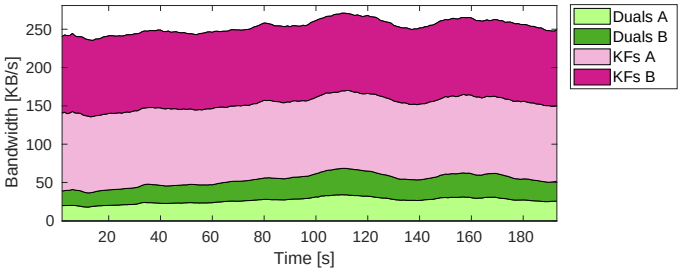
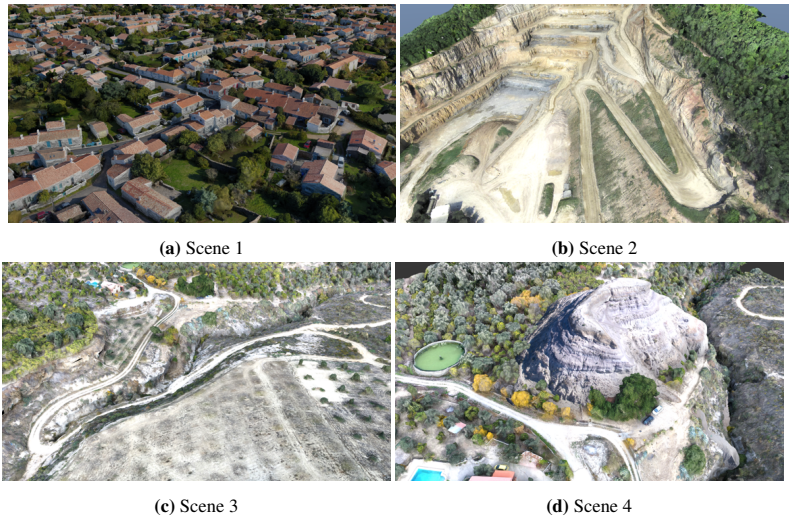


Figure 8.5: Bandwidth usage over time for the different messages and agents during a single experiment. The total bandwidth requirement is approximately 250KB/s, which is easily possible using e.g. standard WiFi communication.

In the proposed system, the data exchanged consists of KF-data and the dual variables exchanged in the ADMM scheme. As we use ROS for the communication, the built-in functionality for monitoring bandwidth usage was used to obtain the data presented in this section. Fig. 8.5 shows an example of the required bandwidth for one of the datasets used in Section 5.4. As it can be observed, the total bandwidth requirement remains generally constant around 250KB/s, whereas the majority of the exchanged data is contained in the KFs with approximately 100KB/s per agent, while the exchange of the dual variables generates about 25KB/s of network traffic per agent. Note that while on the basis of the fixed optimization window of 5s, the package size for the exchange of the dual variables is constant, the rate of the exchange is coupled with the frequency of the optimization-loop, which is subject to fluctuations. A summary of the bandwidth usage over a larger set of experiments is provided in Table 8.2. As it can be observed, the overall bandwidth usage is not subject to large fluctuations and on average is below 250KB/s, which is easily feasible with a standard WiFi module (e.g. IEEE 802.11g standard).

	mean $\pm 2\sigma$ [KB/s]	**max [KB/s]**
Dual Variables	48.6 ± 14.0	68.5
Keyframes	197.3 ± 8.8	214.5
Total	245.8 ± 19.1	270.9

Table 8.2: Bandwidth usage of the different message types summed together for both agents, recording the mean $\pm$ twice the standard deviation for the corresponding measurements.



(a) Scene 1 **(b)** Scene 2

(c) Scene 3 **(d)** Scene 4

Figure 8.6: Snapshots of the four photo-realistic scenes used to render experimental data for the evaluation of the proposed system

5.3 Photo-Realistic Synthetic Datasets

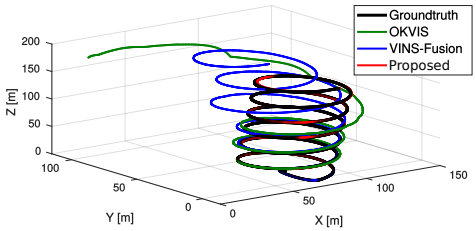
Performing outdoor experiments with two UAVs flying relatively close to each other at high altitudes is extremely challenging as at that altitude the pilot has very limited visual understanding of the motion of the UAVs, posing a significant risk of losing control of the aircraft. Furthermore, obtaining ground-truth of the robots' poses allowing a quantitative evaluation of the proposed framework on real data is problematic due to fluctuations in the accuracy of GPS measurements and the challenging estimation of the orientation. As a result, inspired by the idea in [135], we create synthetic photorealistic datasets from real images to test and evaluate the proposed approach. Analogously to [135], we simulate both UAVs' dynamics using the RotorS Gazebo simulator [48] and utilize the Blender render engine to generate the associated image data from 3D models obtained by photogrammetric reconstruction[1]. The visual-inertial sensor data was simulated to mimic the data-stream obtained from a sensor as in [101], consisting of a global-shutter grayscale image-stream, time-synchronized with the IMU data. The images are rendered with a resolution of 480×752 at 20Hz and the IMU data has a rate of 200Hz. The UWB-distance measurements are simulated by computing the ground-truth relative distance between the UAVs and disturb it with gaussian noise. The UWB-data is simulated at 60Hz with a noise level of 0.1m standard deviation. The simulated UAVs have a maximal

[1] https://github.com/VIS4ROB-lab/visensor_simulator

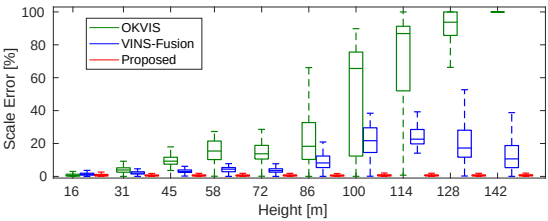
diameter (from tip-to-tip) of 0.85m and during all experiments, we set the smallest allowed baseline to be 1.0m.
In order to control the relative pose between the two UAVs, we employ the control strategy as outlined in Section 5.5, while the estimated scene depth is obtained by computing the median depth from a rendered depth image. We use four different scenes, as shown in Fig. 8.6, and rendered multiple trajectories for each scene.
Scene 1: A suburban housing settlement consisting of smaller houses with gardens, small parks and connecting streets spanning an area of approximately 450m $\times$ 450m. The, from high altitudes, plane-like structure along with the well textured scene, results in well suited imagery for vision based methods.
Scene 2: A mine with different levels of erosion, small trails and some parts with steep flanks, resulting in abrupt changes in the scene depth of up to 35m. The steep flanks in combination with the, at times, uniform texture prohibit lengthy and uniform feature association.
Scene 3: Mediterranean countryside with sections containing different kinds of vegetation as well as some small canyons and trails offering a mixture of texture-rich and low-textured areas along with some depth variation across the canyons.
Scene 4: Same structure as Scene 3 but with an artificially added hill structure of about 40m height in order to increase the depth variations in the scene.

5.4 Comparison to Visual-Inertial SLAM at higher altitudes

In this section, we provide a comparison of the proposed approach against the two highest performing state-of-the-art stereo VIO methods that are publicly available, namely VINS-Fusion [111] and OKVIS [83], to test their accuracy of estimates at increasing scene depths. To illustrate this effect, we generated spiral-shaped trajectories over Scenes 1-3 with gradually increasing altitude. The trajectories have a radius of 25 meters and increase their height with a rate of 25 meters per turn up to a height of approximately 160 meters. The fixed stereo camera baseline for VINS-Fusion and OKVIS was chosen to be 0.22m, which was chosen to be twice the size of the sensor proposed in [101].
An example of such a trajectory along with the aligned estimates of both our system and the stereo-inertial estimators is shown in Fig. 8.7a. Initially, all the estimates are close to the ground-truth, however, with increasing height, both estimates of VINS-Fusion as well as OKVIS start to diverge with increasing altitude, while the estimate of the proposed system remains close to the ground-truth trajectory. As it can be observed, the trajectories of the stereo VIOs are mainly fluctuating in scale and less in the shape of the trajectory itself. In order to evaluate the scale uncertainty, we perform an evaluation in a similar fashion to a relative error evaluation as described in [148]. We select a sub-trajectory of a given length (here 100 frames), align it to the ground-truth with a similarity transform as computed using the approach in [3] and record the scale. The selected sub-trajectory is then shifted by 5 frames and the alignment is repeated until the end of the trajectory is reached, resulting to statistics of the scale error. Using the altitude as an approximation of the scene depth, in Fig. 8.7b, the scale-error statistics with respect to the altitude are reported. As it can be observed, with increasing altitude, the scale errors quickly become significant resulting in a large uncertainty of the estimate. While VINS-Fusion generally shows a slower increase of the scale error than OKVIS, both VIO methods suffer from the same tendency with increased scene depth, whereas the proposed method



(a) Example run for a spiral trajectory on Scene 1



(b) Scale error for different altitudes

Figure 8.7: Evaluation results for spiral trajectories with increasing altitudes. In (a) is the result of a single run, where the estimates of both stereo VIOs start to diverge with increasing altitude, while (b) summarizes the scale errors over all performed spiral experiments, where it is evident that the effectiveness of the imposed scale constraints becomes an issue for (stereo-) VIO methods with increasing altitude.

exhibits a constant scale uncertainty over the full altitude range.
In order to allow for a quantitative comparison between our proposed approach to the two state-of-the-art stereo VIOs, different trajectories at altitudes ranging between $15 - 35$ meters are rendered. In particular two different trajectories are created for Scenes 1-3 summing up to a total trajectory length of 3.4km. For the proposed method, the baseline between the two agents is chosen to be roughly 2m, while this is allowed to fluctuate slightly without considering the particular scene depth. The resulting relative comparison between the relative odometry errors of the stereo VIO methods and the proposed method is shown in Fig. 8.8. As it can be observed, compared to the existing VIO methods, the proposed system has both a lower median error as well as smaller fluctuations in the statistics, as expected, due to the fact that the selected baseline should exhibit more favorable behavior for the overall scene depth. Between the two existing stereo VIOs, OKVIS performs slightly better than VINS-Fusion on the evaluated datasets, whereas to a large extent the increased error of the latter can be traced back to an increased yaw-drift. Note that the evaluated datasets are closer to the intended use-cases of the

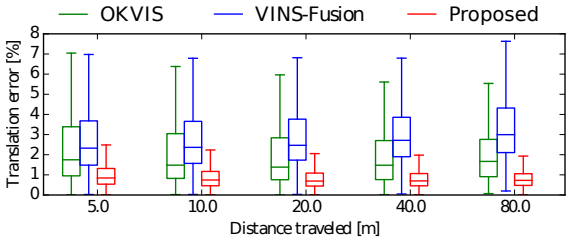


Figure 8.8: Comparison of the relative odometry error between state-of-the-art stereo VIO methods and the proposed approach. The reported errors are obtained by computing the error statistics over all datasets over 3 runs.

proposed system, i.e. scene depths above 10m, than the VIOs, which are generally designed and evaluated on datasets closer to the ground or indoors. Hence, we do not claim to outperform these systems in the general case, however, we can show the advantage of the proposed approach already at flying altitudes significantly below the heights where the stereo VIO systems start to fail.

5.5 Active Baseline Control

As the proposed collaborative system estimates the poses of the agents in a common reference frame, we are able to control the relative distance between the UAV agents. For this purpose, we designed a high-level controller with the goal of controlling the agents' baseline in order to form a virtual stereo camera, as illustrated in Fig. 8.9. For simplicity, we assume that the monocular cameras are mounted at an identical viewing angle onboard each agent and that aligning the agents along their X-axes results to a valid stereo configuration. Note that with a few additional computations, this approach can be adapted to a more generic setup, such as having different mounting angles of the cameras on the UAVs, as well. The relative translation between the cameras in the virtual stereo setup is given by the average of the two agents' poses:

$$\mathbf{p}^k_{WV} = \frac{1}{2}\left(\mathbf{p}^k_{WS_A} + \mathbf{p}^k_{WS_B}\right) . \tag{8.64}$$

In order to obtain the yaw angle, we first project the relative baseline onto the X-Y plane of the inertial frame

$$\mathbf{p}_{proj} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\mathbf{p}^k_{WS_B} - \mathbf{p}^k_{WS_A}\right) \tag{8.65}$$

and from that we compute the resulting orientation as:

$$\psi_{WV} = \operatorname{atan2}\left(\mathbf{p}_{proj,y}, \mathbf{p}_{proj,x}\right) + \pi/2 \, , \tag{8.66}$$

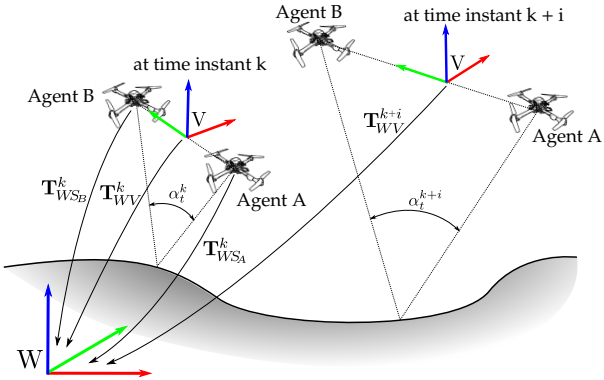


Figure 8.9: Illustration of the virtual stereo setup with two agents. Using the desired location of the virtual stereo center V and a target triangulation angle α_t, the required agent poses are computed.

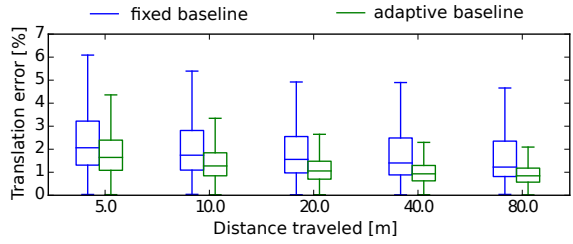
where $\mathbf{p}_{proj,x}$ corresponds to the first and $\mathbf{p}_{proj,y}$ to the second entry of $\mathbf{p}_{proj}$.
For a given 4-DoF target pose (x_t, y_t, z_t, ψ_t) and the scene depth d_s, suitable poses for the agents can be computed by reformulating Equations (8.65), (8.66). Using the desired triangulation angle α_t (here set to $10°$) and d_s, the resulting baseline of the virtual stereo camera is given by

$$b_V = 2 \cdot \tan(\alpha_t/2) \cdot d_s \ . \tag{8.67}$$

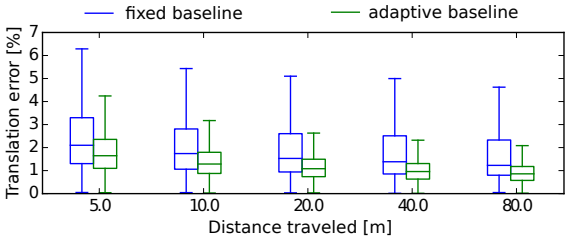
Hence, the resulting agents' target translations are given by

$$\mathbf{p}_{WS_{A/B}} = \begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} + \begin{bmatrix} \cos(\psi_t) & \sin(\psi_t) & 0 \\ \sin(\psi_t) & \cos(\psi_t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \mp b_V/2 \\ 0 \end{bmatrix} , \tag{8.68}$$

where the minus sign corresponds to agent A and the plus sign to agent B, assuming that agent A acts as the right camera in the virtual stereo setup. The target yaw angles of the agents are set to the target yaw angle of the virtual stereo camera. The computed target poses for both agents are fed to the MPC-based position controller running on each agent [62]. Note that in order to limit the speed of the response, for target setpoints that are far away, we linearly interpolate between the current and the target pose, such that the intermediate goal is only a limited distance away from the current pose.



(a) agent A



(b) agent B

Figure 8.10: Comparison of the relative translation error over all runs on all datasets. The results for agent A are shown in (a), whereas (b) shows the corresponding errors for agent B.

5.6 Fixed vs. Adaptive Virtual Stereo Baseline

In this section, we evaluate the influence of actively adapting the virtual stereo baseline, i.e. a fixed α_t, between the two UAVs against maintaining a fixed target distance between the two aircraft. As this generally cannot be achieved using the exact same trajectories, for every dataset, we create two versions; one using a fixed baseline of 2m and another, where the baseline gets constantly adjusted to achieve a triangulation angle α_t (of 10°). However, the waypoints as well as the simulation parameters are otherwise chosen to be identical. The estimator parameters in both versions are identical. In particular, the KF interval is set to 0.15 seconds and the trajectory horizon to 5.0 seconds for all the experiments in this section. In total, five datasets are generated on the four scenes shown in Fig. 8.6, where the waypoints are chosen to follow mostly exploratory paths with some height variation, leading to scene depths ranging between $25 - 120$m. The resulting error statistics of the obtained odometry estimates for both cases are summarized in Fig. 8.10. As it can be observed, the position drift is significantly reduced when adapting the baseline versus having a fixed baseline. The relative translation error of the fixed baseline approach ranges between $1.8 - 2.5\%$, while using an adaptive baseline the error can be reduced to $0.9 - 1.9\%$, corresponding to up to a twofold reduction.

	Fixed Baseline						Adaptive Baseline					
	agent A		agent B		combined		agent A		agent B		combined	
	RMSE [m]	Scale [%]	RMSE [m]	Scale [%]	RMSE [m]	Scale [%]	RMSE [m]	Scale [%]	RMSE [m]	Scale [%]	RMSE [m]	Scale [%]
Scene 1 - 1	3.73	2.28	3.68	2.31	3.70	2.29	2.16	0.66	2.10	0.66	2.13	0.66
Scene 2 - 1	0.92	0.39	0.92	0.39	0.92	0.39	0.81	0.14	0.82	0.15	0.81	0.14
Scene 3 - 1	1.54	1.19	1.51	1.16	1.52	1.17	0.48	0.33	0.48	0.32	0.48	0.32
Scene 3 - 2	1.79	1.63	1.79	1.66	1.79	1.64	0.77	0.24	0.74	0.25	0.76	0.24
Scene 4 - 1	2.71	1.95	2.73	1.92	2.72	1.93	0.88	0.67	0.89	0.71	0.89	0.69

Table 8.3: Comparison of the absolute trajectory errors when using a fixed baseline versus using the adaptive baseline control scheme. All the the reported values are obtained as average over 3 runs. To illustrate the consistency of the distributed approach, the errors of the individual agents (aligned individually) as well as the error of the combined trajectory are reported.

Comparing the global Root Mean Squared Error (RMSE) in the agents' trajectories on the datasets presented in Table 8.3, the differences are clearly visible and are, at times, up to a factor of 3. However, as it can be seen in Table 8.3, the fixed baseline approach mainly suffers from worse scale estimates, which in return, leads to increased RMSEs on the trajectory. This is not surprising and supports our thesis that the correct scale estimation becomes a crucial element for robust and stable pose estimation at high altitudes. For the generated datasets used here, the fixed stereo baseline of 2m, is rather small compared to the scenes' depth, which are to a large extent, given by the flight altitude, and therefore, the scale estimation becomes more uncertain than for larger baselines. On the other hand, on the dataset of Scene 2, no significant difference between the adaptive and the fixed baseline approach can be observed, which can be explained by the fact, that the corresponding scene has more depth variations to it, which leads to having some close (e.g. side walls), but also some farther away parts of the scene (e.g. ground) in the view overlap, which results in smaller differences between the adaptive baseline and the fixed one.
Besides the advantage of actively controlling the baseline between the agents, also the consistency between the estimates of the two agents is indicated in Table 8.3. To illustrate this, we report the global RMSE of the individual agents aligning their trajectories independently, as well as the resulting errors when aligning both trajectories treating the two trajectories as a single one. If the two trajectories were inconsistent with respect to each other, an increase in the combined trajectories error should be observed, however, throughout all datasets this is not the case, indicating the consistency of the distributed estimate. Note that there is no noticeable difference in the consistency between the adaptive and the fixed baseline approach, which indicates that the distributed approach works reliably even when the estimate itself becomes more uncertain (i.e. fixed baseline).

6 Conclusion

In this article, we present a novel framework using two UAVs, equipped with one IMU and one monocular camera each, while measuring the relative distance between them using an Ultrawideband module in order to compute the 6DoF pose estimation for both UAV in real-time to estimate the scene in a virtual-stereo setup. The pipeline is implemented in a decentralized

fashion allowing each agent to hold its own estimate of the map, enabling low-latency pose estimation that does not depend on network delays. In order to ensure consistency across the agents, a consensus based optimization scheme is employed. Using the two agents as a virtual stereo camera with adjustable baseline, potentially large scene depths can be handled, which are problematic for existing VIO system using monocular or (fixed-baseline) stereo rigs.
A thorough experimental analysis using synthetic, photorealistic data reveals the ability of the proposed approach to reliably estimate the pose of the agents even at high altitudes. Using trajectories with increasing altitudes, we show the problematic behavior of existing the stereo VIO methods at high altitudes, while the proposed approach is able to maintain high quality estimation of both VIO agents' poses. Furthermore, the comparison with state-the-art stereo inertial methods demonstrates, that the proposed approach already proves advantageous in terms of estimation accuracy at altitudes marginally higher than 15m. Employing a simple formation controller, which adjusts the baseline between the two agents depending on the observed scene depth, the benefit of having the ability to adjust the stereo baseline on-the-fly is be demonstrated, achieving a nearly twofold reduction in the pose estimation error compared to a fixed target baseline.
The applicability of the approach is verified by reporting practical timing and bandwidth measurements. Owing to the decentralized approach, the proposed system achieves an average latency of 11ms for the pose tracking. The required bandwidth of the overall system remains under 250KB/s and therefore, can easily be handled by a standard WiFi module.
Future work includes employing the system in the field and perform extensive outdoor experiments. Furthermore, it would be interesting to investigate non-uniform KF intervals, to dynamically adjust them depending on the performed motions. Also the investigation of the proposed system to be used as virtual stereo-camera for 3D reconstruction would be highly interesting as this potentially allows to quickly obtain a coarse reconstruction of a large areas.
Following the demonstration of the ability of the proposed method to drastically increase the fidelity of estimates at high flying altitudes using two UAV agents, future work will aim to leverage the power of the asynchronous estimation framework proposed to scale up to bigger number of agents in the air. In this way, we aim to push towards tightly collaborating multi-agent SLAM in a distributed architecture.

7 Appendix

7.1 Rotation Calculus

First we introduce the exponential map, mapping a vector $\boldsymbol{\varphi} \in \mathbb{R}^3$ to a rotation $\mathbf{q}$

$$\mathbf{q} = \exp(\boldsymbol{\varphi}) = (q_0, \breve{\mathbf{q}}) = \left(\cos(\|\boldsymbol{\varphi}\|/2), \sin(\|\boldsymbol{\varphi}\|/2) \frac{\boldsymbol{\varphi}}{\|\boldsymbol{\varphi}\|} \right) , \tag{8.69}$$

where q_0 is the real-part and $\breve{\mathbf{q}}$ is the imaginary part of the quaternion $\mathbf{q}$. The inverse mapping, the logarithmic map, maps a quaternion $\mathbf{q}$ to its corresponding tangent vector $\boldsymbol{\varphi}$:

$$\boldsymbol{\varphi} = \log(\mathbf{q}) = 2\mathrm{atan2}\left(\|\breve{\mathbf{q}}\|, q_0\right) \frac{\breve{\mathbf{q}}}{\|\breve{\mathbf{q}}\|} . \tag{8.70}$$

Using these mappings, boxplus and boxminus operations adopting the functions of addition and subtraction [55] can be constructed as follows:

$$\boxplus : SO(3) \times \mathbb{R}^3 \to SO(3), \tag{8.71}$$
$$\mathbf{q}, \boldsymbol{\varphi} \mapsto \mathbf{q} \circ \exp(\boldsymbol{\varphi})$$
$$\boxminus : SO(3) \times SO(3) \to \mathbb{R}^3 \tag{8.72}$$
$$\mathbf{q}_1, \mathbf{q}_2 \mapsto \log(\mathbf{q}_2^{-1} \circ \mathbf{q}_1) \,,$$

where $\circ$ indicates the concatenation of two quaternions.

7.2 Pose Tracking and Map Point EKFs

The state transition jacobian used in Eq. (8.34) is given by:

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \dfrac{\partial \hat{\mathbf{q}}_{MS}^{k+1}}{\partial \mathbf{q}_{MS}^{k}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dfrac{\partial \hat{\mathbf{q}}_{MS}^{k+1}}{\partial \mathbf{b}_g^k} \\ \dfrac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{\partial \mathbf{q}_{WM}^{k}} & \dfrac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{\mathbf{q}_{MS}^{k}} & \mathbf{0} & \dfrac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{{}_S\mathbf{v}^{k}} & -\Delta t \mathbf{I}_{3\times 3} & \Delta t \left[{}_S\mathbf{v}^{k} \right]^{\times} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times 3} & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_{3\times 3} \end{bmatrix} \tag{8.73}$$

with

$$\frac{\partial \hat{\mathbf{q}}_{MS}^{k+1}}{\partial \mathbf{q}_{MS}^{k}} = \mathbf{R}(\exp(\Delta t \, {}_S\boldsymbol{\omega}_{WS}^{k}))^T \tag{8.74}$$
$$\frac{\partial \hat{\mathbf{q}}_{MS}^{k+1}}{\partial \mathbf{b}_g^{k}} = -\Delta t \Gamma(\Delta t \, {}_S\boldsymbol{\omega}_{WS}) \tag{8.75}$$
$$\frac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{\partial \mathbf{q}_{WM}^{k}} = \Delta t \mathbf{R}(\mathbf{q}_{MS}^{k})^T \left[\mathbf{R}(\mathbf{q}_{WM}^{k})^T \mathbf{g} \right]^{\times} \mathbf{J}_{rp}(\mathbf{q}_{WM}^{k}) \tag{8.76}$$
$$\frac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{\mathbf{q}_{MS}^{k}} = \Delta t \left[\mathbf{R}(\mathbf{q}_{MS}^{-1} \circ \mathbf{q}_{WM}^{-1}) \mathbf{g} \right]^{\times} \tag{8.77}$$
$$\frac{\partial\, {}_S\hat{\mathbf{v}}^{k+1}}{{}_S\mathbf{v}^{k}} = \mathbf{I}_{3\times 3} - \Delta t \left[{}_S\boldsymbol{\omega}_{WS} \right]^{\times} \,, \tag{8.78}$$

where $\Gamma(\cdot)$ is the jacobian of the exponential map given by

$$\Gamma(\boldsymbol{\varphi}) = \mathbf{I}_{3\times 3} - \frac{1 - \cos(\|\boldsymbol{\varphi}\|)}{\|\boldsymbol{\varphi}\|^2} [\boldsymbol{\varphi}]^{\times} + \frac{\|\boldsymbol{\varphi}\| - \sin(\|\boldsymbol{\varphi}\|)}{\|\boldsymbol{\varphi}\|^3} ([\boldsymbol{\varphi}]^{\times})^2 \tag{8.79}$$

and $\mathbf{J}_{rp}(\mathbf{q})$ denotes the jacobian of the local roll-pitch parameterization and is given by

$$\mathbf{J}_{rp}(\mathbf{q}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(roll(\mathbf{q})) & -\sin(roll(\mathbf{q})) \end{bmatrix}^T . \tag{8.80}$$

The function $roll(\mathbf{q})$ extracts the roll angle from a quaternion $\mathbf{q}$.
Analogously, the jacobian matrix of the prediction noise can be described as

$$\mathbf{G} = \begin{bmatrix} \Delta t \mathbf{I}_{2\times 2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \Delta t \Gamma({}_S\boldsymbol{\omega}_{WS}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \Delta t \mathbf{R}(\mathbf{q}^k_{MS}) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \Delta t \mathbf{I}_{3\times 3} & \Delta t \left[{}_S\mathbf{v}^k \right]^\times & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \Delta t \mathbf{I}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \Delta t \mathbf{I}_{3\times 3} \end{bmatrix} \tag{8.81}$$

The corresponding noise matrix $\mathbf{W}$ can be written as

$$\mathbf{W} = \begin{bmatrix} \sigma_g^2 \mathbf{I}_{2\times 2} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_v^2 \mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_a^2 \mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_\omega^2 \mathbf{I}_{3\times 3} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{b_a}^2 \mathbf{I}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{b_g}^2 \mathbf{I}_{3\times 3} \end{bmatrix}, \tag{8.82}$$

where σ_i represents the discrete time noise of the corresponding variable i.
The measurement jacobian as used in Eq. (8.37) for a single observation of the MP i is given by:

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{0}_{2\times 2} & \frac{\partial \pi ({}_C\mathbf{m}_i)}{\partial \hat{\mathbf{q}}^{k+1}_{MS}} & \frac{\partial \pi ({}_C\mathbf{m}_i)}{\partial \hat{\mathbf{p}}^{k+1}_{MS}} & \mathbf{0}_{2\times 9} \end{bmatrix} , \tag{8.83}$$

where

$$\frac{\partial \pi \left({}_C\mathbf{m}_i\right)}{\partial \hat{\mathbf{q}}^{k+1}_{MS}} = -\frac{\partial \pi \left({}_C\mathbf{m}_i\right)}{\partial \, {}_C\mathbf{m}_i} \mathbf{R}(\mathbf{q}_{CS}) \left[{}_S\mathbf{m}_i \right]^\times \tag{8.84}$$

$$\frac{\partial \pi \left({}_C\mathbf{m}_i\right)}{\partial \hat{\mathbf{p}}^{k+1}_{MS}} = \frac{\partial \pi \left({}_C\mathbf{m}_i\right)}{\partial \, {}_C\mathbf{m}_i} \mathbf{R}(\mathbf{q}_{CS}) \mathbf{R}(\hat{\mathbf{q}}^{k+1}_{MS})^T , \tag{8.85}$$

where $\frac{\partial \pi ({}_C\mathbf{m}_i)}{\partial \, {}_C\mathbf{m}_i}$ is the jacobian of the camera model including the lens distortion.

The jacobian $\mathbf{J}_{i,j}$ used to compute the MP uncertainty in Eq. (8.44), is given by

$$\mathbf{J}_{i,j} = \begin{bmatrix} \frac{\partial \pi\left({}_{C_i}\mathbf{m}\right)}{\partial {}_{C_i}\mathbf{m}} \frac{\partial {}_{C_i}\mathbf{m}}{\partial {}_{M}\mathbf{m}} & \frac{\partial \pi\left({}_{C_i}\mathbf{m}\right)}{\partial {}_{C_i}\mathbf{m}} \frac{\partial {}_{C_i}\mathbf{m}}{\partial \mathbf{T}_{MS_i}} & \mathbf{0}_{2\times 6} \\ \frac{\partial \pi\left({}_{C_j}\mathbf{m}\right)}{\partial {}_{C_j}\mathbf{m}} \frac{\partial {}_{C_j}\mathbf{m}}{\partial {}_{M}\mathbf{m}} & \mathbf{0}_{2\times 6} & \frac{\partial \pi\left({}_{C_j}\mathbf{m}\right)}{\partial {}_{C_j}\mathbf{m}} \frac{\partial {}_{C_j}\mathbf{m}}{\partial \mathbf{T}_{MS_j}} \\ \mathbf{0}_{6\times 3} & \mathbf{I}_{6\times 6} & \mathbf{0}_{6\times 6} \\ \mathbf{0}_{6\times 3} & \mathbf{0}_{6\times 6} & \mathbf{I}_{6\times 6} \end{bmatrix}, \tag{8.86}$$

where again the $\frac{\partial \pi({}_{C}\mathbf{m})}{\partial {}_{C}\mathbf{m}}$ corresponds to the projection model. The remaining terms are given by

$$\frac{\partial {}_{C_k}\mathbf{m}}{\partial {}_{M}\mathbf{m}} = \mathbf{R}_{CS}\mathbf{R}_{MS_k}^{T} \tag{8.87}$$

$$\frac{\partial {}_{C_k}\mathbf{m}}{\partial \mathbf{T}_{MS_k}} = \begin{bmatrix} \left[\mathbf{R}_{MC}^{T}({}_{M}\mathbf{m} - \mathbf{p}_{MC})\right]^{\times} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & -\mathbf{R}_{MC}^{T} \end{bmatrix}, \tag{8.88}$$

where $k \in [i, j]$.

Bibliography

[1] Azure Spatial Anchors. https://azure.microsoft.com/en-us/services/spatial-anchors/. Accessed: 2020-05-21.

[2] Oculus Quest VR-Headset. https://www.oculus.com/quest/. Accessed: 2020-05-21.

[3] Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(4):376–380, 04 1991.

[4] AEROWORKS: Collaborative Aerial Workers. url http://www.aeroworks2020.eu, 2015.

[5] M. W. Achtelik, S. Weiss, M. Chli, F. Dellaert, and R. Siegwart. Collaborative Stereo. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 2242–2248, 2011.

[6] S. Agarwal, K. Mierle, and Others. Ceres Solver. http://ceres-solver.org.

[7] J.-L. Blanco, J. González-Jiménez, and J.-A. Fernández-Madrigal. Sparser relative bundle adjustment (srba): constant-time maintenance and local optimization of arbitrarily large maps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, may 2013.

[8] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017.

[9] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. ROVIO: Robust Visual Inertial Odometry Using a Direct EKF-Based Approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[10] M. Bryson and S. Sukkarieh. Observability analysis and active control for airborne SLAM. *IEEE Transactions on Aerospace and Electronic Systems*, 44(1):261–280, 2008.

[11] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart. The EuRoC micro aerial vehicle datasets. In *International Journal of Robotics Research (IJRR)*, 2016.

[12] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart. Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1872–1878, 2015.

[13] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-Time Camera Tracking and 3D Reconstruction using Signed Distance Functions. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[14] A. Caccavale and M. Schwager. A distributed algorithm for mapping the graphical structure of complex environments with a swarm of robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[15] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice. Generic Node Removal for Factor-

Graph SLAM. *IEEE Transactions on Robotics*, 30(6):1371–1385, 2014.

[16] L. Carlone and K. Sertac. Attention and anticipation in fast visual-inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[17] J. Castellanos, J. Tardós, and G. Schmidt. Buildiong a Global Map of the Environment of a Mobile Robot: The Importance of Correlations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[18] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert. Distributed trajectory estimation with privacy and communication constraints: A two-stage distributed gauss-seidel approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5261–5268. IEEE, 2016.

[19] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, H. I. Christensen, and F. Dellaert. Distributed mapping with privacy and communication constraints: Lightweight algorithms and object-based models. *International Journal of Robotics Research (IJRR)*, 36(12):1286–1311, 2017.

[20] S. Choudhary, L. Carlone, C. Nieto, J. Rogers, Z. Liu, H. I. Christensen, and F. Dellaert. Multi Robot Object-based SLAM. pages 729–741, 2016.

[21] T. Cieslewski, S. Choudhary, and D. Scaramuzza. Data-efficient decentralized visual SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2466–2473, 2018.

[22] T. Cieslewski and D. Scaramuzza. Efficient decentralized visual place recognition using a distributed inverted index. *IEEE Robotics and Automation Letters*, 2(2):640–647, 2017.

[23] J. Civera, A. J. Davison, and M. Montiel. Inverse Depth Parametrization for Monocular SLAM. 2008.

[24] M. Cox. The Numerical Evaluation of B-Splines. *IMA Journal of Applied Mathematics*, 10(2):134–149, 1972.

[25] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research (IJRR)*, 27(6):647–665, 2008.

[26] M. Cummins and P. Newman. Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[27] A. Cunningham, V. Indelman, and F. Dellaert. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[28] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[29] K. Daniilidis. Hand-Eye Calibration Using Dual Quaternions. *International Journal of Robotics Research*, 18:286–298, 1998.

[30] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart. Hybrid Predictive Control for Aerial Robotic Physical Interaction towards Inspection Operations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[31] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):1052–1067, 2007.

[32] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–

62, 1972.

[33] I. Deutsch, M. Liu, and R. Siegwart. A Framework for Multi-Robot Pose Graph SLAM. In *IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 2016.

[34] J. Dong, E. Nelson, V. Indelman, N. Michael, and F. Dellaert. Distributed real-time cooperative localization and mapping using an uncertainty-aware expectation maximization approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[35] E. Eade. Lie Groups for Computer Vision. `http://ethaneade.com/lie_groups.pdf`. Accessed: 2016-02-19.

[36] E. Eade and T. Drummond. Scalable Monocular SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 469–476, 2006.

[37] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-D mapping with an RGB-D camera. *IEEE Transactions on Robotics (T-RO)*, 30(1):177–187, 2014.

[38] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[39] J. Engel, J. Stückler, and D. Cremers. Large-scale direct SLAM with stereo cameras. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.

[40] M. Erdelj, M. Król, and E. Natalizio. Wireless Sensor Networks and Multi-UAV systems for natural disaster management. *Computer Networks*, 124:72 – 86, 2017.

[41] M. Faessler, E. Mueggler, K. Schwalbe, and D. Scaramuzza. A Monocular Pose Estimation System based on Infrared LEDs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[42] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. In *IEEE Transactions on Robotics (T-RO)*, 2017.

[43] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[44] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

[45] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-Time Batch Estimation Using Temporal Basis Functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[46] P. Furgale, J. Rehder, and R. Siegwart. Unified Temporal and Spatial Calibration for Multi-Sensor Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[47] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley. Continuous-time batch trajectory estimation using temporal basis functions. *The International Journal of Robotics Research*, 34(14):1688–1710, 2015.

[48] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham, 2016.

[49] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376, 2010.

[50] D. Gallup, J.-M. Frahm, P. Mordohai, and M. Pollefeys. Variable baseline/resolution stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[51] D. Galvez-López and J. D. Tardos. Bags of Binary Words for Fast Place Recognition in Image Sequences. In *IEEE Transactions on Robotics (T-RO)*, 2012.

[52] P. Gohl, D. Honegger, S. Omari, M. Achtelik, M. Pollefeys, and R. Siegwart. Omnidirectional visual obstacle detection using embedded FPGA. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3938–3943, 2015.

[53] C. X. Guo, K. Sartipi, R. C. DuToit, G. A. Georgiou, R. Li, J. O'Leary, E. D. Nerurkar, J. A. Hesch, and S. I. Roumeliotis. Large-scale cooperative 3D visual-inertial mapping in a Manhattan world. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1071–1078, 2016.

[54] G. Heredia, A. E. Jimenez-Cano, I. Sanchez, D. Llorente, V. Vega, J. Braga, J. A. Acosta, and A. Ollero. Control of a multirotor outdoor aerial manipulator. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 3417–3422, 2014.

[55] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representation through encapsulation of manifolds. *Information Fusion*, 2011.

[56] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory*, 4:303–320, 1969.

[57] T. Hinzmann, C. Cadena, and R. Nieto, Juan andSiegwart. Flexible Trinocular: Non-rigid Multi-Camera-IMU DenseReconstruction for UAV Navigation and Mapping. In *2019 IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS*. IEEE, 2019.

[58] T. Hinzmann, T. Taubner, and R. Siegwart. Flexible Stereo: Constrained, Non-Rigid, Wide-Baseline Stereo Vision for Fixed-Wing Aerial Platforms. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2550–2557, May 2018.

[59] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. In *Journal of the Optical Society of America*, 1987.

[60] P. Huke, R. Klattenhoff, C. von Kopylow, and R. Bergmann. Novel trends in optical non-destructive testing methods. *Journal of the European Optical Society - Rapid publications*, 8(0), 2013.

[61] M. Kamel, K. Alexis, and R. Siegwart. Design and Modeling of Dexterous Aerial Manipulator. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 4870–4876, 2016.

[62] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart. Model Predictive Control for Trajectory Tracking of Unmanned Aerial Vehicles Using Robot Operating System. In A. Koubaa, editor, *Robot Operating System (ROS) The Complete Reference, Volume 2*. Springer, 2017.

[63] M. Karrer, M. Agarwal, M. Kamel, R. Siegwart, and M. Chli. Collaborative 6DoF Relative Pose Estimation for Two UAVs with Overlapping Fields of View. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6687–6693, 2018.

[64] M. Karrer and M. Chli. Towards Globally Consistent Visual-Inertial Collaborative

SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3685–3692, 2018.

[65] M. Karrer, M. Kamel, R. Siegwart, and M. Chli. Real-time Dense Surface Reconstruction for Aerial Manipulation. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1601–1608, 2016.

[66] M. Karrer, P. Schmuck, and M. Chli. CVI-SLAM - Collaborative Visual-Inertial SLAM. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, 2018.

[67] A. Kasyanov, F. Engelmann, J. Sückler, and B. Leibe. Keyframe-based visual-inertial online slam with relocalization. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 6662–6669, 2017.

[68] M.-J. Kim, M.-S. Kim, and S. Y. Shin. A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives. In *ACM Transactions on Graphics (SIGGRAPH)*, pages 369–376, 1995.

[69] S. Kim, S. Choi, and H. J. Kim. Aerial Manipulation Using a Quadrotor with a Two DOF Robotic Arm. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[70] G. Klein and D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[71] L. Kneip, D. Scaramuzza, and R. Siegwart. A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[72] L. Kneip, D. Scaramuzza, and R. Siegwart. OpenGV: A unified and generalized approach to real-time calibrated geometric vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[73] K. Kondak, F. Huber, M. Schwarzbach, M. Laiacker, D. Sommer, M. Bejar, and A. Ollero. Aerial manipulation robot composed of an autonomous helicopter and a 7 degrees of freedom industrial manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[74] H. Kretzschmar, C. Stachniss, and G. Grisetti. Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 865–871, 2011.

[75] F. R. Kschischang, B. Frey, and H.-A. Loeliger. Factor Graphs and the Sum-Product Algorithm. 2001.

[76] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A General Framework for Graph Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[77] D. Lefloch, R. Nair, F. Lenzen, H. Schaefer, L. Streeter, M. J. Cree, R. Koch, and A. Kolb. *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, volume 8200, chapter Technical Foundation and Calibration Methods for Time-of-Flight Cameras, pages 3–24. Springer Berlin Heidelberg, September 2013.

[78] J. J. Lenoard and H. F. Durrant-Whyte. *Directed Sonar Sensing for Mobile Robot Navigation*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[79] J. J. Lenoard and H. F. Durrant-Whyte. Mobile Robot Localization by Tracking Geometric Beacons. In *Proceedings of the IEEE International Conference on Robotics and*

Automation (ICRA), volume 7, pages 376–382, 1992.
[80] J. Leonard and H. Durrant-Whyte. Directed Sonar Sensing for Mobile Robot Navigation. In *Kluwer Academic Publishers*, 1992.
[81] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.
[82] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, and R. Siegwart. Keyframe-based Visual-Inertial SLAM using Nonlinear Optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2013.
[83] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual-inertial SLAM using nonlinear optimization. *International Journal of Robotics Research (IJRR)*, 34(3):314–334, 2015.
[84] L. Li. Time-of-Flight Camera - An Introduction. White Paper, May 2014.
[85] M. Li and A. I. Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.
[86] S. Lovegrove, A. Patron-Perez, and S. Gabe. Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras. 2013.
[87] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.
[88] L. Marconi, R. Naldi, A. Torre, J. Nikolic, C. Huerzeler, G. Caprari, E. Zwicker, B. Siciliano, V. Lippiello, R. Carloni, and S. Stramigioli. Aerial Service Robots: an Overview of the AIRobots Activity. In *International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.
[89] A. Marjovi, J. G. Nunes, L. Marques, and A. de Almeida. Multi-robot exploration and fire searching. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1929–1934, 2009.
[90] J. Maye, P. Furgale, and R. Siegwart. Self-supervised Calibration for Robotic Systems. In *Inteligent Vehicles Symposium (IVS)*, 2013.
[91] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982.
[92] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena. C-blox: A Scalable and Consistent TSDF-based Dense Mapping Approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 995–1002, 2018.
[93] G. Mohanarajah, V. Usenko, M. Singh, R. D'Andrea, and M. Waibel. Cloud-based collaborative 3D mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, 12(2):423–431, 2015.
[94] J. G. Morrison, D. Gálvez-López, and G. Sibley. MOARSLAM: Multiple Operator Augmented RSLAM. In *Distributed Autonomous Robotic Systems*, volume 112, pages 119–132. 2016.
[95] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572. IEEE, 2007.
[96] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. In *IEEE Transactions on Robotics (T-RO)*, 2015.
[97] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An Open-Source SLAM System for

Monocular, Stereo and RGB-D Cameras. *IEEE Transactions on Robotics (T-RO)*, 33(5):1255–1262, 2017.

[98] R. Mur-Artal and J. D. Tardós. Visual-Inertial Monocular SLAM with Map Reuse. In *IEEE Robotics and Automation Letters*, 2017.

[99] R. Newcombe, S. Izardi, O. Hiliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[100] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011.

[101] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. Furgale, and R. Siegwart. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[102] H. Oleynikova, M. Burri, S. Lynen, and R. Siegwart. Real-Time Visual-Inertial Localization for Aerial and Ground Robots. 2015.

[103] H. Oleynikova, D. Honegger, and M. Pollefeys. Reactive avoidance using embedded stereo vision for MAV flight. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 50–56, 2015.

[104] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto. Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, pages 1366–1373, 2017.

[105] H. Oleynikova, Z. Taylor, A. Millane, R. Siegwart, and J. Nieto. A Complete System for Vision-Based Micro-Aerial Vehicle Mapping, Planning, and Flight in Cluttered Environments. 2019.

[106] S. Omari, M. Bloesch, P. Gohl, and R. Siegwart. Dense visual-inertial navigation system for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2634–2640, 2015.

[107] Z. Peng, Y. Xu, M. Yan, and W. Yin. ARock: An Algorithmic Framework for Asynchronous Parallel Coordinate Updates. *SIAM Journal on Scientific Computing*, 38:2851–2879, 2016.

[108] N. Piasco, J. Marzat, and M. Sanfourche. Collaborative localization and formation flying using distributed stereo-vision. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[109] P. Pounds, D. Bersak, and A. Dollar. Grasping From the Air: Hovering Capture and Load Stability. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[110] G. Qian, R. Chellappa, and Q. Zheng. Robust structure from motion estimation using inertial data. *Journal of the Optical Society of America*, 18(12):2982–2997, 2001.

[111] T. Qin, P. Li, and S. Shen. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics (T-RO)*, 34(4):1004–1020, 07 2018.

[112] V. Reijgwart, A. Millane, H. Oleynikova, R. Siegwart, C. Cadena, and J. Nieto. Voxgraph: Globally Consistent, Volumetric Mapping Using Signed Distance Function

Submaps. *IEEE Robotics and Automation Letters*, 5(1):227–234, 2020.
[113] W. Rencken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2192–2197, 1993.
[114] L. Riazuelo, J. Civera, and J. Montiel. C2TAM: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems (RAS)*, 62(4):401–413, 2014.
[115] H. Roth and M. Vona. Moving Volume KinectFusion. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.
[116] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.
[117] B. Sagredo and J. Tercero. Z-splines: moment conserving cardinal spline interpolation of compact support for arbitrarily spaced data. 2003.
[118] I. Schiller, C. Beder, and R. Koch. Calibration of a PMD-Camera Using a Planar Calibration Pattern Together With a Multi-Camera Setup. In *The International Archives of the Photogrametry, Remote Sensing and Spatial Information Sciences*, volume XXXVII, Part B3a, pages 297–302. ISPRS Congress, 2008.
[119] P. Schmuck and M. Chli. Multi-UAV Collaborative Monocular SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
[120] P. Schmuck and M. Chli. CCM-SLAM: Robust and Efficient Centralized Collaborative Monocular SLAM for Robotic Teams. *Journal of Field Robotics (JFR)*, 36(4):763–781, 2019.
[121] P. Schmuck and M. Chli. On the Redundancy Detection in Keyframe-Based SLAM. In *International Conference on 3D Vision (3DV)*, pages 594–603, 2019.
[122] T. Schneider, M. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.
[123] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528, 2006.
[124] K. Shoemake. Animating rotation with quaternion curves. In *ACM Transactions on Graphics (SIGGRAPH)*, pages 245–254, 1985.
[125] Stanford Artificial Intelligence Laboratory et al. Robotic operating system.
[126] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.
[127] F. Steinbruecker, J. Sturm, and D. Cremers. Volumetric 3D Mapping in Real-Time on a CPU. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.
[128] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige. Double Window Optimisation for Constant Time Visual SLAM. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.
[129] H. Strasdat, J. Montiel, and A. J. Davison. Real-time monocular SLAM: Why filter? In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2657–2664. IEEE, 2010.

[130] D. Strelow and S. Singh. Motion estimation from image and inertial measurements. *The International Journal of Robotics Research*, 23(12):1157–1195, 2004.

[131] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[132] C. Sung and P. Y. Kim. 3D terrain reconstruction of construction sites using a stereo camera. *Automation in Construction*, 64:65–77, 2016.

[133] A. Tagliabue, M. Kamel, R. Siegwart, and J. Nieto. Robust collaborative object transportation using multiple mavs. *The International Journal of Robotics Research*, 38(9):1020–1044, 2019.

[134] L. Teixeira, F. Maffra, M. Moos, and M. Chli. VI-RPE: Visual-Inertial Relative Pose Estimation for Aerial Vehicles. *IEEE Robotics and Automation Letters*, 3(4):2770–2777, 10 2018.

[135] L. Teixeira, M. R. Oswald, M. Pollefeys, and M. Chli. Aerial Single-View Depth Completion with Image-Guided Uncertainty Estimation. *IEEE Robotics and Automation Letters*, 5(2):1055–1062, 4 2020.

[136] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers. Visual-Inertial Mapping With Non-Linear Factor Recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, 2020.

[137] S. Vemprala and S. Saripalli. Monocular Vision based Collaborative Localization for Micro Aerial Vehicle Swarms. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 315–323, 2018.

[138] T. A. Vidal-Calleja, C. Berger, J. Solà, and S. Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems (RAS)*, 59(9):654–674, 2011.

[139] L. von Stumberg, V. Usenko, and D. Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517, May 2018.

[140] Z. Wang, S. Singh, M. Pavone, and M. Schwager. Cooperative Object Transport in 3D with Multiple Quadrotors using No Peer Communication. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1064–1071, 2018.

[141] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart. Monocular Vision for Long-term MAV Navigation: A Compendium. *Journal of Field Robotics (JFR)*, 30:803–831, 2013.

[142] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. J. Leonard, and J. McDonald. Kintinuous: Spatially Extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, pages 1–8, 2012.

[143] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research (IJRR)*, 34(4-5):598–626, 2015.

[144] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Proceedings of Robotics: Science and Systems (RSS)*, volume 11, 2015.

[145] M. Zeng, F. Zhao, J. Zheng, and X. Liu. A Memory-efficient Kinectfusion Using Octree. In *Proceedings of the First International Conference on Computational Visual Media*, 2012.

[146] H. Zhang, X. Chen, H. Lu, and J. Xiao. Distributed and collaborative monocular simultaneous localization and mapping for multi-robot systems in large-scale environments. *International Journal of Advanced Robotic Systems*, 15(3):1729881418780178, 2018.

[147] R. Zhang, S. Zhu, T. Shen, L. Zhou, Z. Luo, T. Fang, and L. Quan. Distributed very large scale bundle adjustment by global camera consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 05 2018.

[148] Z. Zhang and D. Scaramuzza. A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2018.

[149] D. Zou and P. Tan. CoSLAM: Collaborative Visual SLAM in Dynamic Environments. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2013.

Curriculum Vitae

Marco Karrer
born October 18, 1988
citizen of Röschenz BL, Switzerland

2016–2020	*ETH Zurich, Switzerland* Doctoral studies at the Vision For Robotics Lab; Supervised by Prof. Margarita Chli
2014–2016	*ETH Zurich, Switzerland* Master of Science in Mechanical Engineering $(5.76/6)$
2012–2014	*Zurich University of Applied Science (ZHAW), Switzerland* Research Assistant
2009–2012	*Zurich University of Applied Science (ZHAW), Switzerland* Bachelor of Science in Mechanical Engineering $(5.6/6)$
2008–2009	*Aluminium Laufen AG, Switzerland* Polymechanic
2004–2008	*Aluminium Laufen AG, Switzerland* Apprenticeship as Polymechanic $(5.8/6)$

List of Publications

- Karrer, M., Schmuck, P., and Chli, M. (2018c). CVI-SLAM - collaborative visual-inertial SLAM. IEEE Robotics and Automation Letters (RA-L), 3(4):2762–2769
- Karrer, M., Agarwal, M., Kamel, M., Siegwart, R.Y. and Chli, M. (2018b). Collaborative 6DoF Relative Pose Estimation for two UAVs with Overlapping Fields of View, IEEE International Conference on Robotics and Automation (ICRA)
- Karrer, M. and Chli, M. (2018a). Towards Globally Consistent Visual-Inertial Collaborative SLAM, IEEE International Conference on Robotics and Automation (ICRA)
- Karrer, M. Kamel, M., Siegwart, R.Y. and Chli, M. (2016). Real-time Dense Surface Reconstruction for Aerial Manipulation, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)