Decentralized Monocular-Inertial Multi-UAV SLAM System

Ariane Spaenlehauer

SUBMITTED IN TOTAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

THESIS COMMITTEE

REVIEWERS

SAMIA BOUCHAFA-BRUNEAU, FULL PROFESSOR, IBISC RESEARCH LABORATORY, UNIVERSITY OF EVRY VAL D'ESSONNE, EVRY, FRANCE

TOM DRUMMOND, FULL PROFESSOR, DEPARTMENT OF ELECTRICAL AND COMPUTER SYSTEMS ENGINEERING, MONASH UNIVERSITY, MELBOURNE, AUSTRALIA

EXAMINERS

VÈRONIQUE CHERFAOUI, FULL PROFESSOR, HEUDIASYC RESEARCH LABORATORY, UNIVERSITY OF TECHNOLOGY OF COMPIÈGNE, COMPIÈGNE, FRANCE

SIMON LACROIX, CNRS SENIOR RESEARCHER, LAAS RESEARCH LABORATORY, TOULOUSE, FRANCE

SUPERVISORS

ISABELLE FANTONI, CNRS SENIOR RESEARCHER, ECOLE CENTRALE DE NANTES, NANTES, FRANCE

VINCENT FRÉMONT, FULL PROFESSOR, ECOLE CENTRALE DE NANTES, NANTES, FRANCE

AHMET ŞEKERCIOĞLU, ADJUNCT FACULTY MEMBER, DEPARTMENT OF ELECTRICAL AND COMPUTER SYSTEMS ENGINEERING, MONASH UNIVERSITY, MELBOURNE, AUSTRALIA

Heudiasyc Research Laboratory
UNIVERSITY OF TECHNOLOGY OF COMPIÈGNE
FRANCE
JULY 2019

Abstract

In this thesis, we provide a scheme for localization of a fleet of autonomous UAVs (unmanned autonomous vehicles) within a Technological System-of-Systems architecture. Specifically, we aim for a fleet of autonomous UAVs to localize themselves and to obtain a map of an unknown environment using a minimal set of sensors on each UAV: A front monocular camera and an Inertial Measurement Unit. This is a critically important problem for applications such as exploration of unknown areas, or search and rescue missions. The choices for designing such a system are supported by an extensive study of the scientific literature on two broad fronts: First, about the multi-robot systems performing localization, mapping, navigation and exploration, and second, about the monocular, real-time and inertial-monocular SLAM (Simultaneous Localization and Mapping) algorithms.

Processing monocular camera frames suffers the drawback of lacking the capability of providing metric estimates as the depth dimension is lost when the frames are photographed by the camera. Although, it is usually not a critical problem for single-robot systems, having accurate metric estimates is required for multi-robot systems. This requirement becomes critical if the system is designed for control, navigation and exploration purposes. In this thesis, we provide a novel approach to make the outputs of monocular SLAM algorithms metric through a loosely-coupled fusion scheme by using the inertial measurements.

This work also explores a design for a fleet of UAVs to localize each robot with minimal requirements: No a priori knowledge about the environment, information about neither the position nor the moment in time the UAV takes off and land is required. Moreover, the system presented in the thesis handles aggressive UAV trajectories having dramatic changes in speed and altitude.

In multi-robot systems, the question of the coordinate frames require more attention than in single robot systems. In many studies, the coordinate frame problem is simplified to the representation of the fleet and the expression of the measurements in a global coordinate frame. However, this kind of hypothesis implies either the use of additional sensors to be able to measure the transformations to the global coordinate frame or additional experimental constraints, for example about the starting position of the robots. Our system does not require absolute measurements like GNSS positioning or knowledge about the coordinate frame of each UAV. As each UAV of the fleet estimates its location and produces a map in its own coordinate frame, relations between those coordinate frames are found by our scheme. For that purpose, we extend the well known concept of loop-closures

in single-robot SLAM approaches, to multi-robot systems. In this research work, we also provide an overview of the new effects due to the extended definition of loop-closures we provide in comparison with the loop-closures scheme that can be found in single robot SLAM algorithms.

In addition to the coordinate frame problem, we provide experimental results about the possibilities for improving the location estimate of a fleet by considering the places visited by several UAVs. By searching for similar places using each UAV imagery, using the 2-D information encapsulated in the images of the same scenery from different view points, and the 3-D map locally estimated by each UAV, we add new constraints to the SLAM problem that is the main scheme that can be used to improve the UAV location estimates. We included experiments to assess the accuracy of the inter-UAV location estimation. The system was tested using datasets with measurements recorded on board UAVs in similar conditions as the ones we target.

DECLARATION

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

The core theme of the thesis is robotics, visual SLAM and inertial-monocular SLAM for UAVs. The ideas, development and writing up of all the work in the thesis were the principal responsibility of myself, the candidate, working within the Heudiasyc research laboratory under the supervision of Dr I. Fantoni, Dr V. Frémont and Dr Y. Ahmet Şekercioğlu.

Signed: _____
Ariane Spaenlehauer

Date: May 2019

ACKNOWLEDGEMENTS

My first thanks go to my supervisors Prof. Vincent Frémont, Prof. Isabelle Fantoni, Prof. Ahmet Şekercioğlu.

I thank Prof. Frémont and Prof. Fantoni for having created the topic I have been working on and for the efforts they devoted to this research work along with their supervision.

I would like to thank Prof. Şekercioğlu, whom I have been working with for more than three years and who has supported and helped me to improve myself beyond any expectations.

A very special thank to Monash University, where I was warmly welcomed and supported during three months. Particularly, I thank Dr. Lui and E. Simic for their support, help and the fruitful conversations we had.

For the time and feedback they dedicated to my thesis, I would like to thank all the members of this thesis reviewing committee.

This research work would not have been possible without the financial and logistic support of the LABEX MS2T center. I am grateful to the LABEX for giving me the opportunity to start and achieve my research work. Particularly, I deeply thank Prof. Ali Charara, Prof. Philippe Bonnifait and Prof. Pierre Morizet for paving the way to Ph.D. for the former engineering student I was at that time.

For the time and attention they gave me, I would like to thank Prof. Véronique Cherfaoui, Prof. Yves Grandvalet and Prof. Antoine Jouglet. They all contributed, in one way or another, to the achievement of this thesis.

As a member of the Heudiasyc Research Laboratory, I would like to thank all the researchers and students who spent time listening and giving me feedback about more or less complex questions. I also thank them for bringing a friendly atmosphere. In particular, I would like to mention my office mates Johanna, Xu Hong and Rim, the time we spent in our office was great and our discussions about inter-culturality were so enriching.

And how not to mention the administrative team of the Labex and the Heudiasyc Laboratory, who are, so far, the most effective and efficient team I have known, many thanks for your support and cheers.

Finally, I address a warm thank to my family, in particular my brother and mother, and to my friends from all horizons.

Contents

List of Figures

List of Tables

ACRONYMS

CABAC	Context-Adaptive Binary Arithmetic Coding
DHC-M	Differential Huffman Coding with Multiple Lookup Tables
DLT	Direct Linear Transformation
DOF	Degree of Freedom
FAST	Features from Accelerated Segment Test
FoV	Field of View
FPFH	Fast Point Feature Histograms
GoP	Group of Pictures
GPS	Global Positioning System
GPU	Graphics Processor Unit
ICP	Iterative Closest Point
IPM	Initial Pose Matrix
IW-DVC	Image Warping Based Depth Video Compression
MSE	Mean Squared Error
MV	Motion Vector
MVC	Multiple View Coding
ORB	Oriented FAST and Rotated BRIEF
PGF	Progressive Graphics File
PSNR	Peak Signal-to-Noise Ratio
RANSAC	Random Sample Consensus
RGB-D	Red,Green,Blue-Depth
RPRR	Relative Pose based Redundancy Removal
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SSIM	Structural Similarity
SURF	Speeded Up Robust Features
UM	Uncertainty Matrix
VSN	Visual Sensor Network
WSN	Wireless Sensor Network

LIST OF VARIABLES

$\{A\}$	The coordinate frame $\{A\}$ referred as A in equations.
$\mathbf{M}^T$	The transpose of the matrix $\mathbf{M}$.
$\mathbf{M}^{-1}$	The inverse of the matrix $\mathbf{M}$.
${}^A\mathbf{R}_B$	3-by-3 rotation matrix that rotates vectors from $\{B\}$ to $\{A\}$.
$\mathbf{t}$	3-by-1 translation vector.
$\mathbf{t}^i$	Translation vector calculated from inertial measurements.
$\mathbf{t}^m$	Translation vector calculated from monocular vision measurements.
$\mathbf{t}^g$	Translation vector calculated from ground truth measurements.
${}^W\mathbf{t}_{F_i,F_j}$	Translation vector between the frames F_i and F_j written in the coordinate frame $\{W\}$.
F_i	Camera coordinate frame associated with the image frame i.
${}^A\mathbf{T}_B$	4-by-4 transformation matrix that transforms $\{B\}$ into $\{A\}$.
${}^A\mathbf{T}_{B_p}$	Transformation matrix that transforms $\{B\}$ at time p into $\{A\}$, implies that ${}^A\mathbf{T}_B$ is changing along time with respect to $\{A\}$.
λ	Scaling coefficient.
$\mathbf{b}_a$, $\mathbf{b}_\omega$	IMU biases for the accelerometers and gyroscopes.
$\mathbf{g}$	The gravity vector.
$\mathbf{a}(p)$	3-by-1 vector which contains the accelerometer measurements at time p (one vector component per axis).
$\boldsymbol{\omega}(p)$	3-by-1 vector which contains the gyroscope measurements at time p (one vector component per axis).
ΔT	Time step in seconds.
${}^A\mathbf{p}$	3-by-1 vector of a 3-D measurement in $\{A\}$, usually a 3-D point from the map or the environment.
${}^i\mathbf{p}$	A 3-D point in the inertial coordinate frame.
${}^c\mathbf{p}$	A 3-D point in the camera coordinate frame.
${}^{\mathrm{UAV}_{a_i}}\mathbf{R}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$	The 3-D rotation between the camera coordinate frames of UAVs a and b at respective time, or keyframes, i and j.
${}^{\mathrm{UAV}_{a_i}}\mathbf{t}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$	The 3-D translation between the camera coordinate frames of UAVs a and b at respective time, or keyframes, i and j.
${}^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$	The homogeneous transformation between the camera coordinate frames of UAVs a and b at respective time, or keyframes, i and j.

${}^{j}\mathbf{T}^{\mathrm{i}}_{k}$	The relative transformation between the j^{th} and the k^{th} pose of the inertial coordinate frame.
${}^{\mathrm{UAV}_a}\mathbf{T}^{\mathrm{i}}_{j}$	The j^{th} pose of the inertial coordinate frame placed on the UAV_a with respect to the coordinate frame $\{\mathrm{UAV}_a\}$. We consider the poses of the UAV as homogeneous transformations between a fixed coordinate frame placed on the UAV body and a coordinate frame fixed with regard to the world coordinate frame.
${}^{\mathrm{UAV}_a}\mathbf{T}^{\mathrm{c}}_{j}$	The j^{th} pose of the camera coordinate frame placed on the UAV_a with respect to the coordinate frame $\{\mathrm{UAV}_a\}$. We consider the poses of the UAV as homogeneous transformations between a fixed coordinate frame placed on the UAV body and a coordinate frame fixed with regard to the world coordinate frame.
${}^{\mathrm{c}}\mathbf{T}_{\mathrm{i}}$	The constant transformation between the inertial coordinate frame and the camera coordinate frame that are placed on the UAVs (extrinsic calibration).
${}^{\mathrm{UAV}_a}\mathbf{T}_{\mathrm{UAV}_b}$	The transformation between the coordinate frame of $\{\mathrm{UAV}_a\}$ and the coordinate frame of $\{\mathrm{UAV}_b\}$.
${}^{j}\mathbf{T}^{\mathrm{i}}_{k}$	The transformation between the inertial frames associated with the j^{th} keyframe (or pose) of the UAV_a and the k^{th} keyframe (or pose) of the UAV_b.
$\mathbf{P}$	A covariance matrix.

CHAPTER ONE

INTRODUCTION

1.1 Context

1.1.1 Mobile robotics

Robotics is taking an important and central role in human life. Numerous robotic applications and concepts are emerging to improve our society and our ways of life. Within this increasing number of possibilities, mobile robotics is a great research field because of the obvious economic and societal impacts that have and the challenging complexity of the related research problems [18c].

Mobile robotics is a large research fields that include numerous platforms including ground, aerial, marine and submarine robots. Each platform has its own specificities that includes simultaneously overlapping research problems such as localization and communication, and specific constraints, for example, small quadrotor platforms, in comparison to large heavy ground robot, have limited computational power and payload, and a particular dynamic in the trajectories.

1.1.2 Unmanned Aerial Vehicles (UAVs)

Lately, improved availability of robots has drastically increased the enthusiasm for UAVs (Unmanned Aerial Vehicles) mostly for their capability to fly, and therefore, the fewer limitations they have to move in a lot of different environments. UAVs are an interesting platform as they benefit from their small size, high maneuverability

and ability to fly in challenging environments [Fra+12] for being able to cope with a wide variety of missions. Nowadays, numerous UAVs can be found on the market and a trade-off must be made between the price and capabilities. This trade-off is a critical question since it directly constraints the choice of sensors which dramatically affect the design of the embedded algorithms. In other words, a small, low-cost UAV offers a very limited choice for sensors, and the measurements are likely to be noisier than in expensive systems; specifically for quadrotors whose small payload capacity and limited battery life prevent the use of heavy and powerful sensors. Nevertheless, small quadrotors are exciting platforms to work with because the sensor constraints require more robust and general algorithms.

A significant number of research fields are directly linked with the UAV platforms. A non-exhaustive list of some of those research problems is:

- Study of the computational requirements for the embedded algorithms in order to be suitable with the platform and running either in real-time or fast enough,
- Low level control for the UAV stabilization in various environments including windy environments,
- Design of new platforms to allow additional payload and sensors,
- Conception of specific algorithms to match with the sensors placed on board the UAVs,
- Robustness of the algorithms with regards to the platform, such as controlling UAVs with one or several broken rotors or making robust the tracking and localization schemes to aggressive maneuvers.

This thesis work aimed to design localization algorithms and approaches suitable for small and low-cost UAVs. A particular focus was dedicated to the available sensors in order to make the algorithms suitable for most of the currently available UAVs. We motivated the choice of the platform, which implies several constraints,

particularly regarding the sensors, by grouping the UAVs in a fleet to overcome some of the individual drawbacks. Section 1.2 provides the description of the research work, including its structure and work hypothesis. Section 1.3 gives an overview of the encapsulating project and explain some of the design choices that was made in this work in order to ensure the continuity with regard to the DIVINA challenge team, a project that include the work of several researchers and research axis.

1.2 The M-SLAM research work: A decentralized multi-UAV SLAM system

1.2.1 Objective

In this research work, we aimed to design a system that is able to provide the localization of a fleet of autonomous UAVs for exploration of unknown areas. For the readability, this work is named M-SLAM, an acronym for Multi-UAV SLAM. Several scientific underlying sub-problems have been identified and solved throughout the M-SLAM research work, for example, Chapter 3 explains the design of the experimentation schemes that is required for being able to perform experimentations in similar environments as the ones we target, and Chapter 4 discuss the problem of metric estimates that we needed as an input for the M-SLAM system.

1.2.2 Scientific challenges and hypothesis

In Chapters 5, 6 and 7, we discuss the M-SLAM framework as a Technological System-of-Systems: A decentralized inertial-monocular multi-UAV SLAM system. The goal of this system is to provide for each UAV of the fleet:

1. The localization of the UAV under consideration,

2. The localization of the other UAVs of the fleet with respect to the UAV coordinate frame under consideration,

3. Improving the localization estimates of the UAV under consideration using localization and map information of the other UAVs.

In this system, we use the minimal set of sensors available in every modern low-cost quadrotor: A single front monocular camera and an Inertial Measurement Unit composed of three gyroscopes and three accelerometers. Some of the scientific challenges lie in the fact that the M-SLAM system does not use any absolute or external measurements such as GPS or motion capture systems, nor inter-robot measurements (that could be provided by a recognition system, using visual tags or laser measurements), nor a-priori knowledge about the location, the map or the coordinate frames of any UAVs. When the system starts, there is no knowledge about the localization of the UAVs of the fleet or the moment they take off or land. In addition, with regard to the applications we target, the system must also be able to

- Process measurements generated on-board by the front monocular camera and the IMU,
- Deal with static 3-D environments for both the map and the localization,
- Be robust enough to withstand aggressive trajectories typical of small UAVs as well as significant changes in altitude and speed of the UAVs,
- Cope with noisy measurements (because of the low-cost IMU and possibly blurred images from the camera),
- Provide metric estimates for further exploration, navigation and control strategies.

We consider the M-SLAM system as a decentralized system based on the following definition. We define as a decentralized system, a system where the decisions

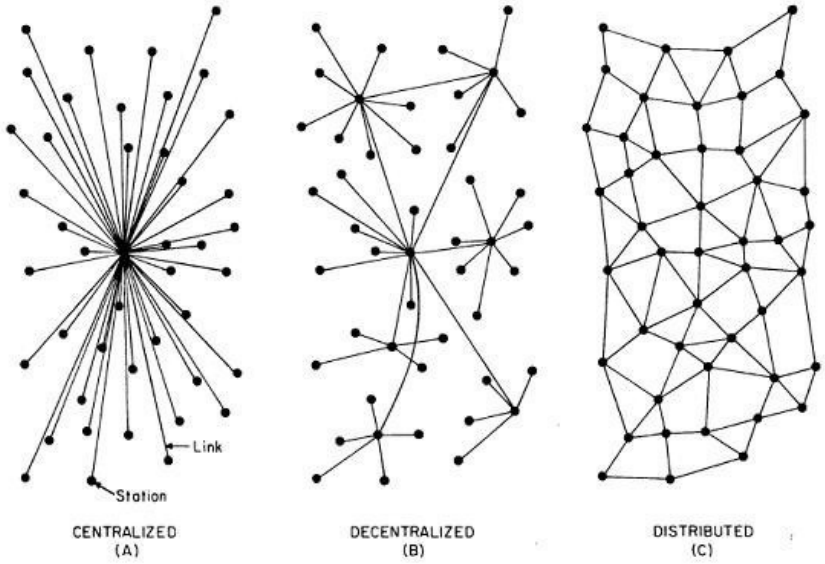


Figure 1.1: Visual representation of the differences between centralized, decentralized and distributed systems.

are not made in a single point; in mobile robotics, a point is often seen as one of the robot of the fleet. In a decentralized system, each robot makes its own decision, the overall behavior of the system is obtained by the aggregation of each robot response. In opposition to decentralized systems, in centralized systems, one of the robot (or a ground station) make the decisions for each robot of the fleet. Finally, in distributed systems, the processing is done across several robots. Distributed systems can be used to lower the computational load for one robot and can be particularly interesting for heavy processing tasks such as solving an optimization problem. Figure 1.1 provides a visual representation of a centralized, a decentralized and a distributed system. A particular attention is required to the definitions of centralized, decentralized and distributed systems as each scientific community provide slightly different definitions and categorizations. The definition we chose fits with the one given generally by the computer science community.

The decentralized aspect of the M-SLAM system lies in the fact that each UAV measures its location and map with regards to its own coordinate frame and estimate the location of the other UAVs of the fleet also with regard to its own coordinate frame. The M-SLAM system is robust to the absence of communication (direct or indirect) between two (or more) UAVs of fleet, in this case, the information

is simply not considered and each UAV use only the information available from the other communicating UAVs. This approach allow each UAV to have enough information to compute target points for autonomous exploration by itself, without the need of a central brain that would send exploration command to each UAV.

1.2.3 The M-SLAM work structure

The structure of the M-SLAM work was split into the following four stages:

1. Identify the most suitable monocular SLAM algorithm to be run on each UAV,
2. Design an inertial-monocular fusion scheme,
3. Find satisfactory datasets to perform the experiments and assess the system, and
4. Design and implement the M-SLAM algorithm.

When this thesis begun, there were few monocular SLAM algorithms capable of being run on-board UAVs [MMT15], and none of them were capable of performing a tightly-coupled inertial-monocular fusion. Monocular vision has the inherent drawback of not being metric [HZ03]: The leading monocular SLAM algorithms provide an internally consistent map and localization but the unit of distance is arbitrary [MMT15] [ESC14]. As metric estimates is one of the requirements of this research work, we designed a fusion scheme for inertial and monocular measurements. The first part of the M-SLAM work was, therefore, devoted to the identification of a suitable monocular SLAM algorithm robust enough to the UAV's fast and aggressive trajectories. The second part of the thesis work was spent creating a new loosely-coupled fusion algorithm to make the SLAM estimates metric. In order to assess this fusion scheme, we reviewed the published benchmarks and datasets in order to find one that would be a faithful representation of the type of environments described in the requirements. Furthermore, we were already searching for datasets that could be used to emulate a fleet of UAVs. After we

designed the loosely-coupled fusion scheme for inertial measurements and monocular camera frames, the first tightly-coupled inertial-monocular SLAM algorithm with an open-source programming code was released [QLS17], meaning that we could use either the loosely-coupled scheme we created or the tightly-coupled SLAM as an input to the M-SLAM system. The fourth, and last, part was devoted to the design of the M-SLAM algorithm and then, its implementation.

The M-SLAM research work is encapsulated in a bigger project that is described in Section 1.3.2. The presentation of the encapsulating project is important as it defines constraints for the work hypothesis, inputs and outputs required by the M-SLAM system.

1.3 Fleet of UAVs for large scale exploration

1.3.1 Underlying research problems

Fleet of UAVs are very promising platforms for exploration of unknown or partially known areas. The high maneuverability, the capacity to fly in almost any environment and the possible low prices of UAVs make it a suitable platform to explore many places. The applications of such a system are wide: Search and rescue missions, bringing critical information for disaster management, or exploration of dangerous places. Lately, international scientific teams have been working to design autonomous fleet of UAVs, however, due to the numerous complex underlying research problems to solve, no one has succeeded in creating a fully satisfactory solution yet.

The underlying problems can be classified in three main research axis being control (dealing with the robot movements), communication (dealing with the data exchanged between the robots) and perception (dealing with the processing of measurements from the sensors). A finer classification would be to separate the environment representation (also known as mapping), the localization (where each robot is), the communication (how the robot exchange data, how to maintain

communication in between the robots when required and strategies to deal with loss of data), the navigation or path-planning (how each robot travels from a location A to a location B) and exploration (how one, or several robots of the fleet, define collaboratively and autonomously the next area to be visited by which robots of the fleet). The DIVINA challenge team presented in Section 1.3.2, that encapsulate the M-SLAM work, aims to create solutions to these research problems.

1.3.2 Overview of the DIVINA challenge team

The study presented in this thesis report is part of a bigger project at the Heudiasyc laboratory called the DIVINA challenge team [18a] and funded by the Labex MS2T (Laboratory of Excellence on Technological System-of-Systems) center [18b]. The goal of the DIVINA challenge team is to design a system that is capable of efficient exploration of large unknown areas using a heterogeneous fleet of autonomous UAVs relying mainly on vision-based sensing capabilities. The Technological System-of-Systems (TSoS) theme is inherent to the fleet of autonomous UAVs. In fact, a fleet of UAVs possesses all the TSoS properties:

- It can be scalable (removing or adding a drone does not modify the fleet behavior),
- Each subsystem (each drone) can be independent from the others,
- If the fleet is considered as a whole, it enables new, expanded missions that are not achievable using a single UAV systems.

The new possibilities offered by the fleet can be seen as the emergent behavior of the TSoS. The applications of the DIVINA research work are wide, some examples are search and rescue missions, surveillance, and mapping of challenging environments.

1.3.3 Structure of the encapsulating project

The DIVINA challenge team space possesses several research sub-problems that must be solved. An overview of the state-of-the-art regarding each of those sub-problems is provided in Section 2.2. The project structure that has been chosen to achieve the DIVINA challenge can be seen through the three main overlapping following sub-projects:

- Control and communication: Cooperative packet-loss management and robust network using fleets of UAVs,
- Perception and communication: Robust multi-robot visual SLAM with a heterogeneous mobile camera network, and
- Perception and control: Efficient exploration using active-SLAM strategies.

In this thesis, we present the work on the M-SLAM system described in Section 1.2 that is included in the perception and control sub-project. The M-SLAM system explores the possibilities for improving the accuracy of the localization of each UAV by using the information gathered from the other members of the fleet. It also provides a decentralized system to enable the every member of the fleet to know the location of the other UAVs. Those mechanisms are critical in order to design an effective control part that will handle the macro-management of the fleet in order to have an efficient exploration as well as the design of satisfactory active-SLAM algorithms.

1.4 Contributions

In order to achieve the scientific challenges underlined in Section 1.2.2, we:

- Created a new loosely-coupled monocular-inertial fusion scheme for metric SLAM estimates published in [Spa+17],

- Designed a decentralized system such that each UAV can estimate the location (3-D poses) of the rest of the fleet without any initial a-priori knowledge,
- Demonstrate the effect on each UAV's location by using the information provided by the other UAVs of the fleet, and
- Generalized the loop-closure mechanism [SK08] of a single-robot system to a multi-robot system.

1.5 Overview of the thesis report

This chapter introduced the scientific challenges and contributions of the M-SLAM work, and provided an overview of the encapsulating project DIVINA as well as the context and the possible applications. Chapter 2 is devoted to the literature review related to the two main topics explored through the thesis work: mobile robot Technological System-of-Systems for exploration of unknown areas that is the main application targeted by the M-SLAM system outputs, and single robot monocular and monocular-inertial SLAM algorithms that are the starting point of the M-SLAM work. In Chapter 3, the challenges related to the experimental part are discussed, an extensive review of the available datasets and benchmarks is provided and the coordinate frame problem is described. In Chapter 4, the description of our loosely-coupled fusion scheme for metric estimates in inertial-monocular SLAM is presented. Chapter 5 is devoted to the front-end of the M-SLAM system, the inputs and their processing is described. Chapter 6 details the back-end of the M-SLAM system, including the management of loop-closures in multi-UAV systems and the correction of the estimates using the pieces of information brought by the fleet. Chapter 7 is devoted to the M-SLAM experimental results. Finally, Chapter 8 concludes the work done throughout the M-SLAM research work and provides an overview of possible improvements and further research.

CHAPTER TWO

FLEETS OF AUTONOMOUS MOBILE ROBOTS AND SINGLE ROBOT SIMULTANEOUS LOCALIZATION AND MAPPING: A SURVEY

2.1 Structure of the literature review

In this literature review chapter we aim to achieve two main objectives:

1. Provide an extensive literature review about mobile robotics for autonomous fleets, and

2. Single-robot SLAM with a specific focus on monocular and inertial-monocular measurements.

The M-SLAM research work (Section 1.2) and its applications, as we defined it, include concepts inherent to Technological System-of-Systems (TSoS) and is based on the studies done in the field of autonomous fleets in mobile robotics. As we use single-robot SLAM as an input to our M-SLAM work, an extensive review of the SLAM algorithms was required.

2.2 An overview of mobile multi-robot systems

2.2.1 Scientific challenges of mobile robot systems

In mobile robotics, Technological Systems-of-Systems (TSoS) have a great potential to fulfill more complex missions than single robot systems can achieve, they can also improve the possibilities of single robot systems. Search and rescue [Kas+16], surveillance [Sas+16], mapping [Guo+16] or weight-carrying [Mel+13] tasks are typical cases for which TSoS can outperform single robot solutions. However, to work as a TSoS, the following closely linked problems need to be tackled:

1. First, whatever the goal is, the robot localization needs to be addressed;

2. Second, for most missions (or to solve the localization part), a mapping process is often required. For example, exploration and navigation are the tasks for which a representation of the environment is needed. In order to address both the localization and mapping, Simultaneous Localization And Mapping (SLAM) algorithms were invented. SLAM algorithms provide a map and the robot localization [SK08];

3. Third, the control of the fleet requires an appropriate design. Three levels of control can be distinguished: The low-level control is used to stabilize every mobile robot, an intermediate level manages collision avoidance and local path planning, and the macro level deals with the global organization of the fleet to fulfill the mission in the most efficient way;

4. Finally, network constraints have to be considered. TSoS is based on collaborative work, therefore, communication between the agents is necessary. The network constraints affect two parts: The path planning to insure the network topology and the load of the links, i.e., the quantity of exchanged data, which directly affect the representation of the localization and map estimates.

2.2.2 Research topics related to fleet of autonomous mobile robots

Mobile fleet of autonomous robots related research topics can be categorized in broad terms as follows:

1. Localization of robots;
2. If required, mapping of the environment to fulfill the mission;
3. Robot stabilization, collision-free navigation and inter-robot coordination; and
4. Networking infrastructure for inter-robot communications.

To our knowledge currently, there is no system that solves all these aspects in a satisfactory scheme. Most of the solutions deal with only one or some of the problems mentioned above. In the following sections, we provide an overview of the current approaches to design collaborative mobile multi-robot systems. Table 2.1 summarizes the problems addressed in this literature review chapter.

In the following sections, we present the literature offering multi-robot solutions to each of the following problems: Localization, mapping, path planning and navigation, and exploration. We also provide an insight to advanced methods that combined one or several problems such as active SLAM approaches or methods combining localization, mapping and path planning components. This literature review focuses mainly on the approaches, therefore it includes experiments done using simulations, ground robots or UAVs, with different set of sensors such as lidars, monocular or stereo cameras, IMUs, or RGB-D sensors. We considered both centralized and decentralized methods in order to have an overall vision about the work done in the robotic community.

2.2.3 Localization and mapping in mobile multi-robot systems

The localization of robots is the first problem to address. The approach can be decentralized through relative estimation with respect to the other robots coor-

Publication	Localization	Mapping	SLAM	Path planning, navigation	Exploration	Network
[Loi+16]	x	x		x		
[Ach+12]			x	x		
[Liu+16]		x		x	x	
[Sas+16]	x			x		
[Qur+16]		x				x
[Nes+16]					x	x
[LTK15]			x			
[For+13]			x			
[RB02]	x					
[Kas+16]	x					
[Cog+12]	x					
[Ram+16]		x				
[Guo+16]		x				
[Sch+15]			x			
[CMM15]			x			
[Alo+15]				x		
[Fra+07]					x	
[Fra+09]					x	
[Sch+18]	x	x				
[ZT13]			x			
[SC18]			x			x

Table 2.1: Literature reference per problem solved as a TSoS.

dinate frames or centralized into a global reference coordinate frame. A Kalman Filter based solution using each robot motion is offered in [RB02]. The poses are updated using robot detection and relative localization. This method can be decentralized by transforming the Kalman Filter into small communicating filters. The experiments were done using ground robots. In [Cog+12], a decentralized 3D quadrotor localization solution is proposed. It uses inertial and visual information to reconstruct anonymous bearing measurements (the robot ID is not required). The relative robot poses and velocities are estimated through a probabilistic multiple registration algorithm (for the relative pose) and a particle filter (for the relative distance recovery).

In visual systems, localization and mapping are often paired. In [Kas+16], the localization task is supported by the mapping process. A drone creates a dense map

using the REMODE algorithm [PFS14], which performs a monocular dense depth estimation for a set of keyframes. The ground robot fleet is localized with respect to elevation maps built based on the dense aerial environment representation provided by the drone. In [Guo+16], the trajectories are estimated only to fulfill the mapping goal. The authors offer a Batch Least Square solution to build a collaborative 3D map using visual-inertial data obtained from multiple overlapping points of view taken from unknown poses.

Recently, in 2018, an interesting framework for map merging in multi-robot systems has been published [Sch+18]. The framework Maplab offers a visual-inertial odometry scheme named ROVIOLI and a modular back-end for experimentation in the field of map merging. Maplab targets to be a useful tool for research in multi-robot systems by making possible to exchange the front-end or back-end algorithms for prototyping new approaches. It is a promising tool for the design of new algorithms for multi-robot localization, mapping and SLAM systems.

2.2.4 SLAM in mobile multi-robot systems

Another way to tackle both localization and mapping problems is to use SLAM algorithms (Simultaneous Localization And Mapping) like in the multi-robot SLAM systems proposed in [LTK15], [For+13], [Sch+15], [CMM15]. Every robot of the swarm[1] runs a SLAM algorithm to know its localization into a local map. Then, all the local maps are merged into a global map, and the fleet is localized with respect to this global frame. Though the method can theoretically be decentralized, the map merging process usually demands a lot of computational power and the transmission of the local maps between the agents can quickly overload the network. For those reasons, research teams often use a ground station to manage the map merging [CMM15], [LTK15] (and also the mapping in [For+13]). In [Sch+15], there is no need for a ground station because the experiments were done on ground

[1]Swarm robotic systems are entities built of several individual robots and hold three properties (Robustness, Flexibility, and Scalability), which are ensured through decentralization, redundancy and local interaction.

robots which offer more computational power than UAVs. Note that the choice of sensors is of great importance regarding the SLAM algorithm. For example, in [LTK15] the monocular multi-map visual odometry algorithm PTAMM [CKM08] is fused with depth measurements from a RGB-D sensor to provide a metric visual SLAM algorithm which outputs a dense map reconstruction and metric robot pose estimates; in [For+13], the authors solve the localization problem using a monocular visual odometry approach [KCS11] and build the map using a collaborative structure from motion system on a ground station; in [Sch+15], a fusion between the data of a stereo camera, inertial measurements from an IMU and marker-based robot detections is performed into local reference filters; the approach presented in [CMM15] relies solely on monocular camera.

One of the most significant and oldest collaborative visual SLAM is the CoSLAM algorithm [ZT13]. CoSLAM provides a solution to deal with dynamic environments using a fleet of cameras only. The poses of the camera are estimated relatively to the other cameras using mainly static points in the environments, the dynamic points are also considered to refine the estimates. The algorithm, that is a feature-based approach, includes an important stage of point classification (dynamic versus static points) and an approach to merge the overlapping local maps created by the cameras. The optimization is done running a Bundle Adjustment algorithm regularly.

Using some of the concepts brought by CoSLAM, the CCM-SLAM algorithm is a centralized collaborative SLAM [SC18] that has been published in 2018. The focus is set on the merging of the local maps of a fleet of UAVs (three UAVs in the experimental part) on server and on the robustness of the approach to networking failure and bandwidth of the communication infrastructure. The UAVs are equipped with a front monocular camera, a communication device and a small processing board. The algorithm is designed such that each UAV can keep its autonomy with regard to the fleet which makes the system scalable. Each UAV performs a visual odometry similar to the front-end of the ORB-SLAM algorithm

and reconstruct a local map. Those data are sent to the server that merge the map and runs a full Bundle Adjustment for the optimization part. The computation is done using relative coordinates in order not to create additional drift if new data is received by the server during an optimization stage. There is no global fixed coordinate frame, however when the fusion takes place, the transformation between the local coordinate frames of each UAV involved in the process must be known. For doing so, a Sim(3) transformation is estimated using the UAV keyframes.

Localization, mapping, and thus SLAM algorithms are usually parts of the perception process. They benefit from the sensor measurements to estimate the agent state, i.e., its localization and a representation of the surrounding environment. To design an autonomous mobile fleet of robots, the localization of the agents is a requirement but is not sufficient to make the system autonomous. An autonomous system must be able to make decisions and take actions using its knowledge about itself and the environment. The perception component brings the knowledge the agent need then to take actions, and therefore to become autonomous. For example, a typical kind of decision an autonomous mobile robot must be able to take, is where to move, and for doing so, first, the agent needs to know where it is. Particularly in mobile robotics, the path planning and the autonomous navigation tasks require the design of suitable algorithms.

2.2.5 Navigation and path planning in mobile multi-robot systems

In [Ach+12], the research team presents a European project which goal is to create a multi UAV vision-based system for collaborative localization, mapping of unknown areas and optimal positioning for surveillance. It is a typical example of research work that uses embedded sensors only for localization, mapping and path planning for the autonomous navigation of the fleets of UAVs. The deployment of a UAV swarm for surveillance purposes is also tackled in [Sas+16]. The UAV localization

is relative and is based on vision processing through visual black and white tags. This solution offers to find a robot distribution that covers all the required areas and plan feasible trajectories to organize the UAVs accordingly. The trajectories may not be optimal, however they consider the UAVs motion constraints, the environment constraints and prevent any drone collisions. Another approach in local path planning for collision avoidance in autonomous swarm of UAVs is proposed in [Alo+15]. The team distinguishes the set of collision-free trajectories from the other feasible trajectories. Thus, the proposed system ensures motion continuity and collision free trajectories for multi agent[2] swarms. [Ram+16] addresses the problem of robot navigation with another method. The focus is on topological map construction, which is a sparse representation of the environment. The benefits of topological maps are their usability for fast navigation purposes. The solution relies on the construction of Voronoi graphs which partition the area into discretized regions. 2D bearing measurements and robot detection and identification are used to build the map. However, all the processing is centralized on a server and the method is not suitable for flying agents as measurements are in 2-D.

2.2.6 Exploration in mobile multi-robot systems

A further step to improve path planning is to increase the efficiency of the system. In addition to collision-free and feasible trajectories, we want a fast and effective exploration for example in term of time, energy or cost. In [Liu+16], a system is proposed for collaborative exploration and mapping with a fleet of quadrotors using IMU and laser scanner measurements. [Fra+07] is a multi-robot extension of [Fra+09] and offers a cooperative exploration algorithm based on incremental generation of sensor-based random trees. The trees are data structures which contain the road map of a previously explored area. Along the road map, local safe regions are detected and recorded, they represent local free space. The exploration phase is built according to the local frontiers of the local free spaces. If several

[2]Agent: An autonomous robot.

robots are engaged in the same area, a local coordination process plans feasible and collision-free trajectories for every agent. When robots have finished their exploration phase, they support the rest of the swarm. The experiments were done using ground robots equipped with laser range finders. This approach is limited to 2-D cases, but an important benefit is that it is completely decentralized and therefore, suitable for large robot swarms.

In [Nes+16], exploration and network process are simultaneously tackled through a solution to maintain a connected network topology while exploring a cluttered area with a robot swarm. The exploration task is managed with target points which are provided by a target generator, a black box which contains the implementation of one of the several existing solutions for effective exploration. The algorithm distributes dynamic roles to the robots of the fleet: connectors help to create the network topology, anchors stay in the neighborhood of their target point, a unique prime traveler is elected to lead the exploration and the rest of team are secondary travelers, they help the leader to achieve its mission and participate in the exploration task.

Navigation, and therefore, exploration, require communication between the robots to avoid collisions, and to plan an effective coverage of the area to explore. As we discussed previously, even the perception part can require communication in some systems in order to benefit from the measurements taken by the fleet instead of processing only the measurements taken by a single robot. Maintaining the communication between the robots is a complex research problem by itself.

2.2.7 Network in mobile multi-robot systems

The role of network should not be underestimated in TSoS because the agents have to communicate data somehow. The main problems to solve are threefold: how not to overload the network (for example, in a multi robot mapping process, a clever data representation of the map has to be found), how to build and maintain the network topology, and how to handle loss of connection cases. The topology

construction is addressed in [Ste+16] which proposes an architecture to maintain an ad-hoc network through mobile robot self-organization. The system is composed of two main loops. On the outer loop, a centralized planner provides way points which represent long term feasible trajectories to each robot. The priority is set to the feasibility of the trajectory rather than on the network links. On the inner loop, the local controller guides the robot toward the next way point while ensuring collision avoidance and network routing. In [Qur+16], the loss of connection issue is discussed. The authors describe a collaborative mapping architecture robust to agent departure due to network or sensor failures. The approach is decentralized and builds a collaborative global map. Each active robot of the fleet participates in the global map improvement. The algorithm is robust enough to handle lost of connectivity or sensor failures, a consensus is always guaranteed between the active robots. The framework can be integrated into Google Maps API [19a].

In the previous sections, we focused on approaches dealing either on the perception, the control or the network parts. However, a few published algorithms provide a solution to two of those parts simultaneously, and are therefore particularly interesting for the applications we target in this thesis work. Those algorithms are discussed in the following section.

2.2.8 Advanced algorithms in mobile multi-robot systems

Advanced algorithms for autonomous exploration

In the following paragraphs, we present the recent work about advanced algorithms for autonomous exploration by a fleet of heterogeneous robot, that is the main objective of the DIVINA challenge team. Most of the work hypothesis made in the M-SLAM research work are constrained by the applications targeted, therefore, those algorithms provide important directions towards which the M-SLAM system must be able to fit. First, we present the concept of active SLAM, a design of great interest to combine control, navigation and localization within a SLAM framework. Second, we present some of the most advanced algorithms to solve the challenge of

autonomous navigation using only an IMU and camera placed on board the UAVs. Finally, we open this survey with a study on the use of the fleet heterogeneity.

Active SLAM

In autonomous mobile robotics, research teams usually start to design the perception components as it is always required. Once a satisfactory solution for the perception part has been found, the autonomous aspects of the system are designed. As discussed previously, the next stage after solving the localization and mapping problems, is the design of a component for autonomous navigation. Similarly, SLAM algorithms open another road for navigation: Active SLAM algorithms.

Active SLAM algorithms gather methods which provide path planning based on SLAM estimates. The main difference to usual navigation algorithms is the attention given to the uncertainty of the estimates. A trade-off is done between the rapidity of the exploration - or more generally, the energy used for the exploration - and the quantity of uncertainty on the SLAM estimates. This trade-off comes from the inherent concept of loop-closures that exist in SLAM algorithms. The loop-closures are the main process to lower the uncertainty on the SLAM estimates (i. e., the localization and the map), but to trigger loop-closures, the robot has to re-visit an already known place. The most effective exploration consists in visiting only once each part of the area, in this case, no loop-closures can be triggered because the robot never re-visited a place. Consequently, the uncertainty of the SLAM estimates keep increasing. A typical example of an active-SLAM algorithm would be to force the robot to re-visit some places in order to bound the increase in uncertainty while keeping a satisfactory exploration of the area. The study [KTT13] describes a multi-robot active SLAM system. The goal of multi-robot active SLAM algorithms is to deliberately trigger inter- and intra-robot loop closures to bound the increasing uncertainty. Therefore, the optimization problem to solve can be described as follow: How to explore an area as fast as possible (regarding the time or the number of steps) with multiple robots using SLAM estimates while

minimizing the uncertainty of the SLAM estimates.

Combined localization, mapping and path planning

We would like to underline that none of the previously cited studies provide a satisfactory solution that deal simultaneously with localization, mapping (or SLAM), navigation, exploration and network for a fleet of UAVs using only an embedded monocular camera and an IMU. However, the work described in [Ach+12] seems very promising to design an effective solution to combine SLAM and navigation. Localization, mapping and path planning are also the focus of [Loi+16] which describes a system composed of a swarm of flying agents (smartphones embedded on quadrotors). All the sensors are placed on the flying platform and are limited to an IMU and a monocular camera. The localization in space is done through two separate filters: a EKF (Extended Kalman Filter) [Sim06] and a UKF (Unscented Kalman Filter) [JU04]. The main novelty lies in the path planning algorithm to prevent drone collisions. The robots also participate in the collaborative feature-based sparse map construction. The target assignment, map merging and loop closure detection are performed on a ground station. However, the trajectory planner is on board each quadrotor. This approach uses the map to correct the drift in localization estimation (the path planning is badly affected by an inaccurate localization) which is a similar approach to SLAM. The method is theoretically scalable, however the computational requirements practically limit the size of the swarm.

Heterogeneity in the TSoS

An interesting prospective in mobile autonomous TSOS and a further step for collaborative mobile multi-robot systems is the incorporation of heterogeneity in the fleet in order to increase the possible missions. In [PHK16], a solution is offered to deal with the additional complexity of heterogeneous TSoS. The study demonstrates the impact of diversity in robot swarms. The authors propose a model to allocate the robots to tasks regarding the robot capabilities (traits) and a

set of tasks which require various specific traits to be achieved.

2.2.9 Conclusion to the survey of approaches for mobile fleet of robot systems

In this first part of the literature survey chapter, we have presented the existing methods to solve the underlying scientific problems to the DIVINA challenge team as well as the overall context in which the M-SLAM work takes place. We discussed approaches that bring answer to the perception problem for fleets of autonomous robots (localization, mapping and SLAM), the medium level control (path planning and collision avoidance) and the high level of control parts of the fleets (exploration and area coverage), and the network parts to ensure communication within the fleet of robots in order to make the ro bot capable of collaboration.

In the second part of this literature survey, we focus on single robot SLAM approaches: The main input of the M-SLAM system. We provide a definition of the SLAM problem, and the different approaches to solve it, then we focus on visual monocular SLAM concepts. We also introduce the mechanism of loop-closure that is critical in the M-SLAM work. In addition, we explain in detail some of the latest and most promising monocular and inertial-monocular SLAM algorithms.

2.3 Single robot Simultaneous Localization and Mapping (SLAM)

2.3.1 Simultaneous localization and mapping: problem definition

Simultaneous Localization and Mapping (SLAM) is a well known problem in robotics defined in the 90s [SK08]. By definition, SLAM is an optimization problem. The goal is to find the optimal configuration of the map m and the robot pose estimates x_t to minimize the robot motion U_t and the sensor measurements Z_t as

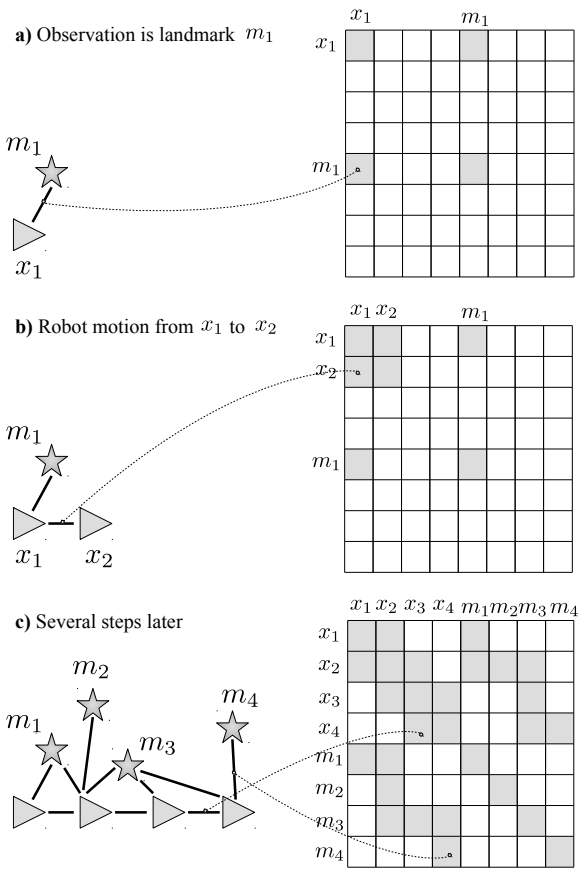


Figure 2.1: Graphical model of the SLAM problem, shaded nodes are observable values and white nodes are the unknown variables the SLAM algorithm should recover from [SK08].

illustrated in Figure 2.1

$$\underset{x_t, m}{\operatorname{argmax}}\ p(x_t, m | Z_t, U_t) \tag{2.1}$$

SLAM algorithms can be complex and highly dependent on the sensors involved. Lately, we have seen great improvements in visual SLAM solutions which are now able to be embedded on robots. However, these solutions suffer from many drawbacks such as lack of robustness to specific movements, the low output rate or the insufficient accuracy of the estimates [SMD10] [WSS11]. For those reasons, several research teams continue to work on the visual SLAM to offer new algorithms and better performance. We cover a number of representative examples in the following paragraphs.

To solve the SLAM problem, three main paradigms can be used [SK08]: Extended Kalman Filter (EKF) [Sim06], particle filters and graph-based optimization

methods [KK16].

Chronologically, EKF was one of the first method used to solve the SLAM problem. It estimates the robot state composed of the robot pose and the features associated with the map landmarks. The state vector that represents the problem is composed of two parts: The first part is used to store the robot pose and remains constant in size, the second part is used to store the features of the map and keeps increasing in size as long as new areas of the environment are discovered and processed. By definition, EKF provides the uncertainty of the map and pose estimates, and take into account the correlations between the robot pose and the features. The state and the covariance matrix, which represents the uncertainty of the system, are updated at every incoming measurement. When a new feature is measured, the robot state is increased as well as the covariance matrix. In contrast to older robot pose estimates which are not recorded (only the most recent pose is used), features are stored because they may be observed again from future locations.

Measuring the same features several times is critical in SLAM because it triggers loop-closures. Loop-closures bring additional information to the system and allow, through an optimization or a data-fusion process to decrease the uncertainty of some of the estimates and therefore, to refine the map and pose estimation. Consequently, loop-closures participate in increasing the global accuracy of the estimates. When the SLAM algorithm is embedded in a robot and no absolute measurements are available, the map and pose estimates drift continuously. This phenomenon can be seen through the increasing uncertainty along the robot path. In such case, the only solution to bound the uncertainty is to detect loop-closures. However, loop-closure detection requires to solve a data association problem. Because of the drift, it can be hard to decide whether a feature is a new one or a known one. The most usual method to solve the data association problem is to used Mahalanobis distance. Mahalanobis distance provides a metric based on the proximity (how far the two landmarks are from each other) and the uncertainty (how inaccurate the

measurements are). Obviously, other methods can be paired with the Mahalanobis distance such as descriptors in visual SLAM approaches which can qualitatively describe the feature in order to match it later. Note that data association is also required for measurement triangulation to create the 3-D map in visual SLAM algorithms.

The main problem with EKF-based methods for SLAM is the quadratic augmentation in the covariance matrix size which limits the system to consider only few features. A large covariance matrix requires memory space and, more importantly, more computational time to be updated. In other words, the global size of the map is seriously limited. One of the most promising approach to overcome the size limitation is to split the global map into local submaps to keep a reasonable size for the covariance matrix.

The second SLAM paradigm is particle filter. Particle filters use representative samples to compute the posterior distribution. A sample is composed of particles, which represent a realist guess about a possible value of the state, i.e., the robot pose and the landmark locations. However, the number of required particles can quickly become a critical problem. This number grows exponentially with the state space dimension, and, in SLAM, the state space dimension depends on the number of observed features. Consequently, a map composed of multiple features, e.g., hundreds or thousands features, is way too large to be handled by a particle filter. A solution to overcome this problem is to break down the maps into low-dimension sequences to maintain a controllable number of particles [Mon+02]. Particle filter solutions for SLAM have several benefits: They can be used for on line SLAM, similarly to EKF they use only the most recent robot pose estimate and data association is straightforward. Another non negligible benefit is the ease implementation of the method. However, the required number of particles remains a major drawback.

The third SLAM paradigm is graph-based optimization methods. In this category of methods, the SLAM is modeled with a graph, the nodes are the unknown

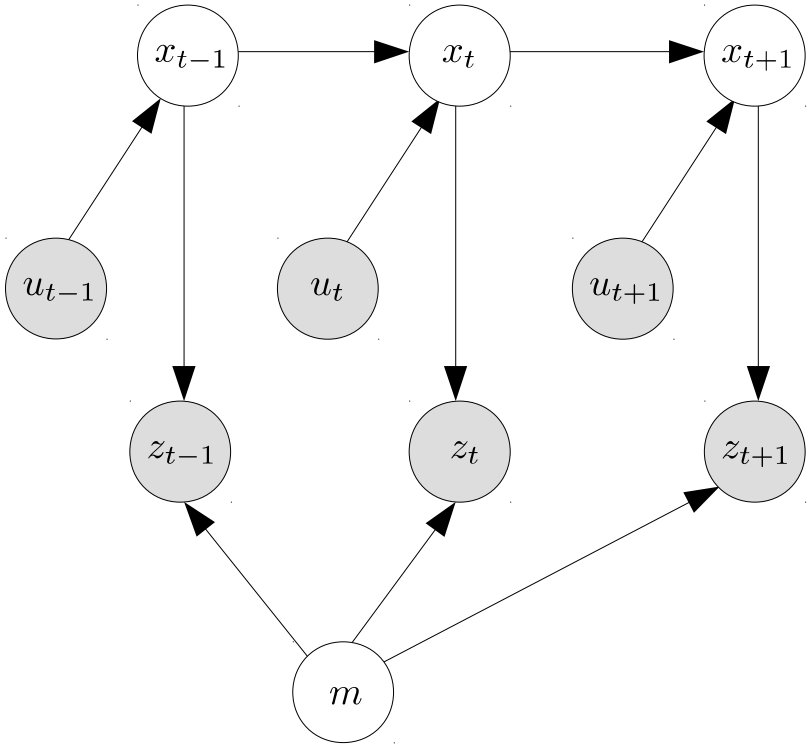


Figure 2.2: Illustration of the graph and the matrix construction from [SK08].

variables that have to be estimated (the robot pose and the landmarks of the map), and the edges represent soft constraints of the odometry and sensor measurements. A sparse matrix of constraints can be filled with the edge information. The matrix is sparse because odometry measurements are available only between two consecutive robot poses and the features can be observed only from a limited number of locations. Figure 2.2 shows how the graph is built and why the related matrix is sparse. Several effective algorithms can be found in the literature to solve this kind of optimization problem such as gradient descent, Gauss-Newton or Levenberg-Marquardt algorithms.

The loop-closures are added to the problem as new graph constraints, and the data association problem is usually solved using a RANSAC method [FB87]. The main benefit of graph-based optimization approaches is the ability to model large map and long robot paths. The memory space grows linearly and the computational time to update the graph is constant (node and edge insertion). However, the solution of the optimization problem itself can be very demanding in resource and computational power, particularly for long paths. This is the reason why

graph-based optimization methods were first used to solve full SLAM in post-processing or off line computation (in contrast to on line SLAM which computes map and pose estimates with the measurement taken on the go). However, the circumstances have changed lately with the introduction of incremental solutions for sparse nonlinear graph-based optimization problems which can compute real-time estimates [KRD08] [Kae+12]. Among the three SLAM paradigms, graph-based optimization methods are the most effective for large-scale localization and mapping.

In conclusion, most of the current SLAM solutions are constrained to static environments of limited size though active research is done to extend the SLAM algorithms to large-scale or dynamic environments. Multi-robot SLAM systems have attracted increasing attention lately and significant advances in graph-based optimization methods have stimulated research for real-time large-scale visual SLAM solutions.

2.3.2 Loop-closures in single-robot SLAM

As introduced in Section 2.3, loop-closures are an important part in SLAM. They are critical when there are no absolute measurements as being the only process to bound the accumulating drift. The loop-closure detection problem is defined as the recognition by the system of locations the robots has visited at least two no-consecutive times.

When the robot re-visit the same place, this piece of information can be used to improve the location estimates and the map as shown in Figure 2.3. However, the process is not straightforward. First, a detection system must be designed to find good candidates for closing loops. Usually, this is done by searching for similar features such as comparing descriptor in visual SLAM approaches. Once a loop candidate has been detected, it is included in the optimization problem. There are numerous ways for including the loop-closures such as filters or graphs, Section 2.3 provides a detailed overview of the main approaches.

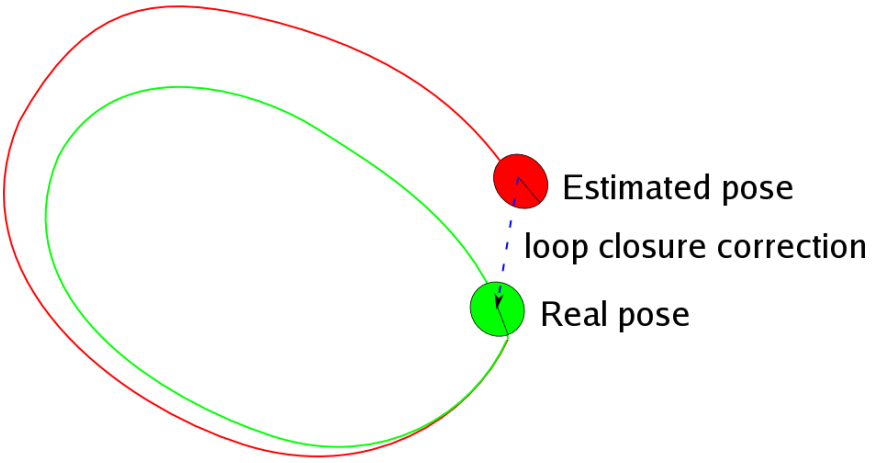


Figure 2.3: Using loop-closures in the SLAM problem from [Ang+08].

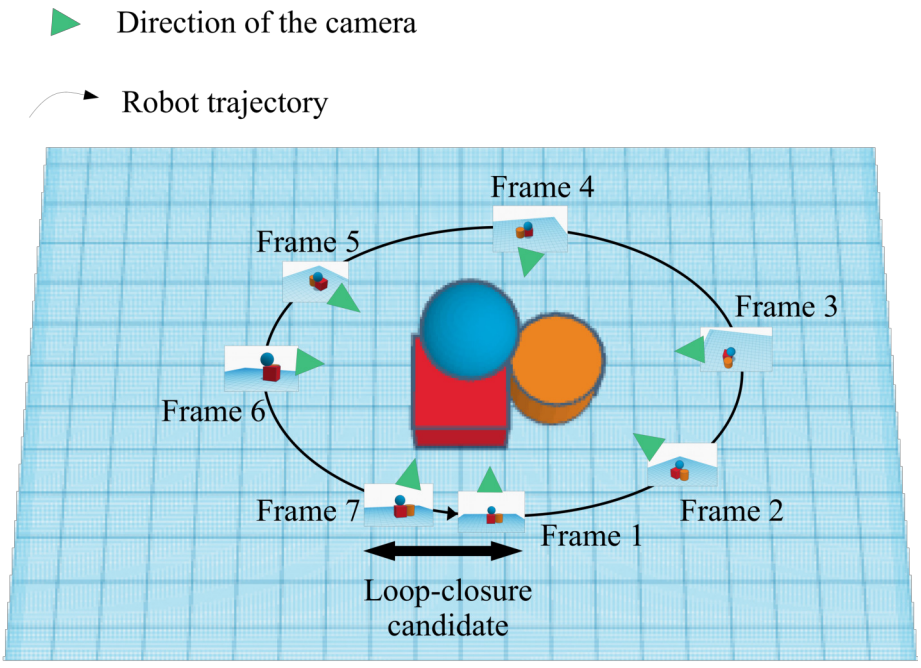


Figure 2.4: Example of the loop-closure detection process.

There are many way to handle loop-closures, working mainly on the map, the location or both and regarding the kind of sensors used to measure the localization and environment. For example, some visual SLAM approaches use bundle adjustment to benefit from the loop-closures, the re-projection error of the map is minimized to improve both the map points and the location estimates.

Figure 2.4 provides an example of a loop-closure detection, Figure 2.5 represents

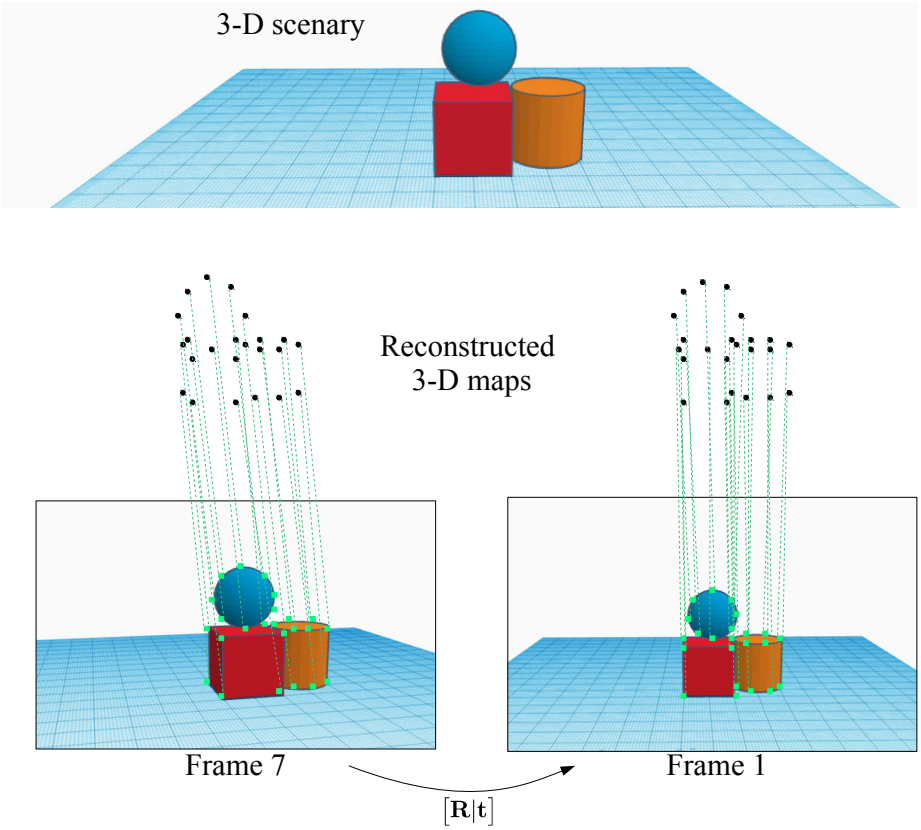


Figure 2.5: Example of the map re-construction for loop-closure processing.

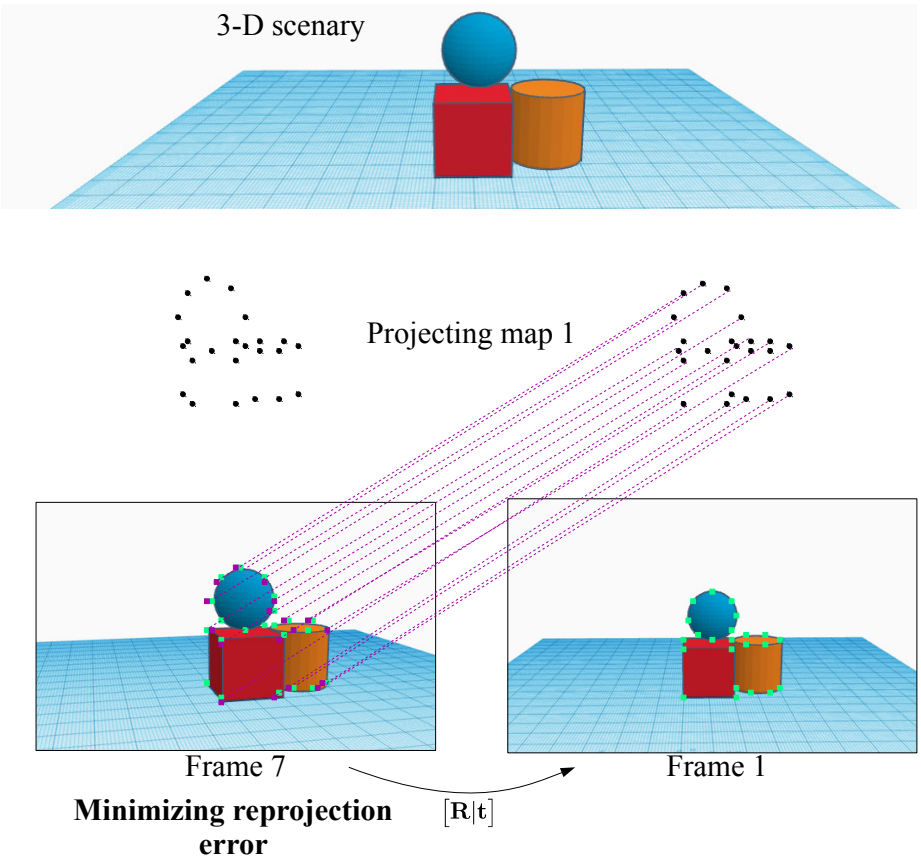


Figure 2.6: Example of the re-projection minimization for loop-closure processing.

the re-construction of the map from each frame done by the SLAM algorithm using the features, and Figure 2.6 is an example of the minimization of the re-projection error. For example, using a bundle adjustment framework and minimizing in term of least-square, the difference between the purple squares and the green squares on Figure 2.6 are minimized by moving either the 3-D map points and/or the transformation $\left[\mathbf{R}|\mathbf{t}\right]$ (or in homogeneous coordinates $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ O_{1x3} & 1 \end{bmatrix}$).

2.4 Visual monocular SLAM concepts, approaches and algorithms

2.4.1 Visual SLAM introduction

In this Section, we focus on SLAM approaches with a specific focus on monocular SLAM, real-time visual SLAM and inertial-monocular SLAM. We provide definitions, concepts and typical vocabulary in this field of research. Through this review, we aim to provide solid knowledge about the similarities and differences of representative algorithms.

To introduce the visual SLAM problem, Figure 2.7 provides a block diagram that is suitable for most of the approaches. In the visual SLAM approaches, we usually distinguish the front-end and the back-end as shown in Figure 2.8. The front-end is in charge of estimating the odometry of the robot therefore it is often is charge of the tracking and the location estimation while the back-end is used to solve the optimization problem which can include estimating or refining the map, and the pose estimates.

For refining the estimates, new constraints are considered in addition to the relative motion estimated for the robot. Those constraints are referred as loop-closures (Section 2.3.2). A loop-closure consists in observing the same scenery or landmarks in non-consecutive frames. By revisiting the same place, additional

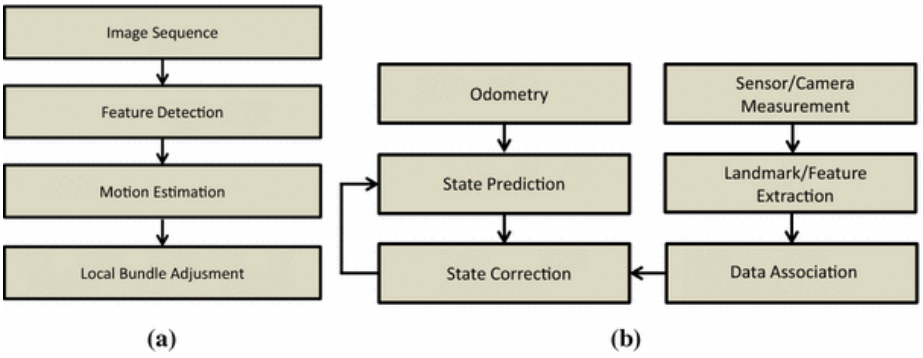


Figure 2.7: Block diagram of a) visual odometry systems and b) filter based visual SLAM systems from [YBH15].

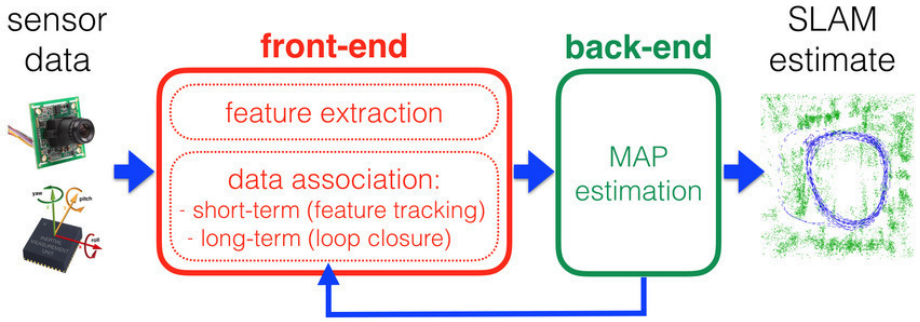


Figure 2.8: Illustration of the front-end and the back-end in a SLAM system from [Cad+16].

constraints can be added to the trajectory and can be used to bound the drift that is accumulating throughout the trajectory because of the aggregation of relative motion estimation from the visual tracking. In the traditional SLAM structure, loop-closures are handled in the back-end as they are an important part of the optimization process.

2.4.2 Problem definition of the visual SLAM

In monocular SLAM algorithms, the goal is to find x, the camera pose, and m, the position of the map points, by measuring either image features or pixel intensities z. There are a couple of related scientific challenges. First, image processing is a time and computational consuming task. In order to embed vision-based algorithms on small UAVs, a particular attention must be given to the performance and resource consumption of the considered algorithm. Second, monocular systems suffer from the inherent scale ambiguity. The world scale cannot be observed, nevertheless,

a consistent map has to be estimated along the UAV trajectory. The map is built by chunks using the incoming frames (at maximum one chunk per frame). By definition, each chunk of map can have a different scale. Therefore, a unifying scale factor needs to be estimated in order to bring consistency to the map, i.e., bigger objects in real world have also to be bigger in the map representation even if they were reconstructed using different frames.

Lately, most monocular SLAM solutions use pose graph optimization methods rather than filters (this point is discussed in the Section 2.4.3). In graph-based monocular SLAM approaches similarities can be observed. Almost all recent monocular SLAM [MMT15] [ESC14] [CC15] are composed of two or three threads: Tracking, mapping, and sometimes loop-closing threads. The tracking thread provides a first estimate for the camera pose. Then, the mapping thread benefits from this pose estimate to update the 3-D map. Finally, loop-closure constraints are researched and, when appropriate, added to the graph and the constraint is propagated to refine the pose and map estimates. The pose graph is usually optimized by minimizing either the re-projection error (in feature-based approaches) or the photometric error (in direct approaches). Re-projection and photometric errors are based on a similar concept that is illustrated in Figure 2.9. The tracking part estimates the relative camera pose between two consecutive frames. In both re-projection and photometric errors, paired pixels are compared. Let us consider a pixel $\mathbf{p}_{F_i}$ in the frame F_i. A map point $\mathbf{m}_{F_i}$ is associated to the pixel $\mathbf{p}_{F_i}$. The map point $\mathbf{m}_{F_i}$ can be projected into the frame F_j using the relative camera pose between the frames F_i and F_j. Thus, we can pair the pixels $\mathbf{p}_{F_i}$ and $\mathbf{p}_{F_j}$ because they should represent the same 3-D map point. In direct methods, the photometric error is the subtraction of the pixel intensities $\mathbf{p}_{F_i}$ and $\mathbf{p}_{F_j}$. If $\mathbf{p}_{F_i}$ and $\mathbf{p}_{F_j}$ do represent the same map point, they should have similar intensities and the photometric error should be nearly zero. The camera pose is refined by minimizing the photometric error, i.e., find what is the relative camera pose which minimizes every photometric error of the pixels in the frames

$$\min_{\mathbf{T}_{F_i,F_j}} \sum_k \left[\mathrm{I}(\mathbf{p}_{k,F_j}) - \mathrm{I}(\omega(\mathbf{p}_{k,F_i}, \mathbf{T}_{F_i,F_j}, \mathbf{m}_{k,F_i})) \right]^2 \tag{2.2}$$

where I the pixel intensity, $\mathbf{p}_{k,F_i}$ the k^{th} pixel in frame F_i, $\mathbf{m}_{k,F_i}$ the 3-D map points associated with $\mathbf{p}_{k,F_i}$, ω the warping function which projects pixels $\mathbf{p}_{k,F_i}$ in the frame F_j using the relative camera pose $\mathbf{T}_{F_i,F_j}$.

In feature-based methods, the keypoints are paired by matching their descriptors. The re-projection error is the distance between the projection of the k^{th} map point $\mathbf{m}_{k,F_i}$ into the frame F_j and the pixel $\mathbf{p}_{k,F_j}$. The camera pose is refined by minimizing the re-projection error

$$\min_{\mathbf{T}_{F_i,F_j}} \sum_k \left[\mathbf{p}_{k,F_j} - \pi(\mathbf{T}_{F_i,F_j}, \mathbf{m}_{k,F_i}) \right]^2 \tag{2.3}$$

where $\mathbf{p}_{k,F_j}$ the k^{th} keypoint in frame F_j, $\mathbf{m}_{k,F_i}$ the 3-D map points associated with $\mathbf{p}_{k,F_j}$, the keypoints of frame F_i, π the projection function which projects keypoints $\mathbf{p}_{F_i}$ in the frame F_j using the relative camera pose $\mathbf{T}_{F_i,F_j}$.

Note that the extraction of features is not perfect though, which introduces noise and may cause a non zero re-projection error while the camera pose is perfectly estimated, similarly, two pixels from different frames which represent the same map point can have slightly different intensities.

2.4.3 Filter and graph-based optimization solutions for visual SLAM algorithms

Two main approaches are used to solve the SLAM problem: Filters using Extended Kalman Filter (EKF) or particle filters and optimization methods like Bundle Adjustment (BA). In [SMD10], the pros and cons for each approach are discussed. In Kalman filtering methods, the computation of the current state (map and pose estimates) is based on the previous state estimate and the current measurements. The history of the pose estimates is not recorded, but each new pose considered causes all the previously linked features to inter-connect. For example, in Figure

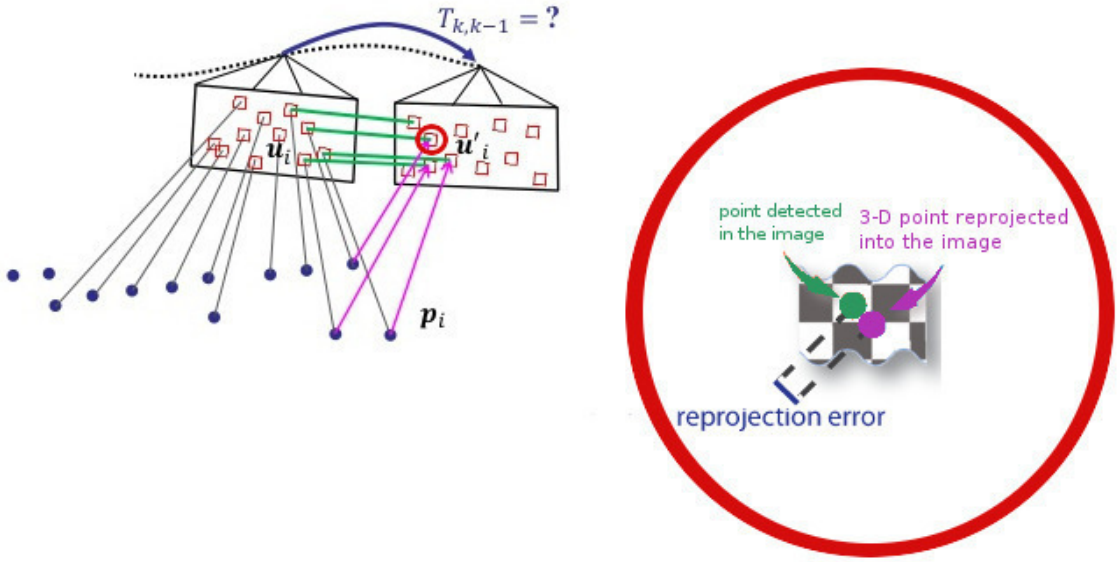


Figure 2.9: Re-projection error schema between two consecutive frames k and $k-1$. The red circle is a detailed view around one of the features. $\mathbf{T}_{k,k-1}$ is the relative robot pose between the two frames, red squares are the extracted features, green lines represent the feature tracking and matching, $\mathbf{p}_i$ are the 3-D map points and purple lines represent the projection of the map points into the frame k. From [19d] and [19c].

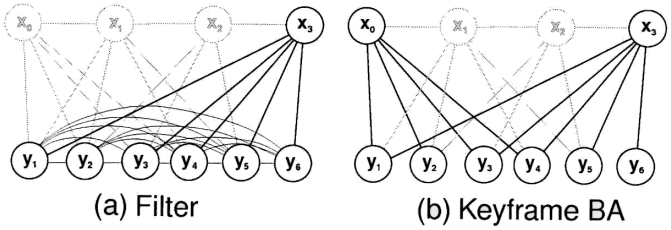


Figure 2.10: Inference progression in a filter (a) and in keyframe-based optimization (b), x are the robot pose and y the map features from [SMD10].

2.10, when the pose x_1 is estimated, the edges between the features $\{y_1, y_2, y_3, y_4\}$ and the pose x_0, from which they were observed, are removed (because the pose x_0 no longer exists); then, the features $\{y_1, y_2, y_3, y_4\}$ are connected with the features $\{y_1, y_2, y_4, y_5\}$ observed from the current pose x_1. In contrast to poses, the features have to be recorded because they can be measured again in the future. In BA optimization methods, both the features and the poses are recorded. Therefore, there are no inter-connection between the features. An edge between a feature and a pose is established if the feature can be observed from the according camera pose.

Computational power and memory are limited, therefore, two aspects are

studied, the size of the robot state (map and pose) and the required computation to update the state using incoming measurements. As previously discussed in 2.3, EKF has the major drawback of quadratic increase in the addition of new features. It is an effective method if the amount of features remains small. Graph optimization methods have not this drawback. Nevertheless, the inclusion of the multiple previous robot poses increase the size of the graph that has to be optimized. The solution used to control the graph size is to consider only poses associated with keyframes, a subset of frames. In addition to this solution, the graph can also be partially optimized by considering only a subset of nodes and edges.

In [SMD10], the authors demonstrate that the accuracy of the monocular SLAM is better improved by increasing the number of features rather than the number of frames. This conclusion supports graph-based optimization with regard to filtering methods.

2.4.4 Factor graph methods

Factor graphs [KFL01] are a novel approach to solve the SLAM problem based on graph optimization [Gri+10]. The focus is set on the use of a different architecture: Consequently, the three usual threads, tracking, mapping and loop-closing are not used. A main benefit of factor graphs is the visual representation which is quite easy to understand. Moreover, different type of measurements can elegantly be integrated. Firstly, the method was used only to solve full SLAM in post-processing. However, the implementation of incremental solver like iSam [KRD08] [Kae+12] have made the approach suitable for real-time estimation. The GTSam framework is available to implement factor graph based solutions [Del12]. Figure 2.12 shown the visual output that can be obtained using the GTSam framework [19b] on a simple SLAM example. Factor graphs are bipartite graphs composed of factor nodes and variables nodes as displayed in Figure 2.11. The variable nodes represent the unknown variable one wants to estimate. The variable nodes are linked through factor nodes. A factor node is probabilistic information about

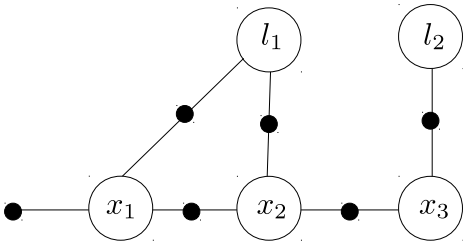


Figure 2.11: Factor graph for landmark-based SLAM from [Del12].

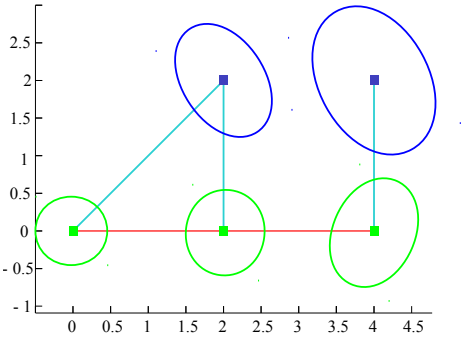


Figure 2.12: The optimized result along with covariance ellipses for both poses (in green) and land- marks (in blue). Also shown are the trajectory (red) and landmark sightings (cyan) from [Del12].

one, or several, variable nodes computed from measurements or prior knowledge. For example, a factor node can represent the error between an estimate current value and prediction. The prediction is calculated using the measurements, the previous state estimates and theoretical models, e.g., the differential equations which describe the system.

2.4.5 Visual SLAM architectures and algorithms

In monocular SLAM, two approaches for the visual processing are mainly used: Direct methods and feature-based methods. The main difference lies in the method to track pixels (in direct methods) or features (in feature-based methods), Figure 2.13 provides an overview of the differences between direct and feature-based methods. A third architecture can be designed through factor graph models (Section 2.4.4).

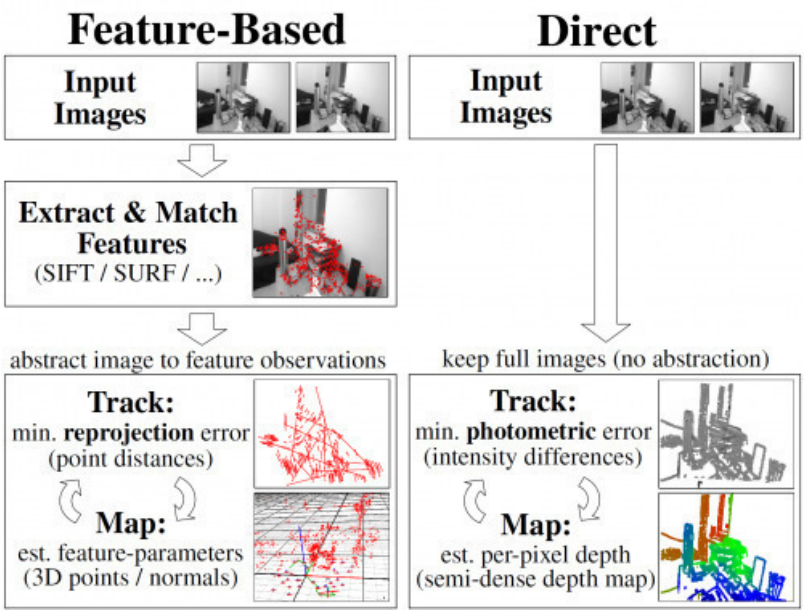


Figure 2.13: Overview of feature-based and direct SLAM methods from [ESC14].

2.4.6 Direct methods

Direct methods use pixel intensities. They benefit from all the information contained in the image rather than constraining the choice of features to a subset of landmarks. Direct methods outperform feature-based methods in poorly textured environments and are not dependent of the choice for the feature descriptor. They also allow dense mapping more easily (a depth value can be estimated for most pixels of every frame). However, dense mapping is computationally very demanding and, currently, there is no suitable methods for real-time dense mapping embedded on highly constrained platform in term of resource and computational power. A reference work in dense monocular parallel tracking and mapping is the DTAM algorithm [NLD11]. DTAM is one of the first real-time monocular system which creates a dense 3D surface model and, more importantly, uses it for the camera tracking, and thus does not rely on feature extraction and benefit from whole image information. Despite the claim of real-time performance, a GPU is required to handle the heavy computational load. Therefore, lately, most direct methods offer semi-dense mapping to soften the resource constraints for real-time applications. The DPPTAM [CC15] can be seen as an improvement of DPTAM. DPPTAM offers

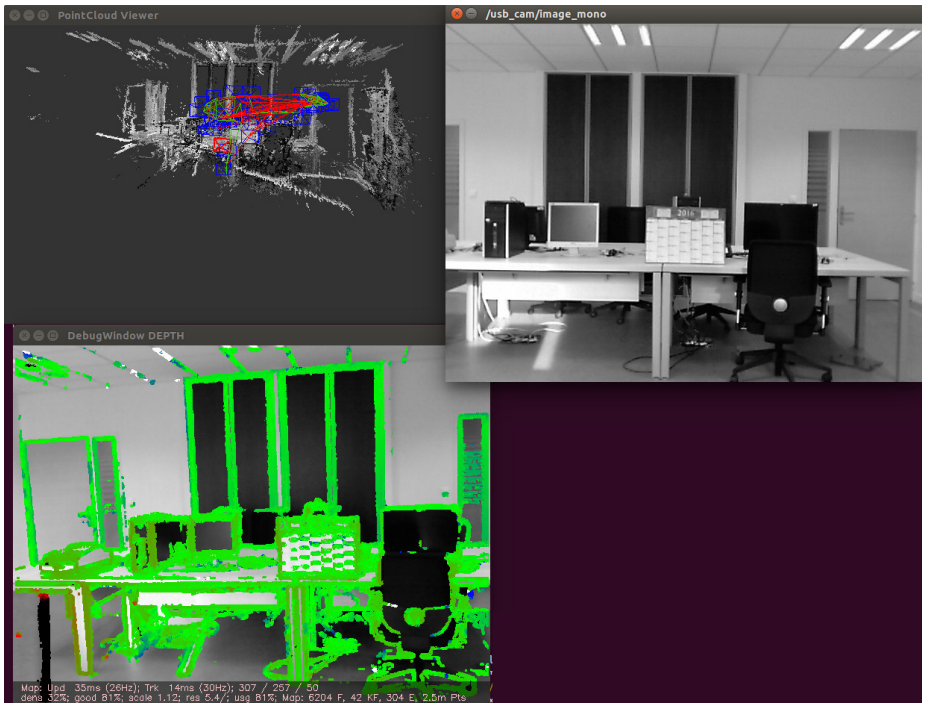


Figure 2.14: Running the LSD-SLAM.

a dense reconstruction of the environment while tracking the camera pose. The computational requirements are lowered by using keyframes and superpixels for planar untextured surfaces. Three threads run in parallel: The tracking, semi-dense mapping and dense mapping threads. The tracking thread estimates the camera pose. The semi-dense mapping thread estimates a partial map using high-gradient pixels and feeds in the tracking thread for the photometric error computation. The dense mapping threads benefit from superpixel approaches to provide a dense map at reasonable computational cost.

2.4.7 LSD-SLAM algorithm

Overview and architecture

One of the latest most promising real-time monocular direct SLAM solution is the LSD-SLAM algorithm [ESC14] (and, then, its improved version called DSO [EKC18]), an illustration of the LSD-SLAM algorithm running is provided in Figure 2.14. The LSD-SLAM is a real-time direct monocular SLAM algorithm which allows to build large-scale maps. The LSD-SLAM algorithm is composed of the usual

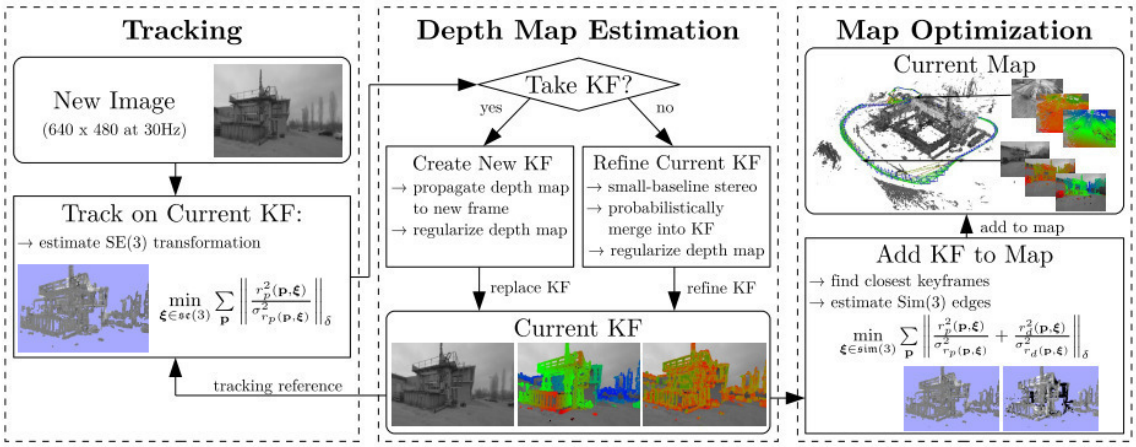


Figure 2.15: Overview of the LSD-SLAM algorithm from [ESC14].

three SLAM modules:

- The tracking module tracks the camera movement using direct monocular visual odometry and outputs an estimate of the camera pose.
- The depth map estimation module (which takes as input the camera pose) estimates an inverse depth value (and its uncertainty) for the most relevant pixels of the image (high-gradient pixels).
- The map optimization module searches for graph constraints, i.e., loop-closures or scale constraints, and optimizes the SLAM pose graph.

Figure 2.15 provides an overview of the structure of the LSD-SLAM algorithm.

Tracking

The frames (coming from the camera video stream) are the main information used to track the camera poses. Some of the frames are promoted to keyframes in the depth map estimation component. These keyframes construct the skeleton of the SLAM (they are used as vertices in the SLAM graph and for the depth map estimation). A frame is promoted into a new keyframe when the camera movement becomes sufficiently wide with regard to the last keyframe. The tracking component continuously tracks the camera pose with respect to the current keyframe, it computes the 3-D translation and rotation of the camera.

Let us take an example considering two images, a keyframe KF and a frame F, the tracking component computes the 3-D transformation $[\mathbf{R}|\mathbf{t}]$ (rotation and translation) between KF and F. The images are composed with gray-scale pixels (each pixel intensity is a value between 0 and 255). The pixels from KF are associated with the pixels from F and the intensity difference is computed. If equivalent pixels were picked (pixels representing the same 3-D point in both images), the intensity difference should be near to zero. The data association is done using a warping function, a 3-D projective function that maps KF pixels with F pixels. The warping function depends on the pixels, the depth value of the KF pixels and the $[\mathbf{R}|\mathbf{t}]$ transformation. The depth value is known (it is computed by the depth map estimation component) but the $[\mathbf{R}|\mathbf{t}]$ transformation remains unknown (we are looking for it), though a first guess is calculated using a motion model.

The goal of the tracking thread is to find the value of $[\mathbf{R}|\mathbf{t}]$ for which the intensity difference between the mapped pixels is minimum. The optimization problem is solved using the Gauss-Newton algorithm [BF95]. It allows to find the value for the parameters $\mathbf{R}$ and $\mathbf{t}$ such that the observations fit the best (in our case, the pixel intensity residuals). Outliers due to occlusions, reflections and limited size of the frames are taken into account, thus to make the optimization more robust, a weight matrix is also computed to down-weight large residuals, i.e., large intensity differences.

There is another point that has to be taken into account, the depth noise can differ significantly for each pixels (because of the use of a monocular direct method for estimating the depth). The depth variance depends on how long the pixels can be observed in consecutive frames and the type of camera motions, e.g., translation movements affect the depth variance of pixels in specific directions). Therefore, the residuals are normalized with the variance of the photometric error. This variance is computed using standard propagation of uncertainty. The way the depth noise is incorporated into the tracking is one of the major novelty proposed by the LSD-SLAM algorithm. Finally, a Huber norm $\|.\|_\delta$ is used to lower the sensitivity to

outliers [Hub64]

$$\mathrm{E}_p(\boldsymbol{\xi}_{ji}) = \sum_{\mathbf{p}\in\Omega_{D_i}} \left\| \frac{r_p^2(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p},\boldsymbol{\xi}_{ji})}} \right\|_\delta \tag{2.4}$$

where

$$r_p(\mathbf{p}, \boldsymbol{\xi}_{ji}) := \mathrm{I}_i(\mathbf{p}) - \mathrm{I}_j(\omega(\mathbf{p}, \mathrm{D}_i(\mathbf{p}), \boldsymbol{\xi}_{ji})) \tag{2.5}$$

and

$$\sigma^2_{r_p(\mathbf{p},\boldsymbol{\xi}_{ji})} := 2\sigma_I^2 + \left(\frac{\partial r_p(\mathbf{p}, \boldsymbol{\xi}_{ji})}{\partial \mathrm{D}_i(\mathbf{p})} \right)^2 \mathrm{V}_i(\mathbf{p}) \tag{2.6}$$

where E_p, the function to minimize, r_p the photometric residuals and σ the standard deviation. $\mathbf{p}$ represents the normalized pixel coordinates, $\boldsymbol{\xi}$ the camera pose, I the image intensity, ω the warping function, D the inverse depth map and V the map of inverse depth variance.

Depth map estimation

The second component is the depth map estimation module. Its first task is to promote the current frame to keyframe if the requirements are completed, i.e., the relative robot pose exceed a threshold which combines relative distances and angles. The second task is to create and refine the depth maps (a map that assigns a depth value to each 2-D pixel). A depth map is not created for each frame, only the keyframes have one. When a new keyframe is created, the associated depth map is initialized by projecting the points of the previous keyframe (the very first keyframe is initialized with a random depth map and large variance, which allows the algorithm to converge after a few of keyframes, finding a better initialization remains for further work). Then, all the frames that are not promoted to keyframes are used to refine the depth map with small baseline stereo comparison. The depth error can be very different according to the considered pixel because it depends of the neighborhood of the pixel and the movement of the camera. Therefore, the depth error is never considered as constant, property which is specific to monocular approaches (in contrast to RGB-D approaches). An important remark is that the

depth map created by the LSD-SLAM is semi-dense, all pixels do not have a depth. This choice allows to dramatically speed-up the algorithm. The pixels chosen to have a depth are representative of the scenery (they provide an intelligible map). In monocular SLAM, the absolute scale of the world is not observable, a scale ambiguity remains. Thus, the depth map is scaled to have a mean inverse depth equals to one and the reconstruction scale factor has to be estimated in order to have a consistent map. It means that all distances are defined up to a scale factor that has to be determined in order to have an internally consistent map. The similarity transformation $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ O_{1x3} & 1 \end{bmatrix}$ (scale, rotation and translation) between keyframes is computed in the map optimization component.

Map optimization

The map optimization module performs three main tasks: Find similarity and loop-closure constraints, computes the $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ O_{1x3} & 1 \end{bmatrix}$ (scale, rotation and translation) transformations for keyframe alignment and optimizes the SLAM pose-graph. In the LSD-SLAM algorithm, a novel method is proposed to perform a direct, scale-drift aware image alignment. The scale factor problem is an important question as the accumulated scale-drifts over trajectories constitutes the main source of error. The solution method is almost the same as the one used in the tracking component, it consists of using Gauss-Newton optimization on variance-normalized residuals. The difference lies in adding a depth residual that penalizes depth deviations.

Considering a pixel $\mathbf{p}$ in the first keyframe, an equivalent pixel $\mathbf{p}_0$ can be found in the second keyframe using the warping function (the same function as in the tracking component). The depth residual is the difference between the depth of the $\mathbf{p}_0$ pixel (information given by the warping function) and the value of the depth map for the $\mathbf{p}_0$ pixel. Using both residuals (photometric and depth), the $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ O_{1x3} & 1 \end{bmatrix}$

transformation between two keyframes can be found

$$E_p(\boldsymbol{\xi}_{ji}) = \sum_{\mathbf{p}\in\Omega_{D_i}} \left\| \frac{r_p^2(\mathbf{p},\boldsymbol{\xi}_{ji})}{\sigma^2_{r_p(\mathbf{p},\boldsymbol{\xi}_{ji})}} + \frac{r_d^2(\mathbf{p},\boldsymbol{\xi}_{ji})}{\sigma^2_{r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}} \right\|_\delta \tag{2.7}$$

where

$$r_d(\mathbf{p},\boldsymbol{\xi}_{ji}) := [\mathbf{p'}]_3 - D_j([\mathbf{p'}]_{1,2}) \tag{2.8}$$

$$\mathbf{p'} = \omega(\mathbf{p}, D_i(\mathbf{p}), \boldsymbol{\xi}_{ji}) \tag{2.9}$$

and $[\mathbf{p'}]_{1,2}$ is the pixel defined with the first and second coordinates of point $\mathbf{p'}$, $[\mathbf{p'}]_3$ is the third coordinate of point $\mathbf{p'}$, and finally

$$\sigma^2_{r_d(\mathbf{p},\boldsymbol{\xi}_{ji})} := V_j([\mathbf{p'}]_{1,2}) \left(\frac{\partial r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}{\partial D_j([\mathbf{p'}]_{1,2})} \right)^2 + V_i(\mathbf{p}) \left(\frac{\partial r_d(\mathbf{p},\boldsymbol{\xi}_{ji})}{\partial D_i(\mathbf{p})} \right)^2 \tag{2.10}$$

Here E, the function to minimize, r_d the depth residual, r_p the photometric residual and σ the standard deviation. $\mathbf{p}$ represents the normalized pixel coordinates, $\boldsymbol{\xi}$ the camera pose, I the image intensity, ω the warp function, D the inverse depth map and V the map of inverse depth variance.

In addition to similarity constraints, loop-closure constraints are also added to the graph. In order to find the possible loop-closures, every time a new keyframe is added to the SLAM graph, the geographically ten closest keyframes are determined (using the pose of each keyframe). Then, these keyframes are sent to an appearance-based mapping algorithm, FAB-MAP [CN08]. The FAB-MAP algorithm determines if two keyframes represent the same scene; then, by tracking the camera pose of each matched keyframe, they can be added to the SLAM graph as loop-closure constraint. Unfortunately, FAB-MAP is a feature-based approach, therefore the use of FAB-MAP in a direct methods such as LSD-SLAM alter some of the benefits provided by the absence of features, such as, avoiding corner detection and descriptor extraction. The use of FAB-MAP in the loop-closure detection process shows the difficulty to detect similar visual places without using features and the difficulty to find a

satisfactory solution to the data association problem in direct methods. Finally, the SLAM graph is optimized using a pose graph optimization method. The graph is composed of keyframes as vertices and constraints as edges (the computed $[sR|t]$ transformation and the loop-closures).

2.4.8 Feature-based methods

Feature-base approaches are the most usual way to solve the monocular SLAM problem. Features are particular pixels and their neighborhood that are detected, described and saved in data structures called descriptors. When a new frame is fed in, an extractor detects the image features. The descriptor describes each keypoint (pixel) neighborhood such that it can be uniquely identified. When required, a matching process can compare two feature descriptors to determine whether the features represent the same 3-D object. Being able to match image features make the camera pose tracking easier to compute and more instinctive.

In feature-based monocular SLAM research, the PTAM algorithm must be presented. PTAM stands for parallel tracking and mapping. This algorithm demonstrated the benefits of keyframe based approaches and the decoupling possibilities between the tracking and mapping threads. Moreover, PTAM showed real-time performance while using Bundle Adjustment optimizations, a standard structure-from-motion optimization approach usually used in off line post-processing because of the heavy computational requirements. PTAM was published in 2008, several outperforming SLAM algorithms have been released since. However, the PTAM skeleton can still be identified in most of these current monocular SLAM algorithms.

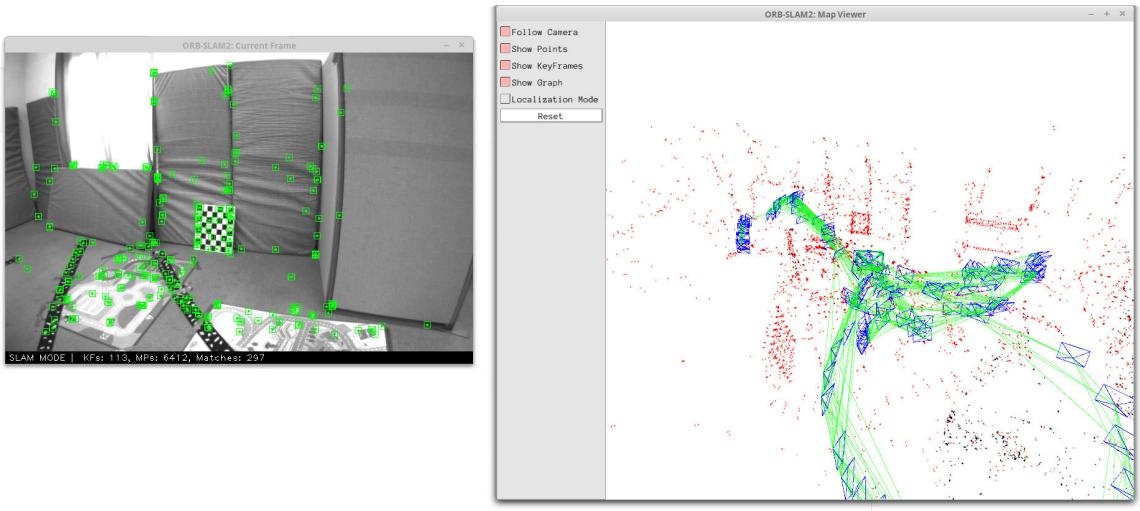


Figure 2.16: Running the ORB-SLAM. The map is in red, the camera pose estimates in blue and the covisibility graph in green.

2.4.9 ORB-SLAM algorithm

Overview and architecture

ORB-SLAM is a feature-based method, an illustration of the ORB-SLAM algorithm running is provided in Figure 2.16. The pose and map estimates are refined using the Bundle Adjustment method to optimize the SLAM graph. Actually, there are three graphs built to fill different purposes. The covisibility graph represents the links between the keyframe poses and the map points. Each node stores a keyframe, edges are set if two keyframes share sufficient observations of the same features. Edges are weighted regarding the number of shared observation of the features. Despite the obvious usefulness of the covisibility graph, the high edge density can be hard to handle for some computation steps. A spanning tree is also constructed. The spanning tree is the minimal connected subgraph of the covisibilty graph, the edges of higher weight are kept. The third graph is the essential graph which is used for loop-closure process. The essential graph is intermediate between the covisibility graph and the spanning tree. It is a subgraph of the covisibility graph but only the edges of high weight are kept. The main benefit is to limit the computation load when loop-closure are propagated. In ORB-SLAM, like most other monocular SLAM algorithms, three threads run in parallel: Tracking, local

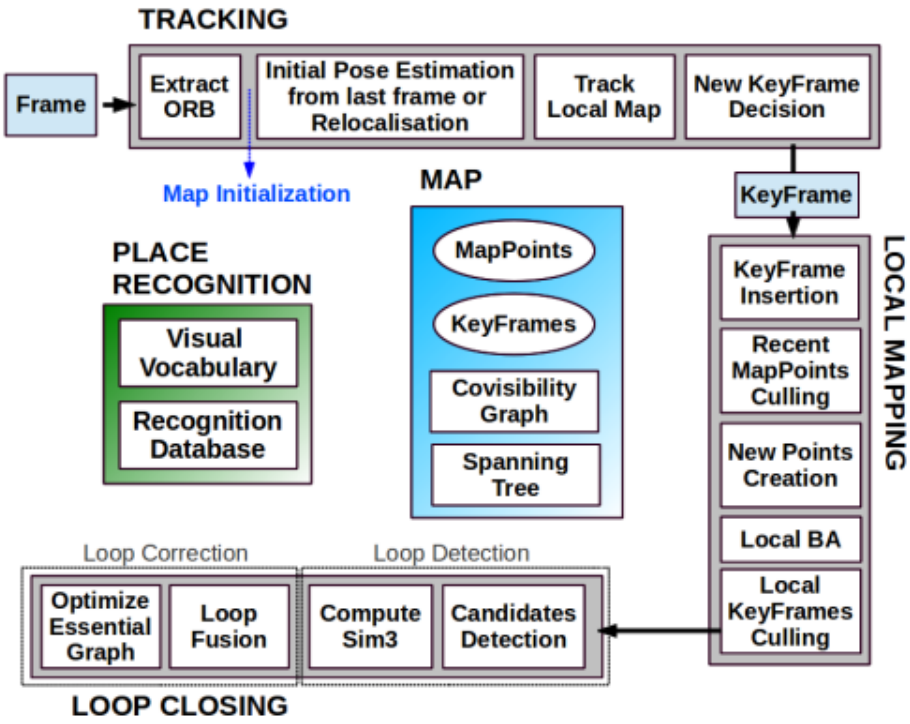


Figure 2.17: The structure of the ORB-SLAM algorithm from [MMT15].

mapping and loop closing. Figure 2.17 provides an overview of the structure of the ORB-SLAM algorithm.

Tracking

The tracking is performed over every incoming frame. Features are detected and recorded using 256 bits ORB descriptors. ORB are oriented multi-scale FAST corners [Rub+11] which outperform the commonly used SIFT [Low04a], SURF [BTV06] or A-KAZE features [ANB11] regarding the extraction time. A first pose estimate is computed using the previous frame pose and a motion model. Then, the pose estimate is improved by matching the features to determine the relative camera motion. The current map is used to search additional features. The final step of the tracking component is the possible election of the current frame into a keyframe. A lot of frames can be elected into keyframe because the algorithm provides a mechanism to discard inefficient keyframes. The keyframe election is based on two main criteria, the new keyframe must be sufficiently different from the previous one and track enough feature to ensure continuity of the tracking. Note that if

the tracking is lost, a re-localization system is called to recover the camera pose if known places are recognized again. After a successful re-localization stage, the tracking can continue as usual.

Local mapping

When a new keyframe is elected, the local mapping thread processes it to update the map estimate. A very restrictive policy is used to prevent any map point outliers including the use of epipolar constraints, map projection into the frames and ORB descriptors matching. Only very good quality features are used to create the sparse map. The depth of keypoints is computed by triangulation (at this stage the keypoint positions and the relative camera motion are known). A system based on bags of words [GT12] which uses the ORB descriptors helps to solve the data association problem required for the pixel triangulation, the re-localization function and the loop-closure detection. A local Bundle Adjustment is performed to optimize the map and pose estimates in the neighborhood of the current keyframe. The neighborhood is determined using the covisibility graph, i.e., the keyframes that observe the same features as the considered keyframe. Local Bundle Adjustment is performed over a subset of keyframe and map points to limit the computational requirement for the optimization. The last step of the local mapping thread is the keyframe culling, which removes insufficiently informative keyframes.

Loop-closure detection and processing

The third thread detects, and when it is possible, performs loop-closures. The candidates are searched based on visual similarity. A recognition database stores the visual words associated with the detected features, and the keyframes they were observed from. This data structure allows fast queries to get a visual similarity metric between the keyframes. Thus, the most similar keyframe that is not directly connected (regarding the covisibility graph) to the current keyframe can be found. Then, by using a similar method as in the tracking thread, the 3-D map points

of the involved keyframes are paired. A similarity transformation is computed using a standard method, RANSAC is used to search candidates and the method of Horn [Hor87] calculate the similarity transformation. If the number of inlier is satisfactory, the loop is closed. An edge is added in the covisibility graph between the two keyframes, the paired map points are corrected to fit and the local keyframe poses are also corrected accordingly. This step makes both loop edge to fit. Finally, the loop closure error is propagated along the whole trajectory. Basically, the poses are optimized through similarity transformations in the essential graph and the map points are corrected accordingly.

2.4.10 Monocular SLAM with regard to the M-SLAM research work

In the M-SLAM system, we assume each drone to be equipped with a monocular camera and an IMU. Every robot of the fleet runs a real-time monocular SLAM algorithm to represent a part of the environment and to localize itself. At the moment we started the M-SLAM work, there were two leading real-time monocular SLAM algorithms which were freely available: LSD-SLAM [ESC14] and ORB-SLAM [MMT15]. LSD-SLAM is a direct method that uses pixel intensities to estimate the map and the robot pose. On the other hand, ORB-SLAM is a feature-based method. We performed a series of tests to estimate the robustness of both methods to UAV movements. ORB-SLAM outperforms LSD-SLAM in term of tracking robustness which is a crucial point because when the tracking is lost, map and robot poses cannot be computed or updated. For this reason, we decided to focus on feature-based approaches for our experiments. Note that both ORB-SLAM and LSD-SLAM programming code are designed for the middleware ROS (Robot Operating System) and they use a multi-thread architecture.

2.4.11 A tightly-coupled monocular-inertial SLAM: VINS-Mono

Context

In 2018, and to our knowledge at that time, the first open-source inertial-monocular real-time SLAM algorithm robust enough to be run on board UAVs was published. This publication created a dramatic improvement in the M-SLAM work progress as the metric scale of the SLAM estimates was not a problem to solve anymore. The algorithm described in [Lin+18] called VINS-Mono provides a robust monocular visual-inertial state estimator that match the inputs we were looking for to design our multi-UAV system. We managed to replicate the results claimed in the publication and compared them with our loosely-coupled approach to estimate metric distances (the approach is detailed in Chapter 4). VINS-Mono is a promising approach for our research topic, therefore, we decided to use it in our experimentation.

System overview

VINS-Mono [Lin+18] offers a robust tightly-coupled monocular-inertial SLAM. The approach can be divided into the following points:

- The initialization stage that allows to robustly bootstrap the system,
- A tightly-coupled fusion between monocular vision and inertial measurements to estimate the 3-D pose and map of the system,
- Loop-closure detection to bound the inherent drift of the odometry due to the absence of absolute measurements,
- Pose-graph optimization for handling large-scale trajectories and to ensure consistency.

Figure 2.18 provides an overview of the structure of the VINS-Mono algorithm. Thanks to the use of inertial measurements, the pose and map estimates are metric.

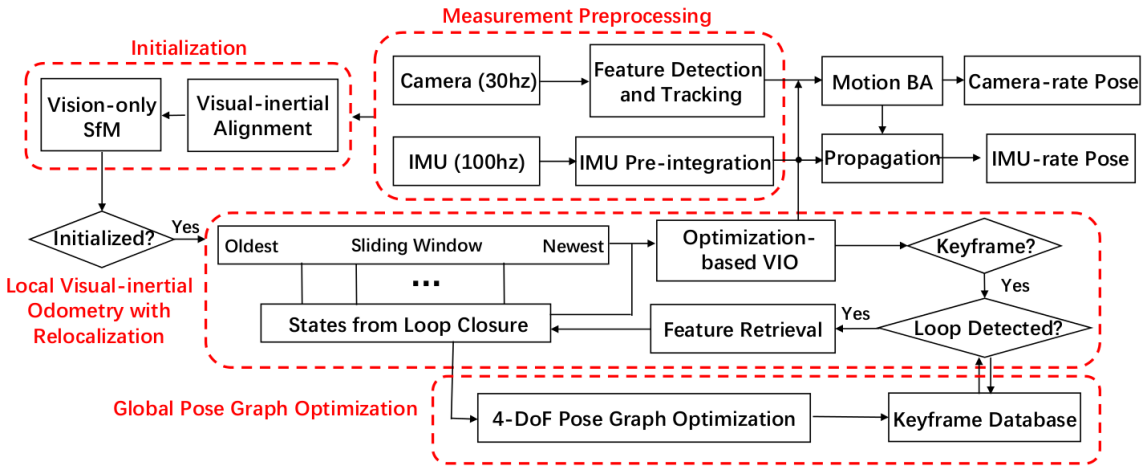


Figure 2.18: The block diagram of VINS-Mono from [Lin+18].

Initialization for bootstrapping the system

The initialization stage is an important part in SLAM. The quality of the initialization ensure the convergence of the system. The goal of the initialization is to provide initial values for the pose, scale, velocity, gravity vector and IMU biases. A loosely-coupled fusion is done between between the output of a vision-only Structure-from-Motion method and the inertial pre-integrated measurements. For the visual part, features are extracted and tracked in two consecutive frames. By using the Five-point algorithm [Nis04], the motion between the two frames is estimated up to scale. If the result is good enough, other consecutive frames are added to the graph. Their poses is estimated through a PnP algorithm [LMF09] using the result given by the two initial frames. The scale is set arbitrarily, the final value of the scale is computed by fusing the inertial measurements. In order to minimize the re-projection error, a bundle adjustment is run. For the inertial part, the inertial measurements are pre-integrated following an extended version of the methods proposed in [LS12] and [SMK15] that provides a first estimate of the pose and velocity between two instant of time. The IMU pre-integration is linearized and transformed into a measurement model. The alignment between the inertial and visual measurements is done by solving a least-square problem as illustrated in Figure 2.19.

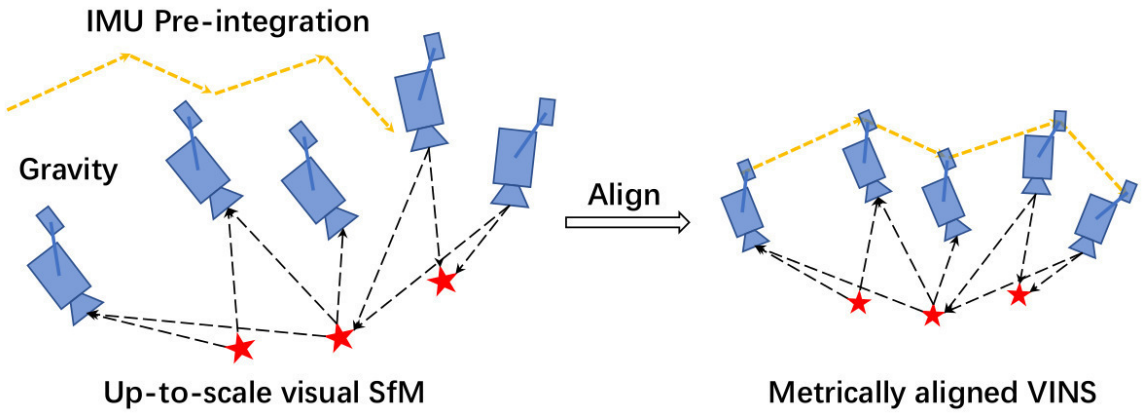


Figure 2.19: Illustration of the visual-inertial alignment for the initialization from [Lin+18].

Monocular-inertial fusion for state estimation

The core of VINS-Mono is the sliding-window monocular-inertial estimator. Figure 2.20 illustrates how the measurements are fused within the sliding window. The state vector is composed of three main parts per keyframe in the window: the states related to IMU (3-D pose, velocity and bias), the states related to the pose estimation by the monocular vision (3-D pose up to scale) and the states that represent the map (each point in a keyframe is represented in the state vector by its reconstructed inverse depth). Because the size of the window is constant (e.g. ten keyframes), the size of the state vector is bounded (it does not increase with the length of the trajectory like in some other SLAM approaches). The monocular and inertial measurements are fused through a bundle adjustment scheme. Technically, the sum of the prior, visual and inertial pre-integrated residuals is minimized using the nonlinear solver Ceres [AM12]. The IMU pre-integration residuals are computed similarly to the initialization. The visual residuals are calculated by re-projected the features in the other keyframes. The features are tracked using the sparse optical flow algorithm KLT [LK81]. A Shi-Tomashi corner detector guarantees that between 100 and 300 corners are available in the images and are uniformly distributed. In order to limit the computational need, a subset of images is elected as keyframes based on the following criteria: The average parallax with the previous keyframe and the tracking quality. The prior residual comes from a marginalization step

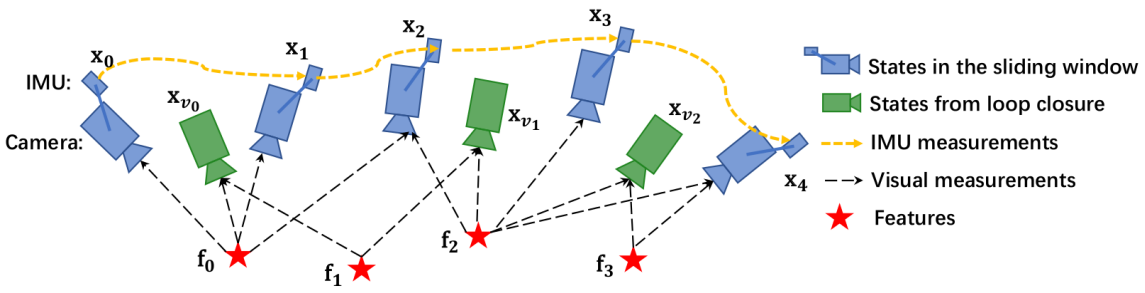


Figure 2.20: An illustration of the sliding window monocular VIO from [Lin+18].

that transform some IMU states and features into a prior in order to limit the computational complexity. Bundle adjustment can be heavy from a computational point view, therefore, VINS-Mono use a lightweight motion-only optimization that allows to be roughly ten times faster than usual state-of-art bundle adjustment based approaches.

Loop-closure detection and closing

VINS-Mono offers a mechanism to detect and close loop-closure in order to bound the drift of the system (that would be an open-loop system drifting freely otherwise). The detection of loop-closure relies on features. The bags-of-Word approach [GT12] is used to represent efficiently the descriptors of the features of each keyframe. Thanks to the Bags-of-Word representation, a similarity score is computed between the keyframes, if the score is above a threshold, a loop-closure candidate is detected. Then, the two keyframes are processed. The feature descriptors are paired (i. e. BRIEF descriptors). This pairing result in numerous outliers that must be filtered. The pairs of keypoints and descriptors are processed first with 2D-2D fundamental matrix RANSAC algorithm and second with a 3D-2D PnP RANSAC scheme. If there are enough inlier at the end of the processing, the loop-closure candidate is considered as a loop-closure. Once a loop-closure is detected, the pose of the new keyframe (the one that triggered the loop-closure) and the other keyframes are estimated similarly to the method used in the VIO estimator, except that a new residual term is added to the minimization problem which represents the loop-closure constraint.

Pose-graph optimization

Pose graph optimization is used to ensure the global consistency of the pose and map estimates. After a loop-closure occurs, the current keyframes are shifted, therefore the global consistency is decreased. The pose graph is composed of keyframes as vertex connected by edges. The edges represent either the relative transformation between two consecutive keyframes or a loop-closure constraint between two non-consecutive keyframes. A residual function is attached to each edge. The graph optimization consists in minimizing those residuals. The size of the graph grows with the length of the trajectory, though some nodes (keyframes) are regularly discarded in order to keep the size of the graph as small as possible. The criteria to discard nodes is based on the relevancy of the keyframe (the keyframe is discarded if it is close to a loop-closure or if it has a similar pose to a consecutive keyframe).

Experimentation

We performed experimentation to check the usability of VINS-Mono and to attempt to replicate the results claimed in the publication. The source code of VINS-Mono is open-source [19e]. The easiest way to run the system is to use the middleware ROS (Robot Operating System) [18c]. VINS-Mono performed well on the EuRoC dataset [Bur+16] and provided the most accurate results we have seen so far in our experimentations. The tracking is robust and loop closures are detected throughout the trajectory and bound the drift. We computed the RMSE between the trajectory estimated by VINS-Mono and the ground truth given by Vicon motion capture system and compared it with our approach discussed in Section 4.

2.4.12 The OKVis algorithm

Another algorithm offering to fuse visual measurements with inertial measurements for single-robot SLAM is described in [Leu+15]. This approach, named

OKVis, was initially designed for stereo vision, though it was later extended to monocular vision. It performs a tightly-coupled fusion using a nonlinear optimization process to solve the SLAM problem. In order to keep a satisfactory processing time, the keyframes are marginalized and a sliding-window bound the number of keyframes and landmark considered for the optimization process. The experiments were conducted in various outdoor environments using a rigid platform that embedded an IMU and either a stereo camera or a monocular camera. The system was not tested on robots or UAVs and does not describe a loop-closure scheme. In the M-SLAM system we describe in this thesis, using OKVis as an input would be an interesting experiment to compare with VINS-Mono and remain for further work.

A recent comparison (2018) between the existing visual-inertial odometry approaches is provided in [DS18]. The authors compare the accuracy of the position estimates and the resource usage of the main algorithms running on datasets recorded on board UAVs. The experimental results of this study shows a better accuracy in the position estimates for the algorithm VINS-Mono in comparison with OKVis.

2.4.13 Conclusion to single robot SLAM systems

Simultaneous localization and mapping (SLAM) is a problem defined by the robotic community decades ago. SLAM consists in a optimization problem to find the best map and robot pose with regard to the measurements.

Numerous approaches provide solutions to the SLAM problem regarding the sensors and the platform involved. These solutions can be classified in: Filter-based solution (mostly Kalman filter type solutions and particle filters) and graph optimization solutions.

Visual SLAM is a particular category of SLAM problem as the main measurements are brought by visual sensors such as monocular or stereo cameras. Research for satisfactory solution for real-time visual SLAM keep being done to fit with difficult constraints such as the computation power available on the platform, the

processing time, the accuracy of the estimates, or the estimation of the uncertainty.

During the M-SLAM research work, we focused mainly on two algorithms for our experimental part: The ORB-SLAM algorithm and the VINS-Mono algorithm due to their good performance processing measurements taken on board the UAVs.

2.5 Conclusion

The M-SLAM research work crosses several scientific fields and use concepts related to them. In this chapter, we covered all the fields that we used to design the M-SLAM system: the SLAM as defined in robotics with a particular emphasize to monocular and real-time SLAM algorithms, inertial-monocular measurements fusion with application to localization and 3-D reconstruction. We also covered the work done in mobile robotic TSoS for the multi-UAV part of the M-SLAM algorithm. The literature review of TSoS was also of great benefit to define what was required with regard to the applications targeted by the M-SLAM system and to fit with the specifications of the DIVINA challenge team.

We provided a detailed description of three recent and promising algorithms: The LSD-SLAM, the ORB-SLAM and the VINS-Mono algorithms. After testing the algorithms on a UAV datasets, we discarded direct methods such as LSD-SLAM because of the lack in the tracking robustness. Both ORB-SLAM and VINS-Mono suit well the M-SLAM system inputs. The main difference lies in the fact that the ORB-SLAM algorithm must be coupled with a fusion scheme for making metric the map and pose estimates, while VINS-Mono include a tightly-coupled scheme that makes the pose and map output metric. We designed a loosely-coupled fusion scheme experimented with the ORB-SLAM algorithm (as VINS-Mono was not available at that moment) as described in Chapter 4, and then compared it with the results given by the newly published VINS-Mono algorithm. As VINS-Mono provides more accurate estimate than our loosely-coupled fusion, we decided to use VINS-Mono in priority to experiment the M-SLAM system. A thorough

comparison between both input systems remains fur future research.

CHAPTER THREE

DATASETS AND EXPERIMENT DESIGN

3.1 Problem definition and review of the available datasets

Perception algorithms can be run using dataset inputs because the datasets can emulate sensor data. However, in order to obtain satisfactory results, the datasets must represent a similar environment as the one we target our system to be run in. At first, we used the KITTI vision benchmark [Gei+13], but it was recorded on a car which dynamics is very different to a UAV's and the IMU measurements were way much accurate than the ones we get from our quadrotors. We also used the TUM vision benchmark [Stu+12], which can be used for monocular SLAM benchmarking. However, the inertial measurements are not provided and the camera is not placed on the UAV, once again, the fast rotations and aggressive maneuvers of UAVs cannot be observed on this dataset. We would like to underline the importance of having an imagery recorded on board a UAV, for example, the LSD-SLAM performs well with the TUM vision benchmark (not recorded from a UAV) but loses the tracking very quickly when used with datasets recorded on board a UAV. The next datasets we used came from one of the ETH Zurich research laboratories [Lee+10] and had the benefits of providing monocular imagery, inertial measurements recorded on board UAVs and ground truth measurements from a motion capture system. Unfortunately, the frame rate was very unsteady and too low which made

it impossible to use for our purposes. However, this work served as a preliminary study to allow, a few years later, the ETH Zurich to publish a new UAV dataset called EuRoC [Bur+16]. The EuRoC datasets provide several sequences recorded from sensors embedded on a UAV. The data is composed of the imagery from the front camera, the inertial measurements from the IMU and a ground truth provided either by a motion capture system (Vicon) or laser measurements (Leica). Finally, a novel dataset was published in April 2017, the Zurich Urban Micro Aerial Vehicle Dataset [MTS17] which is an interesting UAV dataset for further experiments in outdoor environments.

3.2 Emulation of a fleet of UAVs

We designed a localization scheme for a fleet of quadrotors as a Technological System-of-Systems (TSoS). In order to be able to perform experiments, we had to emulate a multi-UAV system. As introduced previously, the EuRoC datasets [Bur+16] provide eleven sequences recorded on board a remotely controlled UAV. The sensor data from the front monocular camera and the IMU are available and synchronized with a fast and accurate motion capture system (Vicon) that we can use as the ground truth for evaluation purposes. The eleven sequences are grouped into three sets, each set was recorded in the same environment. In our experiments, we use one set to represent one fleet of UAV, each UAV of the fleet is represented by one sequence of the considered set. By using this process, we can emulate two experiments that involve three UAVs and one experiment with a fleet of five UAVs. Figures 3.1, 3.2 and 3.3 provide an overview of each UAV's trajectory for a three-UAV experiment using the V01 sequences.

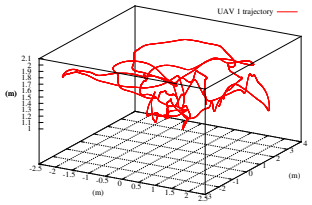


Figure 3.1: V1_01 sequence: trajectory.

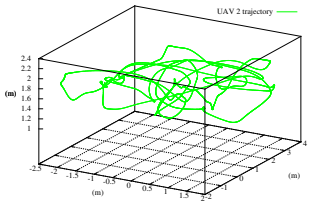


Figure 3.2: V1_02 sequence: trajectory.

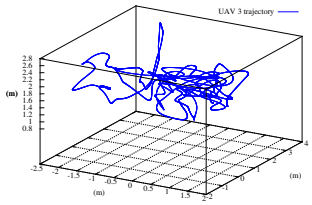


Figure 3.3: V1_03 sequence: trajectory.

3.3 Robotics and vision processing challenges with regards to the M-SLAM system

We want our system to be quite robust to diverse environments. Using the EuRoC datasets, we made our system robust to several points that are usually challenging either in robotics or in perception. The challenging imagery that can be processed by the M-SLAM system is emphasized in the lists below and in Figures 3.4,3.5, 3.6, 3.7, 3.8 and 3.9.

Challenges with regard to Computer Vision

- Blurred images such as in Figure 3.4,
- Abrupt changes in luminosity such as in Figure 3.5,
- Big occlusions and wide change in view point such as in Figures 3.6 and 3.7,
- Challenging textures such as in Figure 3.7, and
- Distinguish the different places that look similar such as in Figures 3.8 and 3.9.

Challenges with regard to robotics and UAVs

- Significant change in speed and altitude in the UAV trajectories, and
- Fast trajectories and aggressive maneuvers.

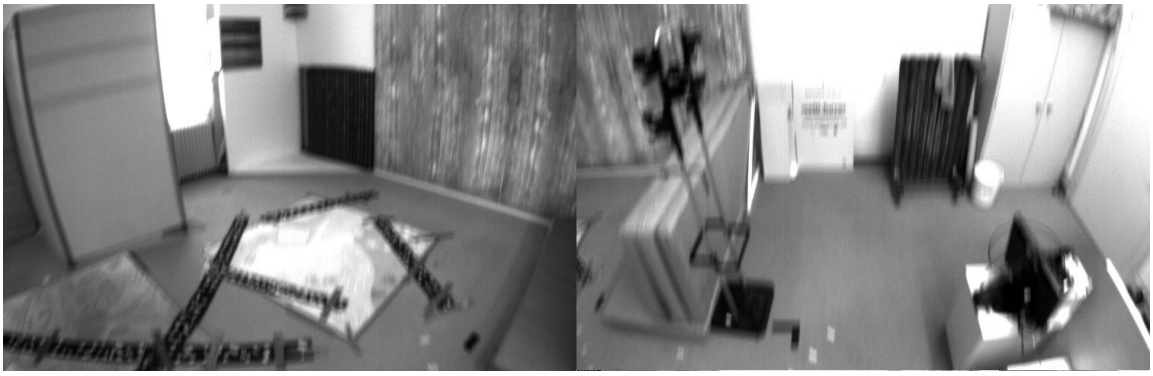


Figure 3.4: An example of blurred images in the EuRoC dataset.

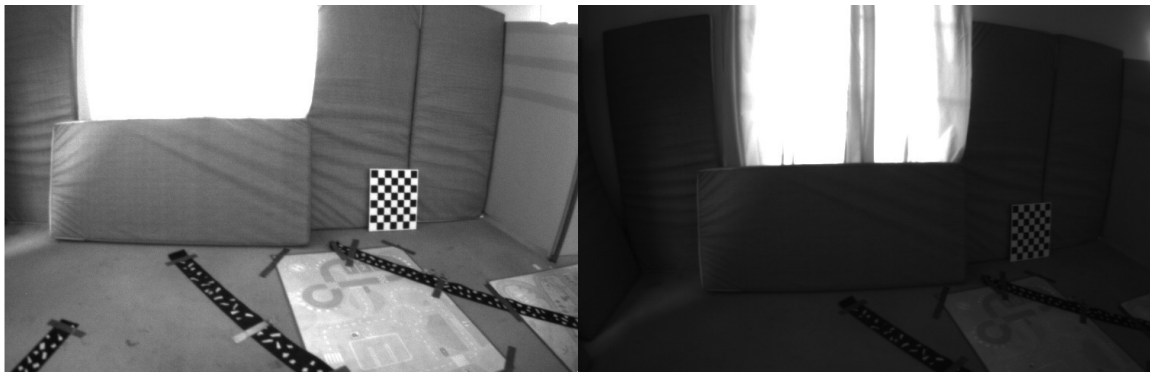


Figure 3.5: An example of dramatic change in luminosity in the EuRoC dataset.

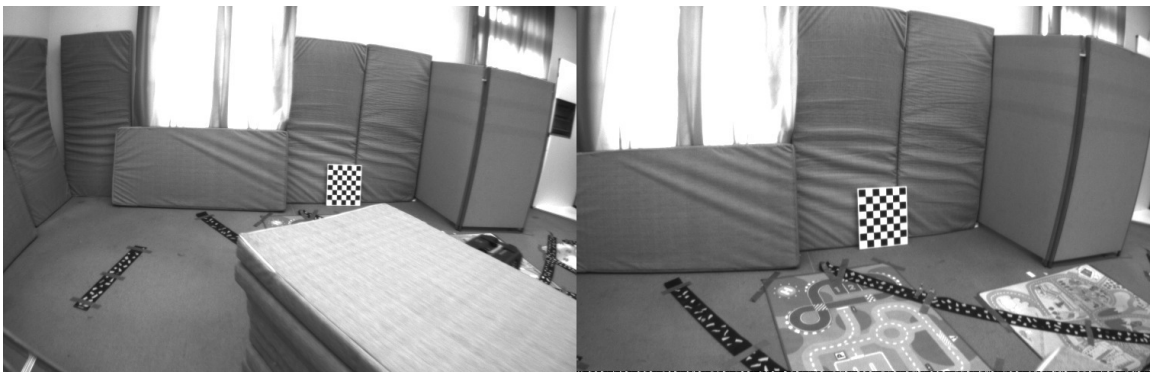


Figure 3.6: An example of occlusion in the EuRoC dataset.

Figure 3.7: Illustration of a big change in view point and the difficulty to process the non-unique texture provided in the EuRoC dataset.

Figure 3.8: An illustration of two different places that look similar, one of the Computer Vision challenge in the EuRoC dataset.

Figure 3.9: Another illustration of two different places that look similar in the EuRoC dataset.

In the EuRoC sequences, there are peaks in instant velocity above 2 m/s with a global average speed of 0.75 m/s. For example, the sequences V1_01, V1_02 and V1_03 that are used to emulate a fleet of three UAVs have the characteristics described in Table 3.1 that can be challenging from a control and robotic point of view. An overview of the fleet trajectory is provided in Figure 3.10, and the changes in velocity (constant velocity is an usual assumption in robotics) are outlined in Figure 3.11.

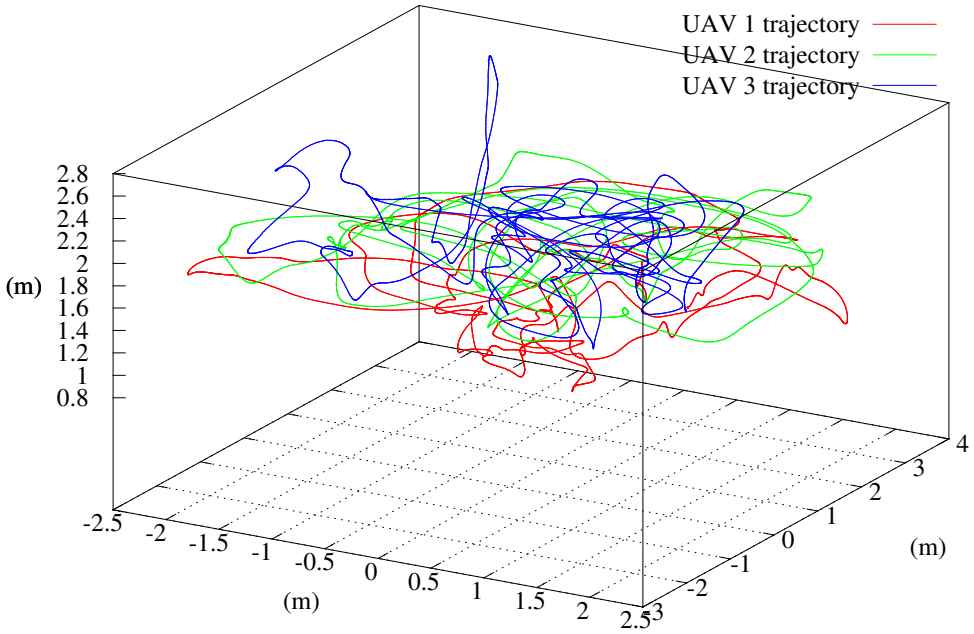


Figure 3.10: The UAV trajectories superimposed to represent the fleet trajectory.

Sequence	Length	Duration	Average velocity	Average angular velocity
V1_01: UAV 1	58.6 m	144 s	0.41 m/s	0.28 rad/s
V1_02: UAV 2	75.9 m	83.5 s	0.91 m/s	0.56 rad/s
V1_03: UAV 3	79 m	105 s	0.75 m/s	0.62 rad/s

Table 3.1: Characteristics of the V1 sequences from [Bur+16].

3.4 About the coordinate frames

3.4.1 Introduction

The coordinate frame part requires particular attention because our work can be included both in the computer vision and in the robotics research communities that use different notations. In addition, throughout this thesis we considered

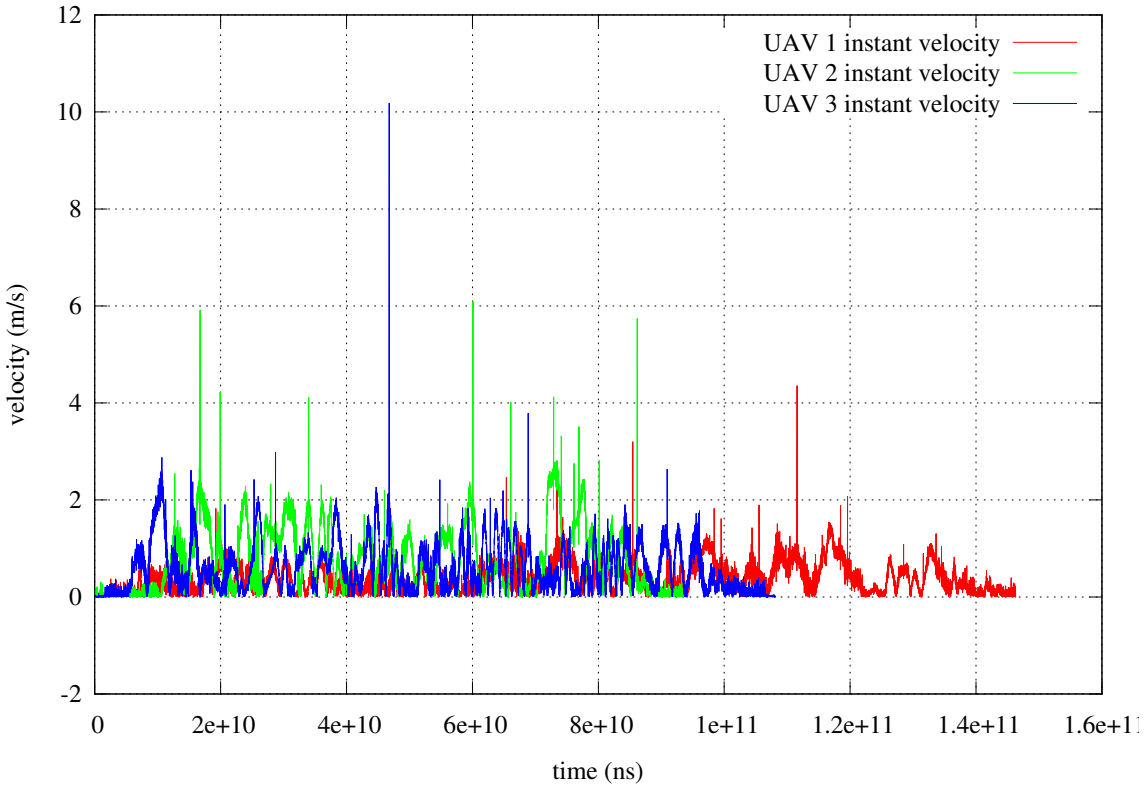


Figure 3.11: Plot of the dramatic change in velocity during the flight.

both single-UAV systems and multi-UAV systems as well as the single sensor case (monocular camera) and the two-sensor case (IMU and camera). As shown in Chapters 1 and 2, we studied monocular single-robot SLAM algorithms, then inertial-monocular single-UAV SLAM algorithms, and finally designed an inertial-monocular multi-UAV SLAM system.

3.4.2 Case 1: Single UAV system using only monocular vision

In single-UAV systems performing monocular SLAM, three coordinate frames must be defined: the camera coordinate frame {CAM}, the UAV coordinate frame {UAV} and the ground truth coordinate frame {GT} that is required for evaluation purposes. From a robotics point of view, the camera coordinate frame can be seen as the body coordinate frame and the ground truth coordinate frame can be seen as the world coordinate frame. Figure 3.12 illustrates the coordinates frames.

If we consider the ground truth coordinate frame as the reference, the UAV coordinate frame is set somewhere after the SLAM initialization and then, remains

static, while the camera coordinate that is attached to the body of the UAV moves when the UAV is flying and can be used to represent the trajectory.

$$^{\mathrm{GT}}\mathbf{T}_{\mathrm{UAV}_a} = \mathbf{M}$$

with $\mathbf{T}$, a 4-by-4 homogeneous transformation matrix that contains both the 3-D rotation and the translation vector and $\mathbf{M}$ a 4-by-4 constant matrix throughout one experiment. The trajectory represented by the UAV poses that are computed by the SLAM algorithm are given by $^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{c}}$. The outputs given by the SLAM algorithm can be assessed by comparing, for example, the difference in position given by the ground truth measurements and the SLAM algorithm,

$$\mathrm{e}_{\mathrm{position}} = \sum_{j=0}^{N} \|(^{\mathrm{GT}}\mathbf{t}_j^{\mathrm{c}} - ^{\mathrm{GT}}\hat{\mathbf{t}}_j^{\mathrm{c}})\|_2^2$$

with $\mathbf{t}$ a 4-by-1 translation vector in homogeneous coordinates and N the number of poses computed by the SLAM algorithm for the UAV, and $\|.\|_2$ the L^2 norm of a vector. $^{\mathrm{GT}}\mathbf{t}_j^{\mathrm{c}}$ is the measurement of the j^{th} position of the camera coordinate frame measured by the ground truth system (for example, a motion capture system like Vicon) in the ground truth coordinate frame, $^{\mathrm{GT}}\hat{\mathbf{t}}_j^{\mathrm{c}}$ is the estimate of the j^{th} position of the camera by the SLAM algorithm in the ground truth coordinate frame.

$$^{\mathrm{GT}}\hat{\mathbf{t}}_j^{\mathrm{c}} = ^{\mathrm{GT}}\mathbf{T}_{\mathrm{UAV}}\, ^{\mathrm{UAV}}\hat{\mathbf{t}}_j^{\mathrm{c}}$$

with $^{\mathrm{GT}}\mathbf{T}_{\mathrm{UAV}}$ computed using the measurement timestamps after the SLAM initialization.

3.4.3 Case 2: Single UAV system using two sensors

In single UAV systems fusing inertial and monocular measurements, another coordinate frame is added in relation to the case 1 in Section 3.4.2. Figure 3.13 illustrates the coordinates frames. The additional coordinate frame, that is the

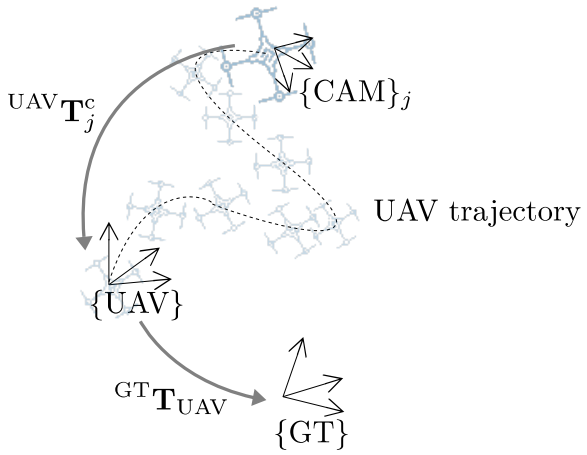


Figure 3.12: Illustration of the coordinate frames involved in a experiment using a single UAV and a single sensor.

inertial coordinate frame {IMU}, is attached to the IMU and is fixed on the UAV frame. Therefore, the coordinate frame representing the body frame could be either the camera coordinate frame or the inertial coordinate frame. We have decided to choose the inertial coordinate frame {IMU} as the body coordinate frame as it is usually done in the robotics community. The transformation ${}^{\mathrm{c}}\mathbf{T}_{\mathrm{i}}$ between the inertial and camera coordinate frames is known and constant because our system is calibrated and both sensors are attached to the same rigid body: The UAV frame. Therefore, we can express measurements ${}^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{c}}$ given in the camera coordinate frame with regard to the inertial coordinate frame, and reciprocally, we can express measurements ${}^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{i}}$ given with regard to the inertial coordinate frame in the camera coordinate frame,

$$ {}^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{i}} = {}^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{c}}\,{}^{\mathrm{c}}\mathbf{T}_{\mathrm{i}} $$

Similarly, any measurement ${}^{c}\mathbf{p}$ (3-D point in homogeneous coordinate) expressed in the camera coordinate frame {CAM} can be expressed in the inertial coordinate frame {IMU} using the calibration matrix ${}^{i}\mathbf{T}_c$ that describes the transformation between the camera and the inertial coordinate frame

$$ {}^{i}\mathbf{p} = {}^{i}\mathbf{T}_c\,{}^{c}\mathbf{p} \tag{3.1} $$

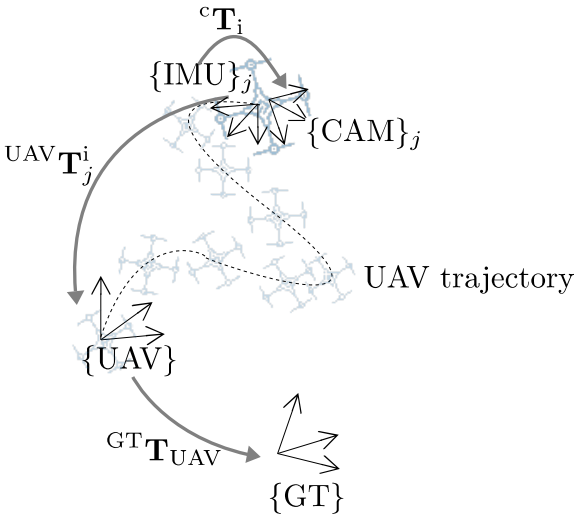


Figure 3.13: Illustration of the coordinate frames involved in a experiment using a single UAV and two sensors.

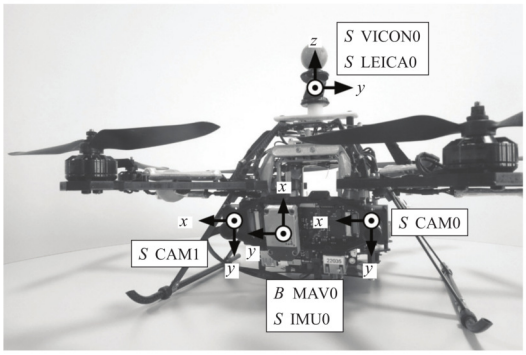


Figure 3.14: The coordinate frames fixed on the UAV from the EuRoC datasets [Bur+16].

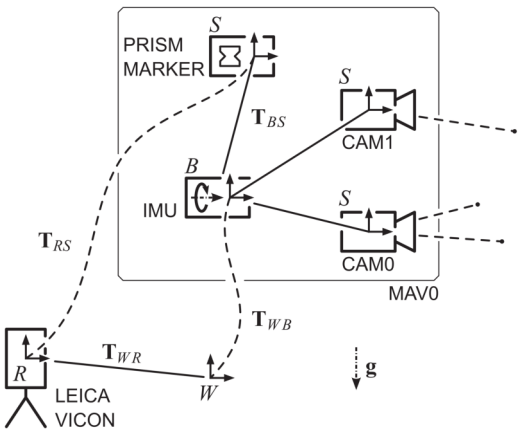


Figure 3.15: Schema of the coordinate frames from the EuRoC datasets [Bur+16].

Figures 3.14 and 3.15 provide an overview of the coordinate frames defined in the EuRoC datasets that we used to perform our experiments. In single UAV systems, the UAV coordinate frame is often superimposed with the world coordinate frame.

3.4.4 Case 3: Multi-UAV system using only monocular vision

The multi-UAV case differs from the previous cases by the addition of as many coordinate frames as the number of UAV of the fleet $\{UAV_a\}$. Figure 3.16 illustrates the coordinates frames. The relation between each UAV coordinate frame is unknown.

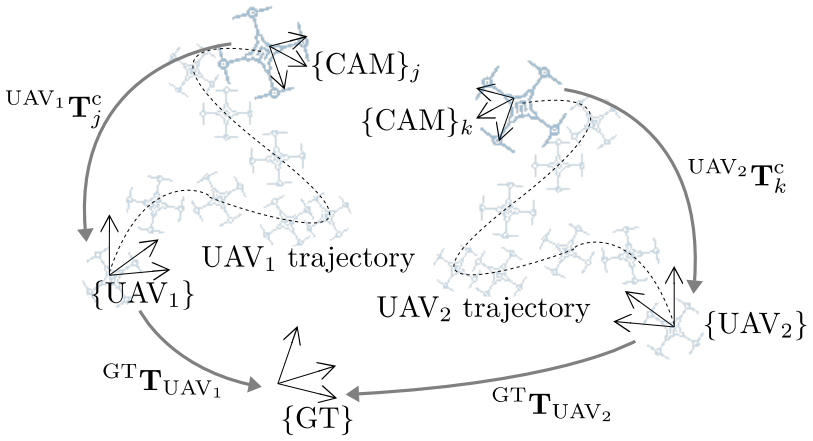


Figure 3.16: Illustration of the coordinate frames involved in a experiment using two UAVs and a single sensor per UAV.

For example, in a three-UAV experiment, there are three UAV coordinate $\{\mathrm{UAV}_1\}$, $\{\mathrm{UAV}_2\}$ and $\{\mathrm{UAV}_3\}$. The transformations $^{\mathrm{UAV}_1}\mathbf{T}_{\mathrm{UAV}_2}$, $^{\mathrm{UAV}_2}\mathbf{T}_{\mathrm{UAV}_3}$ and $^{\mathrm{UAV}_3}\mathbf{T}_{\mathrm{UAV}_1}$ are unknown because there is no a-priori knowledge about the starting position of each UAV and no inter-UAV detection system. The coordinate frames of each UAV are set somewhere in the space after the UAV SLAM initializations.

3.4.5 Case 4: Multi-UAV system using two sensors

The multi-UAV case with multiple sensors is similar to the case 3 in Section 3.4.4 except that we must decide on the sensor that will be the reference. Figure 3.17 illustrates the coordinates frames. Following the same reasoning as for the case 2 3.4.3, we have decided to use the inertial coordinate frame of the considered UAV to be the reference (or body coordinate frame). The transformation between the measurements taken in the camera coordinate frame and the inertial frame is done using the calibration matrix $^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}^{\mathrm{UAV}_a}$. As we built our experiment using several sequences of the EuRoC datasets recorded on board the same UAV, in our experiments, the calibration matrices $^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}^{\mathrm{UAV}_a}$ are the same for each UAV,

$$^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}^{\mathrm{UAV}_1} =^{\mathrm{i}} \mathbf{T}_{\mathrm{c}}^{\mathrm{UAV}_2} =^{\mathrm{i}} \mathbf{T}_{\mathrm{c}}^{\mathrm{UAV}_3} =^{\mathrm{i}} \mathbf{T}_{\mathrm{c}}$$

Figure 3.17 represents the coordinate frames at an arbitrary time t in a two UAV

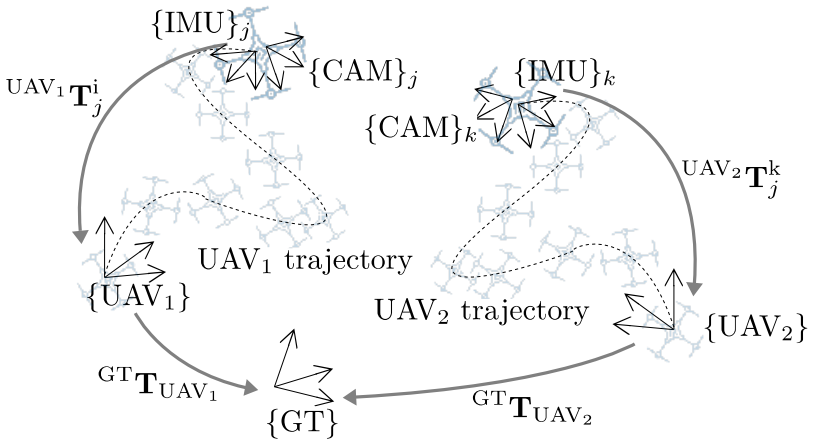


Figure 3.17: Illustration of the coordinate frames involved in a experiment using two UAVs and two sensors.

experiment. After the initialization, the {UAV$_1$} and {UAV$_2$} are set somewhere and, then remain static with regard to the ground truth coordinate frame {GT}. Each UAV is represented by the inertial coordinate frame that is attached to its body. The SLAM algorithm that is running on board the UAV$_a$ outputs map points and pose estimates with regard to {UAV$_a$}, in other words, the SLAM algorithm outputs either $^{\mathrm{UAV}_a}\mathbf{T}^{\mathrm{i}}_j$ (inertial-monocular) or $^{\mathrm{UAV}_a}\mathbf{T}^{\mathrm{c}}_j$ (pure monocular vision) regarding the sensors involved.

CHAPTER FOUR

INERTIAL-MONOCULAR FUSION FOR METRIC ESTIMATION

4.1 Introduction

In this thesis chapter, we present the loosely-coupled fusion scheme we designed to have metric estimates using a monocular SLAM algorithm and inertial measurements. This approach is used to estimate the scaling coefficient that transforms non-metric outputs such as the map and the poses estimated by the SLAM algorithm into metric estimates. We used the ORB-SLAM algorithm [MMT15] as the monocular SLAM algorithm and the EuRoC datasets [Bur+16] to perform the experiments. This work was published on the IEEE International Conference on Multi-Sensor Fusion and Integration for Intelligent Systems (MFI) [Spa+17].

4.2 A visual-inertial approach

Monocular vision has an inherent drawback, the scale of the world cannot be observed. It means that the 3D reconstruction of the environment is done up to a scale factor. Two scales have to be distinguished for the sake of comprehension: the reconstruction scale and the scaling coefficient. The reconstruction scale is estimated by the SLAM algorithm and makes the map consistent, i.e., big objects in real world are also bigger in the map built even if they were observed from

different camera poses. This scale is estimated by the SLAM algorithm, therefore a drift usually appear. However, the experiments we did using the EuRoC datasets showed that this drift is negligible in the ORB-SLAM system. The scaling coefficient is used to resize the map to have metric distances. In other words, the reconstruction scale provide estimates in arbitrary unit of distances while the scaling coefficient can be used to provide estimates in meters. It is impossible to compute a scaling coefficient in a monocular camera system [HZ03], at least another sensor has to be added. The second sensor is required to measure distances. Several sensors can meet the requirements such as lidars, ultrasounds or IMUs. The use of IMUs is often favored because of their small size, low cost and availability in every modern UAVs. However, the IMU does not measure distances directly but acceleration and angular velocities in the inertial frame. Obviously, the positions can be recovered by integrating the acceleration measurements but, consequently, the estimates drift very quickly with the error accumulation which prevent any long-term integration.

By basing our approach on distances, in contrast to map-based approaches, we can benefit from the research in inertial-visual odometry because the map is not required.

4.3 Overview of the approach and context

4.3.1 Our method for scale estimation

The scale ambiguity can be seen as a distance problem. The representation of distances outputted by the visual algorithm and distances in the real world, i.e., metric distances, are different. If world distances can be expressed in meters, the arbitrary distances computed by the monocular algorithm have no units. Nevertheless, monocular algorithm provide consistent measurements. Therefore, the scaling coefficient λ we are looking for is a scalar,

$$d_{\mathrm{w}} = \lambda d_{\mathrm{m}} \tag{4.1}$$

with d_w, the distances in the world (in meter) and d_m the distances from the monocular SLAM algorithm (arbitrary units of distance).

The distances can be represented by the L^2 norm of translation vectors $\|.\|_2$. Therefore, our reasoning is based on L^2 norms of translation vectors. We want to find the scaling coefficient λ such that

$$\|\mathbf{t}_w\|_2 = \lambda \|\mathbf{t}_m\|_2 \tag{4.2}$$

with $\mathbf{t}_w$ the metric translation vector (in meter) and $\mathbf{t}_m$ the non-metric translation vector from the monocular SLAM algorithm.

4.3.2 Related works

Two main approaches for monocular visual-inertial fusion can be distinguished in the literature: Loosely-coupled filtering [WS11] [HC14] [Sa+13] and tightly-coupled systems [MT17] [Con+16].

In tightly-coupled approaches, the fusion is done at a low level of the system. Therefore, this requires a deep understanding of the involved algorithms and specific design for the system.

The method described in [Con+16] is the inertial extension of the DPPTAM [CC15], a direct SLAM algorithm. The tracking thread is modified to include the IMU measurements. The Gauss-Newton optimization is used to minimize the intensity and IMU residuals. The state vector is composed of the position, orientation and velocity of the robot and the IMU biases. The IMU measurements are integrated between two consecutive keyframes. The IMU residuals are the error of the inertial integration between two keyframe with regard to the state value at the corresponding time. The intensity residuals are the photometric error between two keyframes. They are calculated by reprojecting the map points in the keyframes using the estimate of the relative camera pose. The optimization of both residuals provide the final pose estimate of the current keyframe with regard to the

world coordinate frame.

The method described in [MT17] is the inertial extension of ORB-SLAM [MMT15]. In ORB-SLAM, no functionality is provided to calculate the uncertainty of pose estimates. Therefore, the implemented method needs to avoid the direct use of the uncertainty of the camera pose. To represent information about uncertainty, the authors use information matrices computed either from the preintegration of the IMU measurements or from the feature extraction. The reprojection error and inertial error are minimized using the Gauss-Newton optimization. The reprojection error comes from the reprojection of map points in the current keyframe while the IMU error is derived from the preintegration equations described in [For+15].

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k} \Big(\mathrm{E}_{\mathrm{proj}}(k,j) + \mathrm{E}_{\mathrm{IMU}}(i,j) \Big) \tag{4.3}$$

where $\boldsymbol{\theta}$ the state vector which contains the position, orientation, velocity of the IMU at frame j in the world coordinate frame and the IMU biases, k a given match between a pair of features, i the last keyframe and j the current frame.

$$\mathrm{E}_{\mathrm{proj}}(j) = \rho\Big((\mathbf{x} - ß(\mathbf{X}_c))^T \boldsymbol{\Sigma}_x (\mathbf{x} - ß(\mathbf{X}_c)) \Big) \tag{4.4}$$

where $\mathbf{x}$ is the tracked keypoint, ß the projection function from the camera model, $\mathbf{X}_c$, the map point associated with the keypoint $\mathbf{x}$ in the camera coordinate frame. ρ is the Huber robust cost function.

$$\mathrm{E}_{\mathrm{IMU}}(i,j) = \rho(\begin{bmatrix} \mathbf{e}_R^T & \mathbf{e}_v^T & \mathbf{e}_p^T \end{bmatrix} \boldsymbol{\Sigma}_I \begin{bmatrix} \mathbf{e}_R^T & \mathbf{e}_v^T & \mathbf{e}_p^T \end{bmatrix}^T) + \rho(\mathbf{e}_b^T \boldsymbol{\Sigma}_R \mathbf{e}_b) \tag{4.5}$$

where $\mathbf{e}_R$, $\mathbf{e}_v$, $\mathbf{e}_p$ comes from the IMU preintegration equations, they represent the IMU position, orientation and velocity errors computed from the integration of the IMU measurements. $\mathbf{e}_b$ is the bias error computed from the IMU bias estimation. The $\boldsymbol{\Sigma}$ variables are information matrices. The value of $\boldsymbol{\Sigma}_x$ comes from the feature extraction, $\boldsymbol{\Sigma}_I$ is given by the calculation of the preintegration equations and $\boldsymbol{\Sigma}_R$

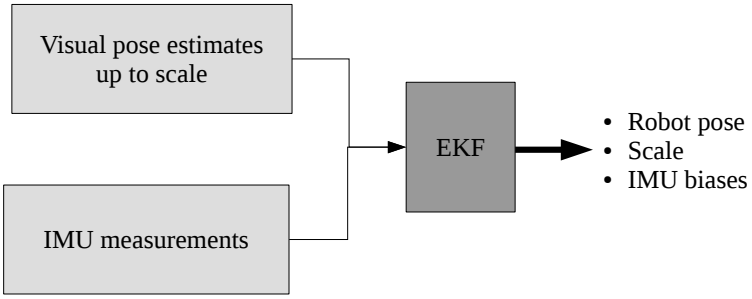


Figure 4.1: Architecture of a visual-inertial fusion using EKF.

represent the variance of the bias random walk. If the map is not updated, a prior knowledge term is added to the minimization. Note that IMU measurements are also used for the local mapping because the velocity and bias states are included into the Local Bundle Adjustment optimization of the local mapping thread. [MT17] also provides a novel technique for IMU initialization (scale, gravity vector, velocity and IMU biases) using the first ORB-SLAM estimates. The code of this approach is not available and the complexity of the ORB-SLAM processing code makes a personal implementation hardly conceivable. Moreover, the value of $\mathbf{\Sigma}_x$ is questionable. The matrix should be the inverse of the Hessian matrix (second derivative of the projection error). However, here the projection error is not derived, instead the variance associated to the feature extraction compose the value of the matrix $\mathbf{\Sigma}_x$.

In contrast to tightly-coupled approaches, in loosely-coupled approaches, the vision part is considered as a black box, only the output of the box is used. In most loosely-coupled algorithms such as [WS11] [HC14] [Sa+13], the filter, which fuses the measurements, is derived from Kalman Filtering, e.g., Extended Kalman Filter or Multi-State Constraints Kalman Filter as illustrated in Figure 4.1. The state is, at least, composed of the position, orientation, velocity and biases of the IMU. The differential equations, which govern the system and the IMU measurements, are used to predict the state. The incorporation of monocular visual measurements is done through the measurement model when the Kalman gain needs to be computed (the visual measurements update the state when the innovation is calculated).

In [WS11], the state vector additionally includes the calibration states (the constant relative position and orientation between the IMU coordinate frame and the

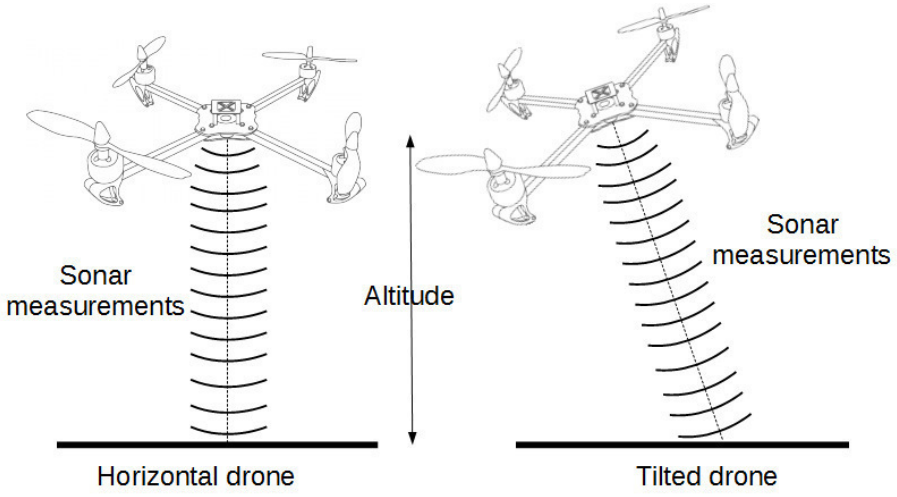


Figure 4.2: Schema of sonar measurements on UAVs from [Mor78].

camera coordinate frame) and a failure detection system. When a failure is detected (abrupt changes in the orientation estimates with regard to the measurement rate), the related visual measurements are automatically discarded to prevent the corruption of data.

In [HC14], the authors use trifocal tensor geometry which considers epipolar constraints in triples of consecutive images instead of pairs of images. Therefore, in addition to the usual IMU states, the state vector also contains the pose and orientation of the two previous keyframes.

In [Sa+13], the fusion is done by using measurements from three sensors: In addition to the visual and inertial sensors, a sonar is included in the system to measure distances (the altitude between the UAV and the ground). IMU is used to detect whether the UAV is flying level or tilted. If it is level, the sonar measurements are directly used to estimate the altitude. Otherwise, IMU measurements help to rectify the incorrect altitude information due to the tilting of the UAV. Figure 4.2 provides an overview of the use of the sonar measurements for altitude estimation and, therefore, metric scale estimation.

The scale factor estimation is represented as an optimization problem between the sonar and visual altitude measurements which is solved using the Levenberg-

Marquardt algorithm [Mor78].

$$\underset{\lambda}{\operatorname{argmin}} \sum_{k=1}^{T} \|\Delta z_k^{\text{sonar}} - \lambda \Delta z_k^{\text{camera}}\|_2^2 \tag{4.6}$$

With $\Delta z_k^{\text{sonar}}$ and $\Delta z_k^{\text{camera}}$ the changes in altitude computed by the sonar and the camera, λ is the unknown scale factor due to monocular vision.

We propose an approach for fusing monocular and inertial measurements in a loosely-coupled manner which is simple to implement, requires small computational resources and so, is suitable for UAVs. We decided to design a loosely-coupled approach to make the methods used for visual tracking and IMU measurement integration easy to replace with any other method ones may prefer, which ensures better flexibility and usability for our approach. In our studies, we used the ORB-SLAM algorithm [MMT15] for the visual tracking part and Euler forward integration for the inertial measurements processing.

4.4 Scaling coefficient estimation with IMU measurements

4.4.1 Coordinate Frames

Our system is composed of two sensors, a monocular camera and an IMU, attached on a rigid flying body, the single UAV. As described in Section 3, we distinguish four coordinate frames: camera $\{CAM\}$, vision $\{VIS\}$, inertial $\{IMU\}$ and world $\{W\}$ coordinate frames. The world coordinate frame $\{W\}$ corresponds to the UAV coordinate frame as defined in Section 3.4.3. The IMU measures data in $\{IMU\}$ attached to the body of the UAV. The integration of IMU measurements results in the estimation of the pose of the IMU in $\{W\}$. The monocular pose estimation algorithm outputs the camera poses in $\{VIS\}$, which corresponds to the

first $\{CAM\}$ coordinate frame when the tracking starts, i.e.

$$\{VIS\} \triangleq \{CAM\}_{\mathrm{t}=0} \tag{4.7}$$

The matrix ${}^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}$, which represents the transformation between $\{CAM\}$ and $\{IMU\}$, is constant, and can be computed off-line using a calibration method [MR08]. In the EuRoC dataset sequences that we used for our experiments, ${}^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}$ is already provided. We consider that $\{W\}$ corresponds to $\{IMU\}$ at the moment tracking starts, when the $\{VIS\}$ coordinate frame is generated, so

$$\{W\} \triangleq \{IMU\}_{\mathrm{t}=0} \tag{4.8}$$

In the following paragraphs, we consider that the monocular pose estimation algorithm outputs measurements in the world coordinate frame by applying the formula

$${}^{\mathrm{w}}\mathbf{p} = {}^{\mathrm{i}}\mathbf{T}_{\mathrm{c}}{}^{\mathrm{v}}\mathbf{p} \tag{4.9}$$

where $\mathbf{p}$ is a 3-D measurement by the monocular pose estimation algorithm.

4.4.2 Integration of IMU Measurements

We assume having a monocular pose estimation algorithm which outputs consistent camera poses in the world coordinate frame. As a monocular camera cannot be used to calculate metric distances, we use the IMU measurements to compute the scaling coefficient. IMU is a sensor composed of three accelerometers and three gyroscopes which respectively measure accelerations and angular velocities along each of the three axis of the inertial frame. The pose of the IMU can be estimated by integrating the accelerometer and gyroscope measurements. However, the integration of the IMU measurement noise and the IMU biases makes the pose estimates to drift fast. Therefore, IMU measurements must be integrated only over a short period for limiting the drift and consequently to corrupt the estimates. We integrate IMU

Figure 4.3: IMU measurements obtained between the two subsequent image frames are used to calculate the translation vectors and rotation matrices.

measurements between two consecutive image frames. So, if F_k and F_{k+1} are two coordinate frames associated with image frames k and $k+1$, as displayed in Figure 4.3, the integration results in the estimation of the rotation matrix ${}^{F_k}\mathbf{R}_{F_{k+1}}$, velocity ${}^{\mathrm{w}}\mathbf{v}_{F_k,F_{k+1}}$ and translation ${}^{\mathrm{w}}\mathbf{t}_{F_k,F_{k+1}}$ vectors between the two consecutive frames F_k and F_{k+1} in the world coordinate frame using IMU measurements.

We integrated the IMU measurements using Euler forward integration following the description given in [Con+16]

$$ {}^{F_i}\mathbf{R}_{F_j} = \prod_{p=k}^{k+N-1} \exp_{\mathrm{SO(3)}}([\boldsymbol{\omega}(p) + \mathbf{b}_\omega(p)]^\wedge \Delta T) \tag{4.10} $$

where $\boldsymbol{\omega}$ is the vector of gyroscope measurements, $\mathbf{b}_\omega$ the gyroscope bias, k the time step of frame F_i, $k+N$ the time step of frame F_j, $N-1$ the number of IMU measurements between the consecutive frames F_i and F_j and ΔT is the time step size (in seconds)

$$ {}^{\mathrm{w}}\mathbf{v}_{F_i,F_j} = \sum_{p=k}^{k+N-1} [{}^{\mathrm{w}}\mathbf{R}_{I_p}(\mathbf{a}(p) + \mathbf{b}_a(p)) - \mathbf{g}]\Delta T \tag{4.11} $$

where ${}^{\mathrm{w}}\mathbf{R}_{I_p}$ is the rotation matrix between the world coordinate frame and the IMU coordinate frame at time p, $\mathbf{a}$ is the vector of accelerometer measurements, $\mathbf{b}_a$ the accelerometer bias and g the gravity vector

$$\begin{aligned} {}^{\mathrm{w}}\mathbf{t}_{F_i,F_j} =& N{}^{\mathrm{w}}\mathbf{v}_{F_i}\Delta T + \\ & \frac{1}{2}\sum_{p=k}^{k+N-1}\Big[\left(2\left(k+N-1-p\right)+1\right) \\ & \left(w_p^{\mathbf{R}}\left(\mathbf{a}(p)+\mathbf{b}_a(p)\right)-\mathbf{g}\right)\Big]\Delta T^2 \end{aligned} \tag{4.12}$$

The $\exp_{SO(3)}$ operator maps an vector of so(3) to a matrix of SO(3). The wedge operator $.\wedge$ convert a 3×1 vector into an element of so(3), i.e., a skew-symmetric matrix of size 3×3. The IMU biases, $\mathbf{b}_a$ and $\mathbf{b}_\omega$, were modeled as a random walk process

$$\mathbf{b}_a(k+1) = \mathbf{b}_a(k) + \Delta T \boldsymbol{\sigma}_a^2 \tag{4.13}$$

where $\boldsymbol{\sigma}_a^2$ is the variance associated to the IMU accelerometers

$$\mathbf{b}_\omega(k+1) = \mathbf{b}_\omega(k) + \Delta T \boldsymbol{\sigma}_\omega^2 \tag{4.14}$$

where $\boldsymbol{\sigma}_\omega^2$ is the variance associated to the IMU accelerometers

Note that the IMU integration equations (Equations 4.10,4.11, 4.12) can be replaced with another approach for numerical integration (such as [For+15]) as long as this calculates the translation vector between two consecutive frames in the world coordinate frame $\{W\}$.

It is assumed that the estimates provided by the monocular pose estimation algorithm $\mathbf{x}^{\mathrm{m}}$ drift slower than the estimates $\mathbf{x}^{\mathrm{i}}$ computed using the IMU measurements, i.e., $\mathbf{x}^{\mathrm{m}}$ is more accurate than $\mathbf{x}^{\mathrm{i}}$. At each incoming frame, we used the value of $\mathbf{x}^{\mathrm{m}}$ to initialize $\mathbf{x}^{\mathrm{i}}$. We observed that a good initialization for the IMU estimates can greatly improve the accuracy of the estimation of the scaling coefficient. However, the initialization of the estimates $\mathbf{x}^{\mathrm{i}}$ is not discussed in this thesis and is part of the further improvement we plan to do.

We also benefit from the high measurement rate of the IMU (between 100 Hz and 200 Hz) to provide fast pose estimates. The pose estimates from the vision algorithm $\mathbf{x}_{\mathrm{m}}$ can be updated once per new frame at maximum. Therefore, the camera pose is updated at the frame rate, usually around 30 Hz, which can be too slow for some applications such as control or navigation.

4.4.3 Calculation of Scaling Coefficient

We want to find the scaling coefficient λ as follows

$$\|{}^{W}\mathbf{t}^{\mathrm{i}}\|_2 = \lambda \|{}^{W}\mathbf{t}^{\mathrm{m}}\|_2 \tag{4.15}$$

where ${}^{W}\mathbf{t}^{\mathrm{i}}$ are the translation vectors of the camera position given by the integration of IMU measurements in the world coordinate frame, ${}^{W}\mathbf{t}^{\mathrm{m}}$ are the translation vectors of the camera position given by the monocular odometry or SLAM algorithm in the world coordinate frame.

For each incoming new frame F_j, the translation of the camera between the consecutive frames F_i and F_j in the world coordinate frame given by the monocular vision algorithm ${}^{W}\mathbf{t}^{\mathrm{m}}_{F_i,F_j}$ is measured. We then integrate the corresponding IMU measurements using the Eq. (1), (2) and (3) to obtain the corresponding translation from the inertial measurements ${}^{W}\mathbf{t}^{\mathrm{i}}_{F_i,F_j}$

$$\lambda_{F_i,F_j} = \frac{\|{}^{W}\mathbf{t}^{\mathrm{i}}_{F_i,F_j}\|_2}{\|{}^{W}\mathbf{t}^{\mathrm{m}}_{F_i,F_j}\|_2} \tag{4.16}$$

So Eq. 4.16 provides an estimated value of the scaling coefficient λ.

We can measure λ for each frame, but the measurement noise on each measurement is significant because both the outputs of the SLAM algorithm and the IMU integration drift. Four methods have been tested to calculate the scaling coefficient $\hat{\lambda}$ using the measurements λ_{F_i,F_j}. The four methods are: moving average on λ_{F_i,F_j} with an additive model for the error, moving average on $\log(\lambda_{F_i,F_j})$ with a

multiplicative model for the error, an autoregressive Filter and a Kalman Filter.

The moving averages are calculated over the available measurements at time t

$$\hat{\lambda}_1 = \frac{1}{M} \sum_{k=2}^{M-1} \left(\frac{\| {}^{W}\mathbf{t}^{\mathrm{i}}_{F_k,F_{k+1}} \|_2}{\| {}^{W}\mathbf{t}^{\mathrm{m}}_{F_k,F_{k+1}} \|_2} \right) \tag{4.17}$$

where the error model is additive

$$\hat{\lambda}_2 = \exp\left(\frac{1}{M} \sum_{k=2}^{M-1} \log\left(\frac{\| {}^{W}\mathbf{t}^{\mathrm{i}}_{F_k,F_{k+1}} \|_2}{\| {}^{W}\mathbf{t}^{\mathrm{m}}_{F_k,F_{k+1}} \|_2} \right) \right) \tag{4.18}$$

where the error model is multiplicative and M is the number of frames at the considered discrete time t. The first frame is skipped because the error on the first IMU measurement is generally large.

We also decided to implement an autoregressive filter (AR) to estimate $\hat{\lambda}$. Moreover, this filter can be used to check whether the measurements are correlated in time. The current value of the filter output $y(i)$ is a weighted linear combination of the p previous outputs (p is the order of the filter) and the current measurement. The weights are computed by solving the Yule-Walker equations

$$y(i) = K + s(i) + \sum_{j=1}^{p} \alpha_i y(i-j) \tag{4.19}$$

where $y(i)$ is the output of the AR filter at discrete time i, α_i are the weights calculated with the Yule-Walker equations, s is a zero-mean random variable with

$$s(i) = \lambda(i) - K \tag{4.20}$$

$$\lambda(i) = \frac{\| {}^{W}\mathbf{t}^{\mathrm{i}}_{F_i,F_{i+1}} \|_2}{\| {}^{W}\mathbf{t}^{\mathrm{m}}_{F_i,F_{i+1}} \|_2} \tag{4.21}$$

The weights α_i and bias term K are calculated solving

$$\begin{bmatrix} \mu \\ c_1 \\ c_2 \\ \vdots \\ c_p \end{bmatrix} = \begin{bmatrix} 1 & \mu & \mu & \dots & \mu \\ \mu & c_0 & c_1 & \dots & c_{p-1} \\ \mu & c_1 & c_0 & \dots & c_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mu & c_{p-1} & c_{p-2} & \dots & c_0 \end{bmatrix} \begin{bmatrix} K \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} \tag{4.22}$$

where c_p is the cross-correlation of the signal y with temporal lag p and μ is the average of y at time i.

The AR filter diverged and it led to poor accuracy for the estimate $\hat{\lambda}$ in all EuRoC sequences. These results show that the measurements are not temporally correlated and do not follow an autoregressive model.

Finally, a Kalman Filter has been implemented to estimate $\hat{\lambda}$. The model is

$$\lambda_k = a\lambda_{k-1} + w_k \tag{4.23}$$

$$z_k = h\lambda_{k-1} + v_k \tag{4.24}$$

where $a = 1$ and $h = 1$.

The prediction step is done using

$$\hat{\lambda}_{k|k-1} = a\hat{\lambda}_{k-1|k-1} \tag{4.25}$$

and the a priori variance $p_{k|k-1}$

$$p_{k|k-1} = a^2 p_{k-1|k-1} + q \tag{4.26}$$

where q is the covariance of the model white noise w.

The correction step is calculated as follows

$$k_{k|k} = \frac{hp_{k|k-1}}{h^2 p_{k|k-1} + r} \tag{4.27}$$

where k is the Kalman gain and r is the covariance of the measurement white noise v

$$\hat{\lambda}_{k|k} = \hat{\lambda}_{k|k-1} + k_{k|k}(z_k - h\hat{\lambda}_{k|k-1}) \tag{4.28}$$

and the a posteriori variance $p_{k|k}$

$$p_{k|k} = p_{k|k-1}(1 - hk_{k|k}) \tag{4.29}$$

4.4.4 Experimental results

We experimented the proposed method using the sequences from EuRoC dataset [Bur+16] and the ORB-SLAM algorithm [MMT15]. Table 4.1 displays the output of our experimentations. The EuRoC dataset provides eleven sequences recorded by an Asctec Firefly hex-rotor helicopter in two different environments, a room equipped with a Vicon motion capture system and a machine hall. We used the frames from one of the front stereo camera (Aptina MT9V034 global shutter, WVGA monochrome, 20 FPS) to emulate monocular vision and the measurements of the MEMS IMU (ADIS16448, angular rate and acceleration, 200 Hz). The ground truth is measured either by the Vicon motion capture system in the sequences recorded in the Vicon room, or by a Leica MS50 laser tracker and scanner in the machine hall environment.

In the following, the sequences referred as V1_01, V1_02 and V1_03 were recorded in the Vicon room with configuration of texture 1; the sequences referred as V2_01, V2_02 and V2_03 were recorded in the Vicon room with configuration of texture 2; the sequences referred as MH01, MH02, MH03, MH04 and MH05 were recorded in the machine hall using the Leica system. Note that the trajectory of the UAV is different in each sequence.

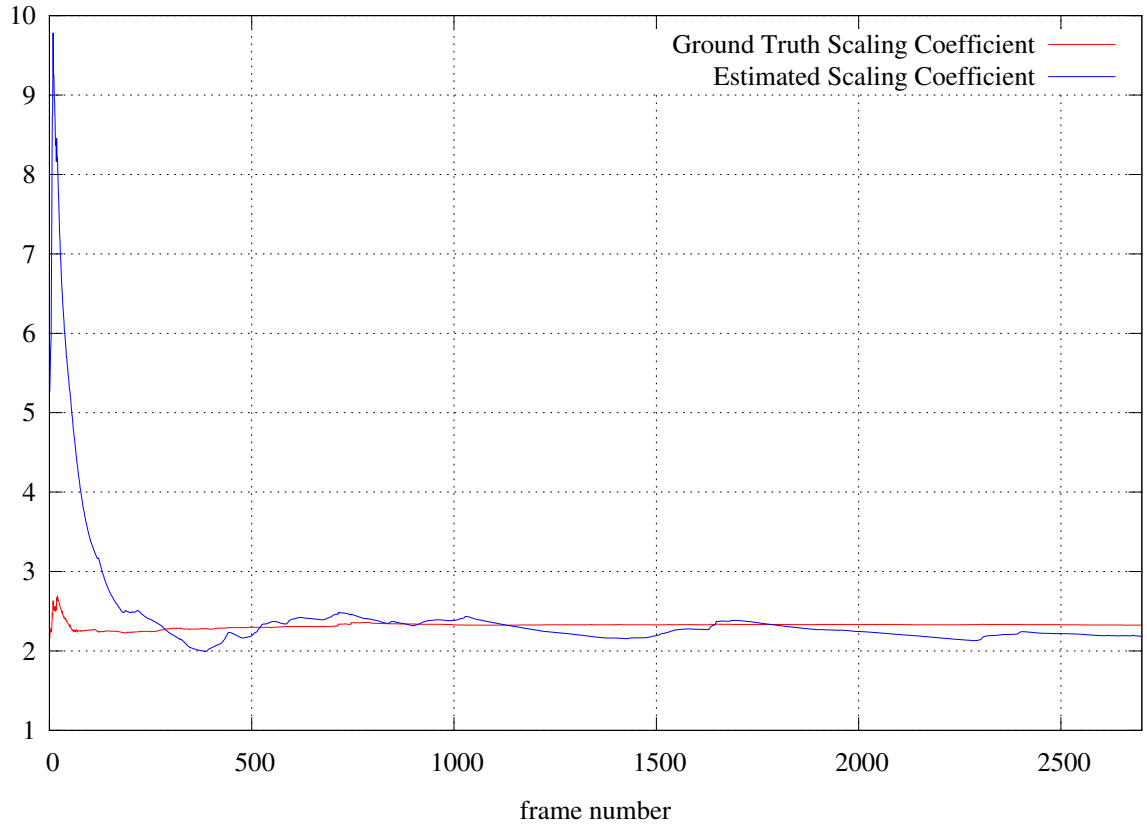


Figure 4.4: The estimation of the scaling coefficient λ using the sequence V1_01 from EuRoC dataset (In blue the ground truth calculated from the Vicon estimates, in red the proposed estimation method).

The estimation of the scaling coefficient during the sequence V1_01 is pictured in Figure 4.4. The value of the scaling coefficient can be compared to a ground truth value, which is computed using a moving average with Vicon (or Leica) measurements ${}^{W}\mathbf{t}^{\mathrm{GT}}$ instead of IMU measurements

$$\lambda^{\mathrm{GT}} = \frac{1}{M} \sum_{k=1}^{M-1} \left(\frac{\| {}^{W}\mathbf{t}^{\mathrm{GT}}_{F_k,F_{k+1}} \|_2}{\| {}^{W}\mathbf{t}^{\mathrm{m}}_{F_k,F_{k+1}} \|_2} \right) \tag{4.30}$$

The error e_λ between the ground truth scaling coefficient λ^{GT} and the coefficient we estimate using inertial measurement $\hat{\lambda}$ is calculated as follows

$$e_\lambda = \| \lambda^{\mathrm{GT}} - \hat{\lambda} \|_1 \tag{4.31}$$

where $\|.\|_1$ is the L^1 norm.

We rescaled the trajectory provided by the monocular algorithm. As presented

Table 4.1: Scaling coefficient and effect on the trajectory.

EuRoC Sequence	$\hat{\lambda}$			λ^{GT}	e_λ			RMSE (m)			Total distance (m)	
	$\hat{\lambda}_1$	$\hat{\lambda}_2$	KF		$\hat{\lambda}_1$	$\hat{\lambda}_2$	KF	$\hat{\lambda}_1$	$\hat{\lambda}_2$	KF	Ground truth	Best estimate
V1_01	2.49	1.89	12.42	2.31	0.19	0.42	10.11	**0.22**	0.51	12.25	59.26	63.91
V1_02	1.80	1.41	7.57	2.35	0.55	0.94	5.22	**0.55**	0.95	5.25	76.58	58.74
V1_03	2.55	1.92	6.73	3.54	1.00	1.63	3.19	**0.46**	0.75	1.47	77.53	55.78
V2_01	2.92	2.33	1.17	3.02	0.10	0.70	1.86	**0.09**	0.60	1.67	36.93	35.66
V2_02	4.32	1.87	82.15	3.56	0.77	1.684	78.80	**0.47**	1.03	47.99	83.92	101.93
V2_03	1.94	1.42	1.26	2.15	1.84	2.36	2.52	**1.66**	2.13	2.27	139.15	71.30
MH01	251.58	7.57	10.10	6.92	208.66	0.65	3.18	216.55	**0.68**	3.30	85.50	93.54
MH02	51.91	2.64	1.08	2.94	48.97	0.30	1.86	146.69	**0.90**	5.56	84.39	75.88
MH03	35.81	2.31	4.93	3.77	32.04	1.46	1.16	39.85	1.82	**1.45**	131.13	171.45
MH04	36.74	3.78	9.28	7.51	29.23	3.74	1.17	32.78	4.19	**1.98**	103.58	127.94
MH05	33.21	2.05	2.27	2.74	30.47	0.69	0.47	109.16	2.48	**1.68**	116.02	96.22

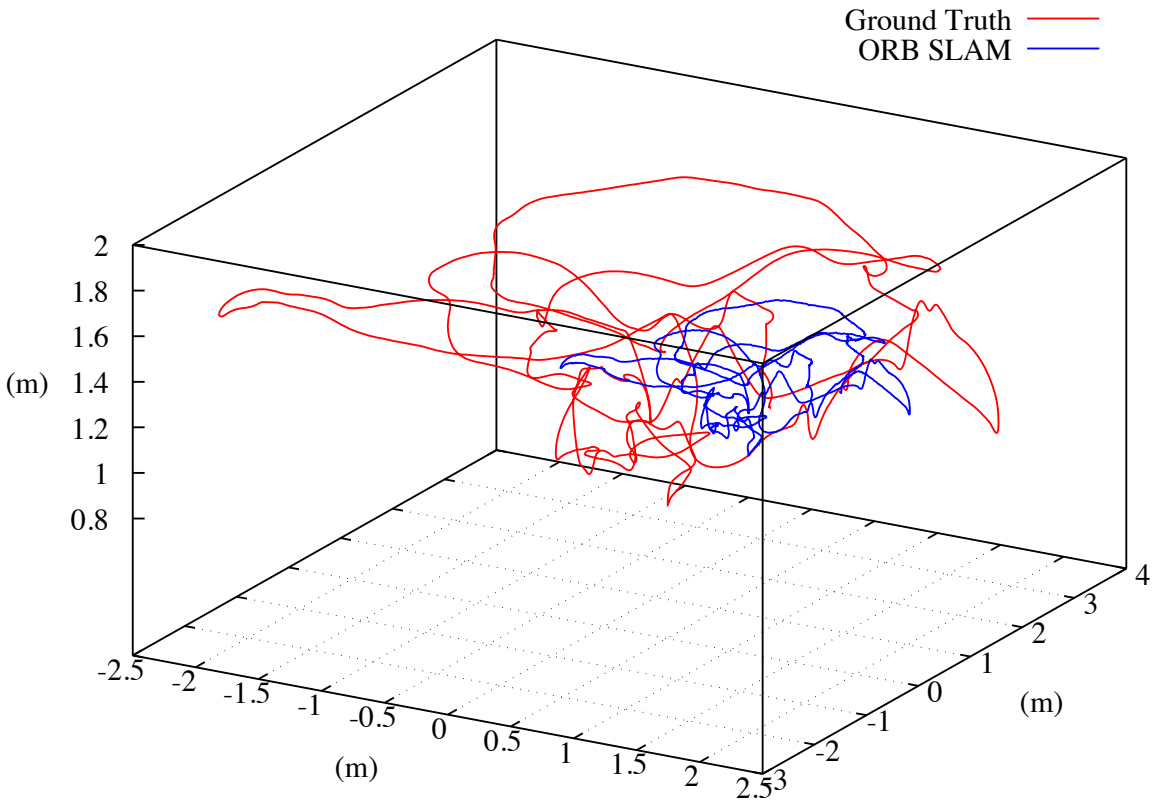


Figure 4.5: The trajectory of the camera during the V1_01 sequence from EuRoC dataset in the world coordinate frame (ORB-SLAM measurements are in blue, the ground truth measured with a Vicon motion capture system are in red).

in Figure 4.5, the distances given by the monocular algorithm are arbitrary but consistent, therefore the UAV's trajectory is scaled differently than the ground truth. In Figure 4.6, we rescaled the monocular trajectory of the sequence V1_01 using the ground truth λ^{GT} and estimated $\hat{\lambda}_1$ scaling coefficients. We computed the root-mean-square deviation (RMSE) for each sequence as follows

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{M} \left(\lambda^{\mathrm{GT}}\mathbf{x}_i^T - \hat{\lambda}\mathbf{x}_i^T\right)\left(\lambda^{\mathrm{GT}}\mathbf{x}_i - \hat{\lambda}\mathbf{x}_i\right)}{M}} \tag{4.32}$$

where M is the number of frames in the sequence and $\mathbf{x}$ is the position of the camera in the world coordinate given by the monocular algorithm (ORB-SLAM for our experiments). The RMSE for each EuRoC sequence is given in Table 4.1.

The initial trajectory of the UAVs in the sequence V1_01 provided by the ORB-SLAM algorithm is displayed in Figure 4.5. The same trajectory rescaled using the scaling coefficients λ^{GT} and $\hat{\lambda}_1$ is displayed in Figure 4.6.

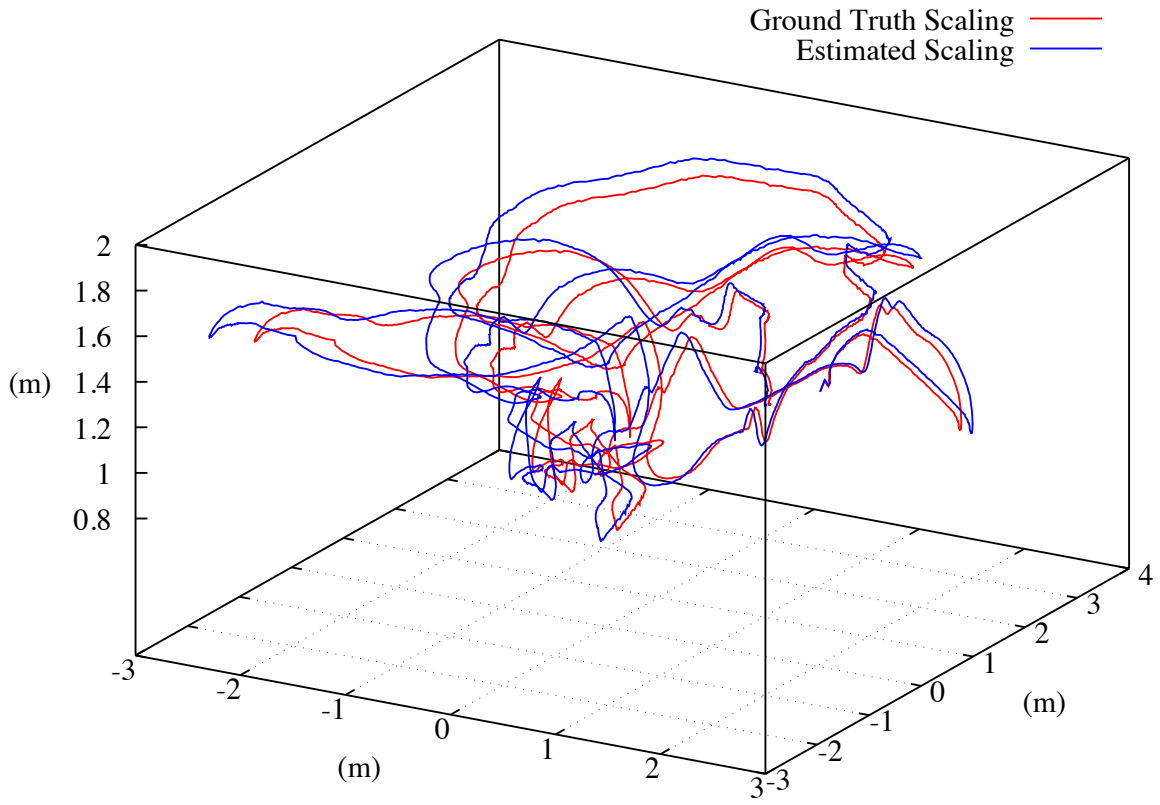


Figure 4.6: The rescaled trajectory of the camera provided by ORB-SLAM in the V1_01 sequence from EuRoC dataset in the world coordinate frame (the blue trajectory was rescaled using the value of λ, the red trajectory was rescaled using the ground truth scaling coefficient).

As expected, the bigger the scaling coefficient error e_λ, the bigger the RMSE. The sequences recorded in the Vicon room provide trajectories with lower RMSE than the sequences recorded in the machine hall. The difference between the two sets of sequences can be explained by the strong excitation of the IMU for the calibration of the Leica laser that incorporates a lot of noise in the measurements $\mathbf{t}^i$. In the Vicon sequences, the estimate $\hat{\lambda}_1$ outperforms. The sequences recorded in the machine hall are very challenging for the inertial fusion because the UAV performed very fast translational movements during a few seconds for Leica ground truth calibration purposes which result in large acceleration measurements and partially corrupt the inertial estimates as shown in Figure 4.7. Therefore, in the machine hall sequences where strong noise corrupts some IMU measurements, $\hat{\lambda}_2$ is far better than $\hat{\lambda}_1$, which never managed to completely absorb the strong perturbations of the calibration. Interestingly, Kalman Filter (KF) gives also satisfactory results for Leica

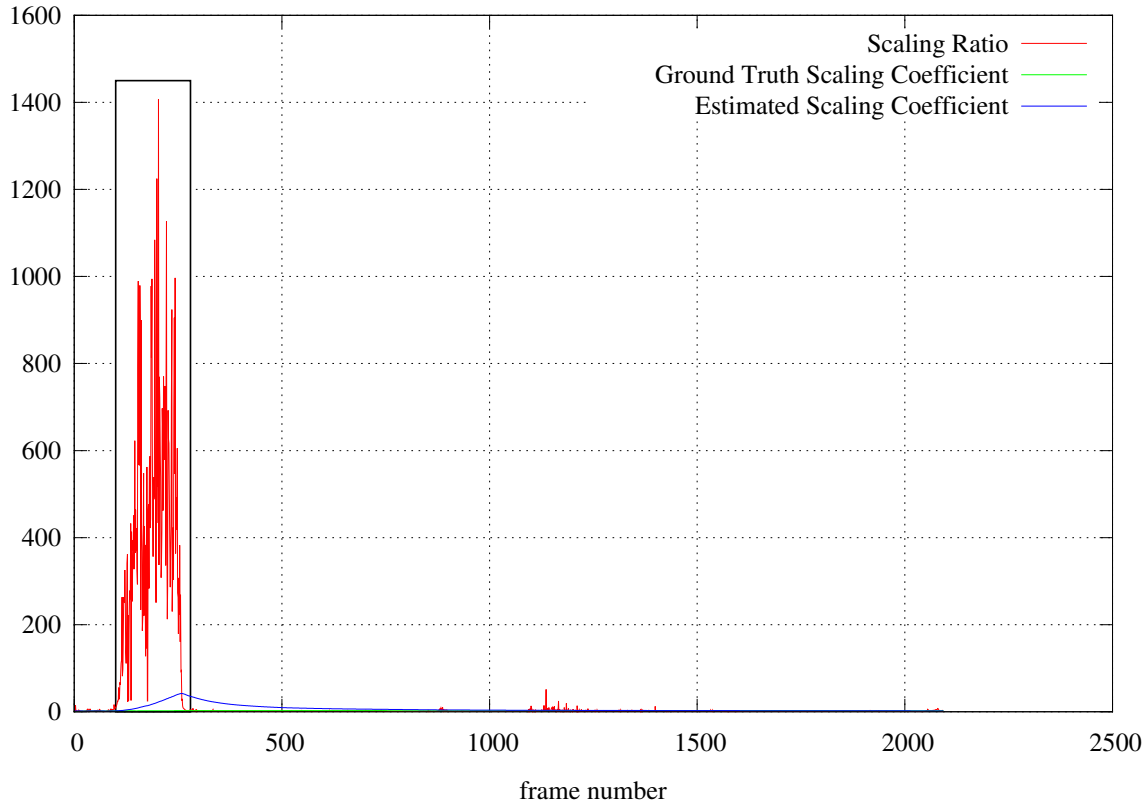


Figure 4.7: An example of the corruption of estimates due to the fast translational movements (shown in the rectangular area as marked) for the Leica calibration in one of the sequences recorded in the machine hall environment (MH05).

sequences. The results of Kalman Filter can be further improved with a finer tuning of the process noise variance q. For instance, with a smaller value for q, the RMSE of sequences MH_03 and MH_04 drops to 0.17 m and 0.76 m respectively. As every single EuRoC sequence is quite different from the others, finding a nice tuning value for the Kalman Filter is not straightforward. We recommend to tune the filter accordingly to the type of flight the UAV performs (smoothness and aggressiveness of trajectories, motion speed, angular velocities). If a Kalman Filter cannot be implemented or tuned, $\hat{\lambda}_2$ remains a acceptable estimate. More broadly, keeping the IMU out from large perturbations by using smooth trajectories provides more accurate estimates. The estimates computed through the AR filter, which are not presented because of the dramatically large value of RMSE, show that there is no temporal correlation of the error.

4.4.5 Concluding Remarks and Future Work

We presented a fast and easy-to-implement method for the calculation of the scaling coefficient by fusing inertial measurements with monocular pose estimation. Monocular camera systems, due to their nature, can not provide the real-world scale of the pose estimates. To overcome this problem, we use the inertial measurements produced by an IMU to i) estimate the scaling coefficient, which relates the monocular camera pose estimation to the real-world scale, and ii) speed up the pose estimation by exploiting the availability of the inertial measurements in very high rates.

The method is highly modular, which makes each component to be easily replaceable with one's preferences without impacting the overall operation of the system.

To improve the current method, we plan to further investigate the initialization of the IMU integration process, particularly with the incorporation of the current estimate of the scaling coefficient when appropriate.

We defined three approaches for calculating the scaling coefficient with regard to the nature of the trajectory followed by the UAV. We found that the Kalman Filter approach gives accurate estimates when the tuning is done well, which unfortunately can be hard to do for some applications. Determination of the tuning value of the process noise is a complex topic which will be part of our future research work and experimentations.

CHAPTER FIVE

MULTI-UAV SLAM: PLACE RECOGNITION, FEATURE MATCHING AND ESTIMATION OF THE POSE-GRAPH CONSTRAINTS

5.1 Introduction

In the following chapters, we consider that we have a system capable of outputting metric SLAM estimates whether using a loosely-coupled or a tightly-coupled fusion scheme. In this chapter, we describe the front-end of the M-SLAM system: its design, architecture and how it is related to the single UAV metric SLAM on one hand (the topics related to single UAV metric SLAM were discussed in Chapters 2 and 4), and to the back-end process on the other hand (Chapter 6 presents the back-end part in details).

5.2 Related work

In this chapter, as well as in Chapters 6 and 7, we present the M-SLAM system, an inertial-monocular SLAM system for a fleet of UAVs. Few literature can be found on a similar system that uses only monocular vision, inertial measurements running on a UAV platform. We discussed some approaches in Section 2.2.4. One of the closest approach to the applications we target is the CVI-SLAM algorithm

[KSC18] published recently in 2018. The CVI-SLAM algorithm can be seen as the inertial extension of the CCM-SLAM algorithm we presented in Section 2.2.4. The architecture of CVI-SLAM is similar to the CCM-SLAM (a centralized collaborative SLAM for multi-UAV systems [SC18]), the focus is set on the map merging of local map estimated by the UAVs on a distant server. A particular attention is devoted to the communication part in order to ensure each UAV can remains autonomous and working even in case of data loss or network failure. The main novelty with regard to CCM-SLAM is the inertial-visual odometry algorithm that is run on the front-end instead of vision-only odometry. The inertial measurements are pre-integrated and a nonlinear optimization process minimizes both the visual and inertial residuals in order to fuse both type of measurements. This tightly-coupled scheme (similar to the one used in VINS-Mono) allows to get metric estimates for the UAV pose, and consequently, metric local maps.

5.3 Objective

The M-SLAM is composed of a front-end and a back-end. The front-end manages the input data and process the sensor measurements. The back-end generates the SLAM pose-graph and improves the localization of the fleet by fusing the different location information. In other words, the front-end is a perception component while the back-end is an data fusion component. The role of the front-end is threefold:

1. Gather the data from every UAV of the fleet,

2. Detect coincidence candidates and filter false coincidence detections,

3. Process the keyframes, the features and the map points in order to estimate a SE(3) constraint for each coincidence.

5.4 Special Euclidean group and UAV coincidences: definitions

5.4.1 The SE(3) transformation

In this report, we note as SE(3) and SO(3) direct isometries that are a subgroup of the Euclidean group (also called special Euclidean group). We use the SE(3) and SO(3) groups to describe the positions or movements of the rigid body that is the UAV frame.

SO(3) transforms are used to represent 3-D rotation that can be parameterized either using Euler angles (yaw, pitch, roll), 3-by-3 rotation matrices, or quaternions. Other representations can be found in the literature, particularly in the work about Lie algebra.

SE(3) transforms add a 3-D translation with regard to SO(3) transforms. We use the SE(3) transforms to represent the transformations to change from a coordinate frame to another one or to represent the rigid body (the UAV frame) trajectory in the 3-D environment.

5.4.2 Definition of coincidences

We define a coincidence as being the fact for one or two UAVs to visit the same place. We consider the traditional loop-closure in single-robot SLAM as a particular case of robot coincidence. An intra-coincidence occurs if the same UAV visits at least two times the same area non consecutively; an inter-coincidence occurs if two different UAVs visit the same area. Coincidences are defined spatially, inter-coincidences can be trigger even though the UAVs are not visiting the area at the same time. The data required to define a coincidence are the following:

- The two UAV identifiers,
- The two moment identifiers (it can be timestamps or keyframe numbers),

- The poses of the UAVs in their coordinate frame ${}^{UAV_i}\mathbf{T}_j$, and

- Either the two keyframes (images) or the list of features (keypoints and descriptors) that belongs to the keyframes if they are numerous enough (otherwise we must be able to extract additional features later in the process).

5.5 Architecture of the front-end

Each UAV of the fleet runs its own SLAM algorithm, the output of this single-robot SLAM (as well as useful pieces of information like the camera keyframes) are used as an input for the M-SLAM front-end. As previously discussed, the choices for the SLAM algorithm are very limited. We identified, through our extensive search, only two monocular SLAM algorithms with a tracking part robust enough to be run on a UAV and available as open source: ORB-SLAM [MMT15] and VINS-Mono [QLS17]. The main benefit of VINS-Mono over ORB-SLAM is that it is a tightly-coupled inertial-monocular SLAM algorithm, therefore the estimates are metric. This is a significant advantage as if we work with a non metric algorithm, we will have to compute the scaling coefficient between each pair of UAVs which makes the research problem more complex. Moreover, with regard to the DIVINA challenge team as described in Subsection 1.3.2, metric estimates suit better the whole context. One approach consists in using directly the outputs of a tightly-coupled inertial-monocular SLAM such as VINS-Mono as shown in Figure 5.1. Another approach is to use a loosely-coupled fusion between a monocular SLAM algorithm and inertial measurements to make metric the estimates such as the approach described in Chapter 4 that we experimented using the ORB-SLAM algorithm [Spa+17] as shown in Figure 5.2.

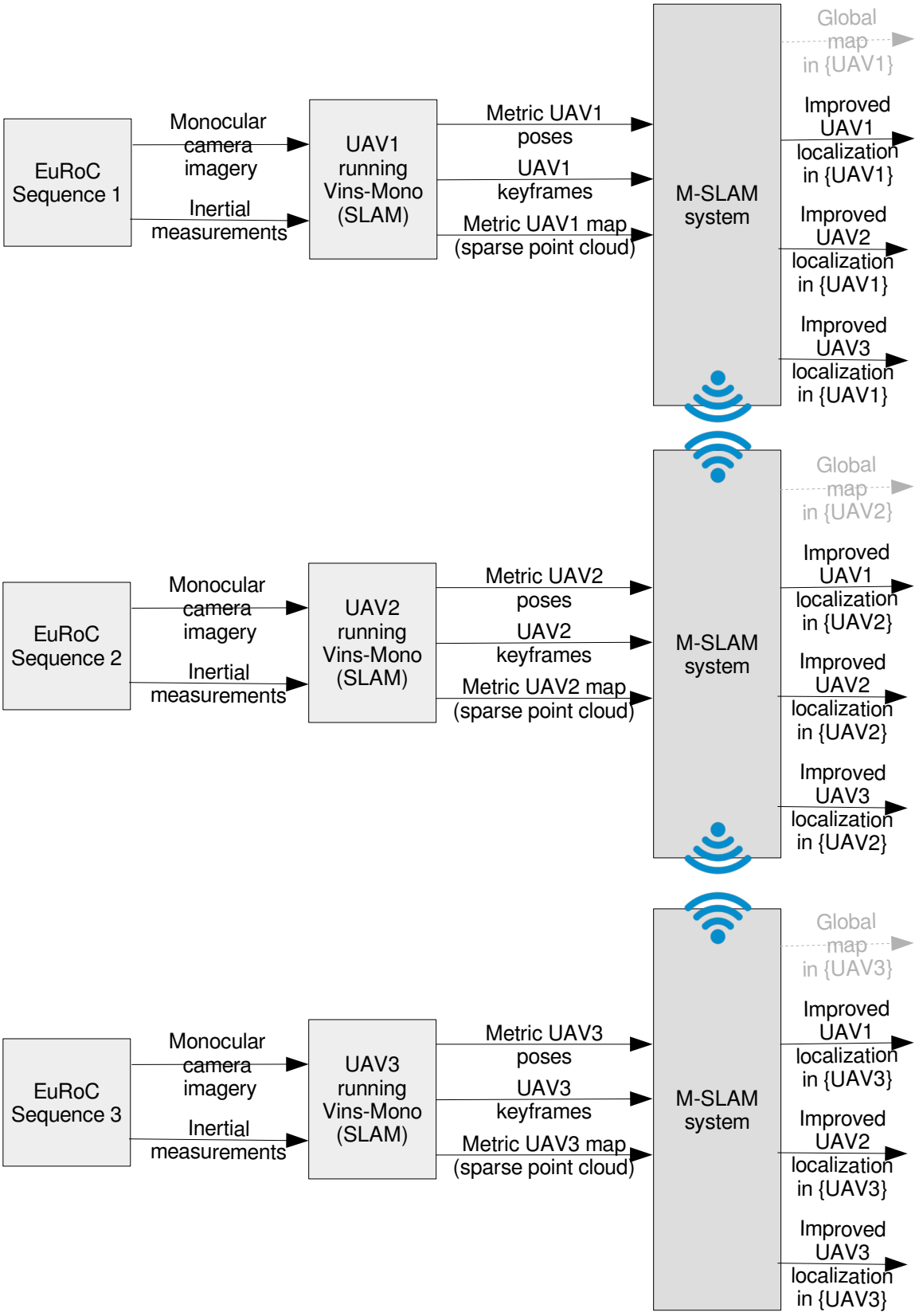


Figure 5.1: Overview of the architecture - case of a tightly-coupled fusion.

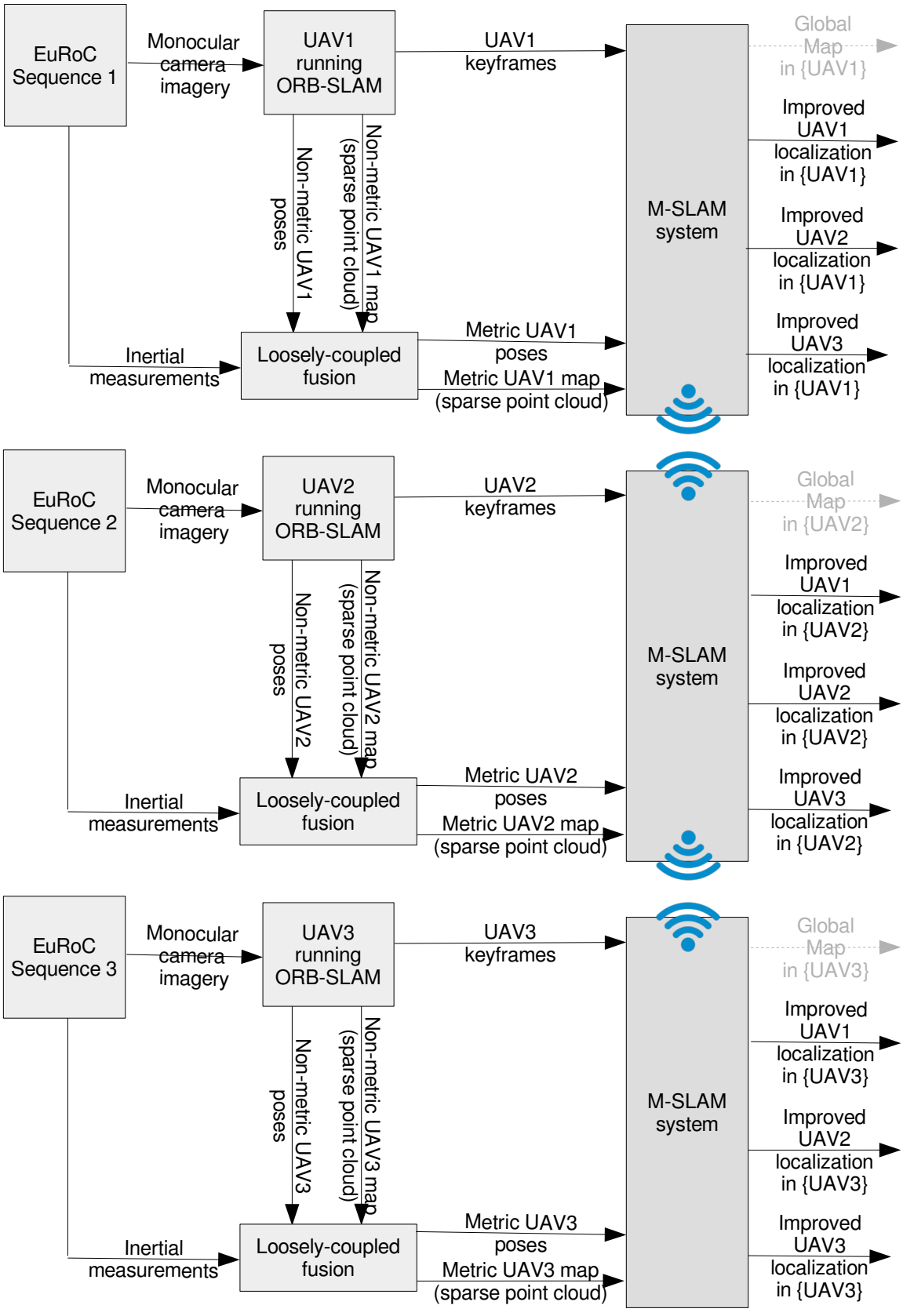


Figure 5.2: Overview of the architecture - case of a loosely-coupled fusion.

5.6 Inputs and process overview

The SLAM front-end gathers the required data from each UAV of the fleet: The keyframes elected by the single-UAV SLAM or, at least, the keypoints and their descriptors, the metric pose of each UAV in its coordinate frame ${}^{\mathrm{UAV}_k}\mathbf{T}_j^{\mathrm{i}}$ (the j^{th} pose, associated to the j^{th} keyframe, of the inertial coordinate frame attached to the UAV_k) and the metric map (sparse 3-D point cloud) associated to the area explored by each UAV in its coordinate frame. The keyframes are sent to the coincidence detection component to find coincidence candidates (Section 5.8). Then, using a 2D-2D RANSAC algorithm [LD13] that estimates the essential matrix, we find a first normalized estimate of the coincidence transformation and filter the false detection (Subsection 5.9.1). Finally, the metric map points are used in a 3D-2D RANSAC algorithm [LMF09] to refine the SE(3) transformation and estimate the norm of the translation vector, as well as discarding the possible remaining false coincidence candidates (Subsection 5.9.2).

5.7 About the network

In this work, we assume that UAVs exchange data, which implies the use of a network infrastructure. The network and communication aspects are not discussed in this thesis. We assume that, each time a UAV elect a new keyframe, it broadcasts the following data:

- The UAV identifier and the keyframe identifier (ID or timestamps),
- The metric pose in its coordinate frame that corresponds with the new keyframe ${}^{\mathrm{UAV}}\mathbf{T}_j^{\mathrm{i}}$, and
- The associated estimated metric map (sparse 3-D point-cloud that is the depth estimation of the processed features n the keyframe).

We also assume that the rest of fleet receives entirely those data.

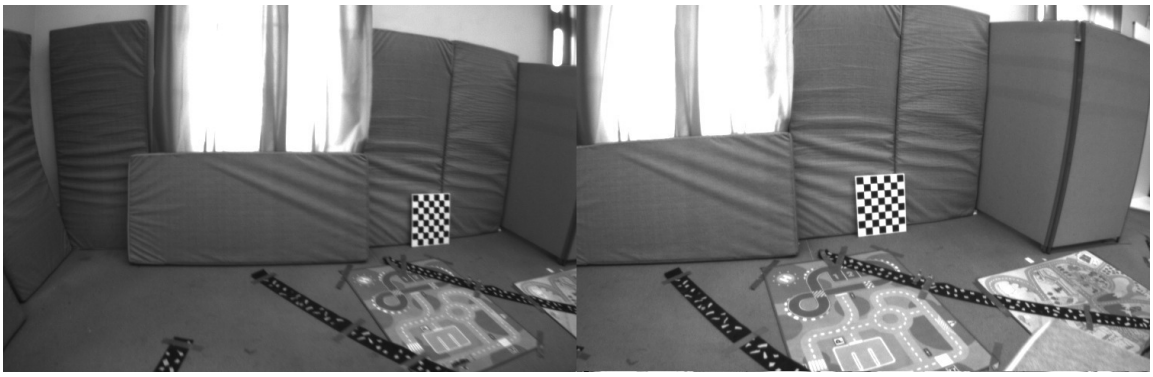


Figure 5.3: An example of the detection of a good coincidence candidate.

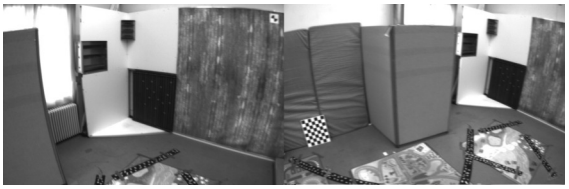


Figure 5.4: Detection of a good coincidence candidate despite blurred images.

Figure 5.5: Detection of a good coincidence candidate despite a massive occlusion.

5.8 Coincidence detection

Once the data of each UAV have been gathered and stored, they are sent to the coincidence detection system. We have decided to use a feature-based system for its robustness and simplicity. Like in the most recent feature-based SLAM algorithms [MMT15] [QLS17], we use the Bags-of-Words approach [GT12] that provides a convenient way to use visual words for fast visual comparison. The Bags-of-Words approach allows to quickly get a similarity score between a pair of images. This approach detects a lot of candidates for coincidence, including false detections that must be discarded to prevent the pose-graph to be badly constraint. Figures 5.3, 5.4 and 5.5 illustrate high quality coincidence candidates detected by the detection system, however, false detections are also numerous in the approach we chose as represented by Figure 5.6, and must be filtered out.

In order to detect the coincidences candidates, we tried two slightly different methods though both of them are based on Bags-of-Words. The first one was directly inspired from VINS-Mono [QLS17] and uses the BRIEF features [Cal+10]

Figure 5.6: Two examples of false detection of coincidence candidates.

provided by the SLAM algorithm VINS-Mono. The second one was build similarly to the coincidence system of ORB-SLAM [MMT15] and uses ORB features [Rub+11].

The results given by the BRIEF features were not satisfactory. A lot of coincidences were detected, but most of which were false detections. Therefore, a powerful filtering mechanism was required in order to discard all those false detections. We could not afford to keep false detections because it would have corrupted the results in the back-end process. The usual way to filter the detection is to run a Fundamental matrix RANSAC algorithm [HZ03] [FB87] (Subsections 5.9.1 and 5.9.2). However, the experiments showed that in order to have perfectly filtered results, we ended having few coincidences candidates in a three UAVs experiment (less than five, which is not enough). As we use those coincidences to constrain the pose-graph to improve the estimation of the location of the UAVs, we needed to detect more good quality coincidence candidates, which would help the filtering process.

The ORB based process provided better results than the BRIEF based process as displayed in Figures 5.3, 5.4 and 5.5. This is due mainly to the fact the ORB is a better feature than BRIEF and most importantly because we use pyramid levels in ORB. Using ORB features, we can have numerous coincidence candidates with good quality (almost no false detections). Therefore, even though ORB features are slower to extract than BRIEF, we cannot afford the poor quality results given by BRIEF features. Figure 5.7 illustrates the coincidence detection part of the front-end of the M-SLAM system.

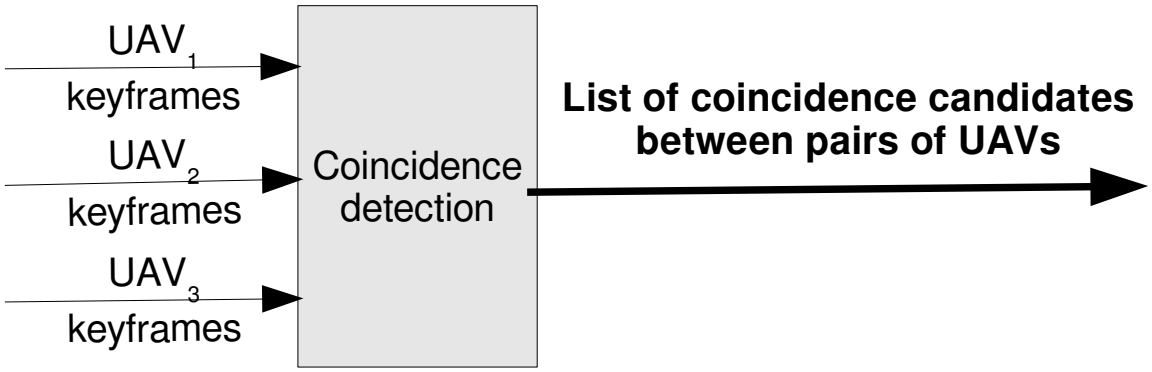


Figure 5.7: Diagram of the coincidence detection process for a three-UAV experiment.

5.9 Estimation of the SE(3) constraint

5.9.1 Essential matrix estimation

After the detection of coincidence candidates, we compute a first estimate of the SE(3) transformation between the two camera coordinate frames involved in the coincidence. The estimation of those SE(3) constraints (one per coincidence) is critical as it is the main new measurement added to the pose-graph that is used in the back-end to improve the UAV location. We use standard algorithms to compute the transformation [Nis04], [LD13], [LMF09] though we must distinguish the intra-UAV coincidence case from the inter-UAV coincidence case because we observed that the inter-UAV coincidences were more challenging to process than intra-UAV coincidences. The critical difference between inter-robot and intra-robot coincidences is the knowledge about the SE(3) transformation between the matched pair of keyframes. In a single UAV system, intra-robot coincidences are detected along the trajectory of the same robot. When a coincidence is detected and closed, thanks to the SLAM estimates, the system already has a first estimate of the transformation between the two edges of the coincidence. Even if the SLAM estimates are drifting, there is a pose estimate expressed in the same coordinate frame for both edges because all the measurements are given in the same coordinate frame, i.e., the local UAV coordinate frame {UAV} as illustrated in Figure 5.8.

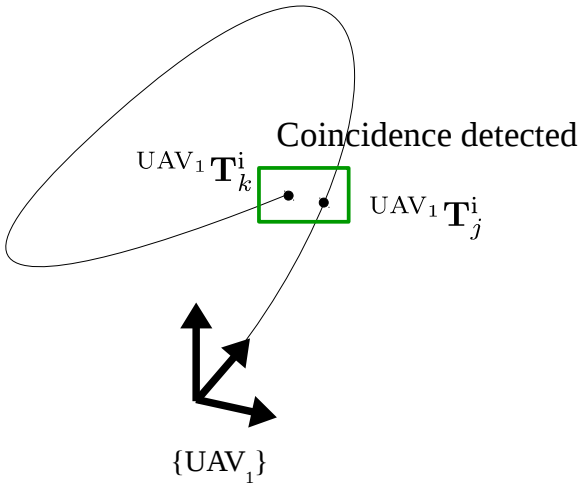


Figure 5.8: Differences between coincidences: A example of intra-UAV coincidence.

In a multi-UAV system, there is no information about the transformation between the two edges of an inter-robot coincidence because each UAV estimates is expressed in a different coordinate frame (at least until the first coincidence between two UAVs occurs). Therefore, as shown in Figure 5.9, we have no knowledge about the transformation between the two edges of the loop. We must compute a first estimate of this transformation in order to ensure a satisfactory data fusion process.

As displayed in Figure 5.7, after isolating the inter-UAV coincidences using the Bags-of-Words approach, we can process each coincidence to estimate the associated transformation $^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ (the transformation between the camera coordinate frames of UAVs a and b at respective time, or keyframes, i and j). We use the geometry of the keyframes to compute the transformation as shown in Figure 5.10. A usual method is to use approaches like Fundamental Matrix RANSAC [Har95] or Essential Matrix RANSAC [Lon87]. In our system, the camera are calibrated, we know the intrinsic parameters of the camera model, therefore we can use either the fundamental or essential matrices. We detect and match SIFT features [Low04b] in the pairs of keyframes that are provided by the coincidence detection process. Then, we use an iterative 5-points algorithm to estimate the essential matrix $\mathbf{E}$ [LD13]. The essential matrix transcripts the geometric constraints between two overlapping images. In other word, the essential matrix includes

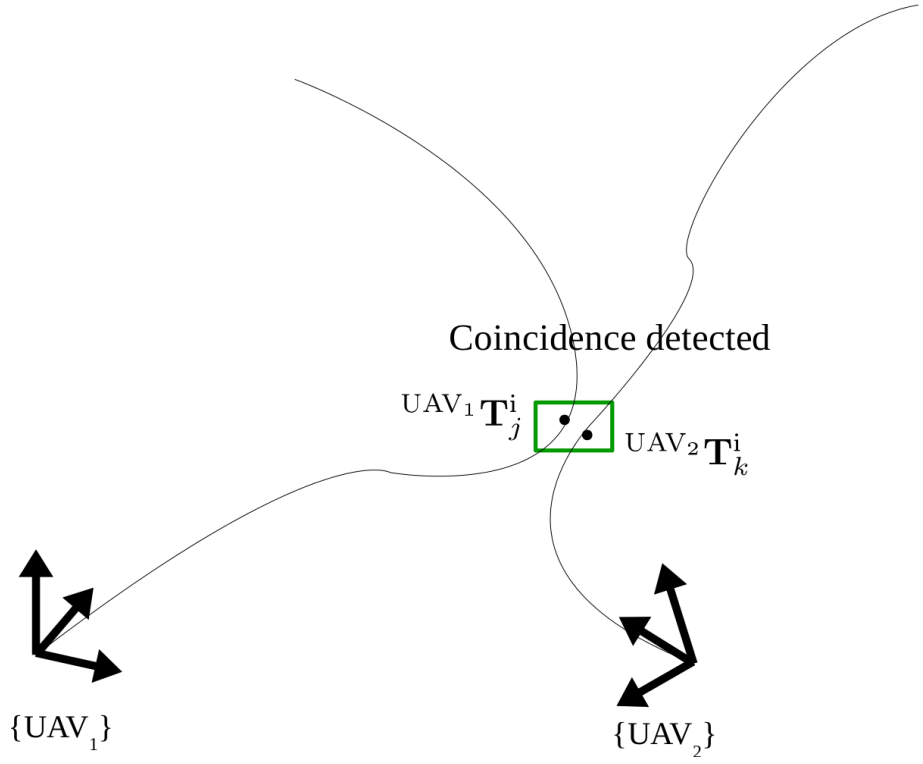


Figure 5.9: Differences between coincidences: An example of inter-UAV coincidence.

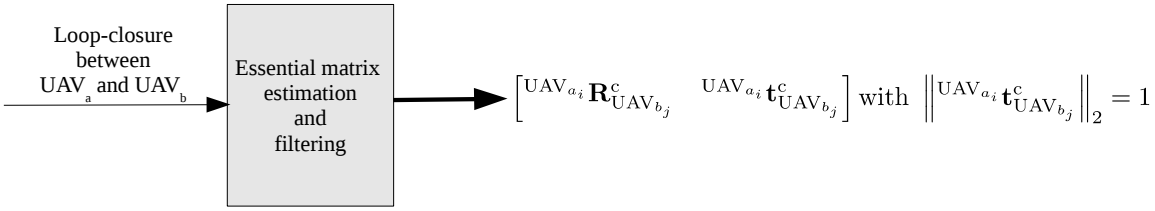


Figure 5.10: Diagram of the estimation of the essential matrix process.

the 3-D rotation ${}^{\mathrm{UAV}_{a_i}}\mathbf{R}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ and the direction of the translation ${}^{\mathrm{UAV}_{a_i}}\mathbf{t}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ (with $\|{}^{\mathrm{UAV}_{a_i}}\mathbf{t}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}\|_2 = 1$) between the two view points. The RANSAC part included in [LD13] that is used to select the inlier point correspondences also allow to filter some false detections: If there is not enough inliers, we discard the coincidence candidate.

In Section 5.8, we discussed the use of either BRIEF or ORB descriptors and concluded that ORB descriptors provided satisfactory results. However, to perform the matching of features required to have a satisfactory Essential Matrix RANSAC result, ORB descriptors were not descriptive enough with regard to the challenging texture of the EuRoC datasets we used for the experiments. Therefore, we decided

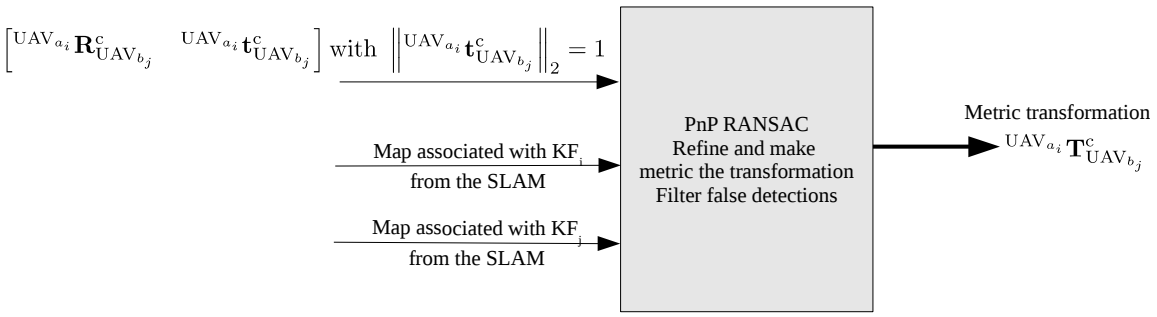


Figure 5.11: Diagram of the PnP scheme.

to use one of the most descriptive feature: SIFT. This choice comes at a cost, the extraction and matching parts are much slower than using other descriptors like ORB. Environments with more unique and distinctive textures than the EuRoC environment would allow to use faster features to compute like ORB.

5.9.2 Use of the metric map

So far, we used only the 2-D information available in the keyframes to compute ${}^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$. Therefore, the translation vector is normalized, so we have no information about the metric distance between the two camera coordinate frames. In order to refine the estimate of ${}^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ and to make the translation metric, we use the PnP algorithm [LMF09] as described in Figure 5.11. Hence, we use the 3-D map points to find the optimal ${}^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ such that the re-projection error is minimized. In other word, each map point of the keyframe i is projected in the keyframe j, the re-projection error is the difference between the corresponding keypoint in j and the projection of the map point. The PnP is initialized with the transformation computed from the essential matrix $\mathbf{E}$. As the map provided by the SLAM is metric, the resulting ${}^{\mathrm{UAV}_{a_i}}\mathbf{T}^{\mathrm{c}}_{\mathrm{UAV}_{b_j}}$ is also metric. Similarly to the previous step, the RANSAC part that is used to select the inliers is used to discard false detections. Figure 5.12 summarizes the whole process from the detection of candidates to the filtered coincidences with the SE(3) constraint estimated.

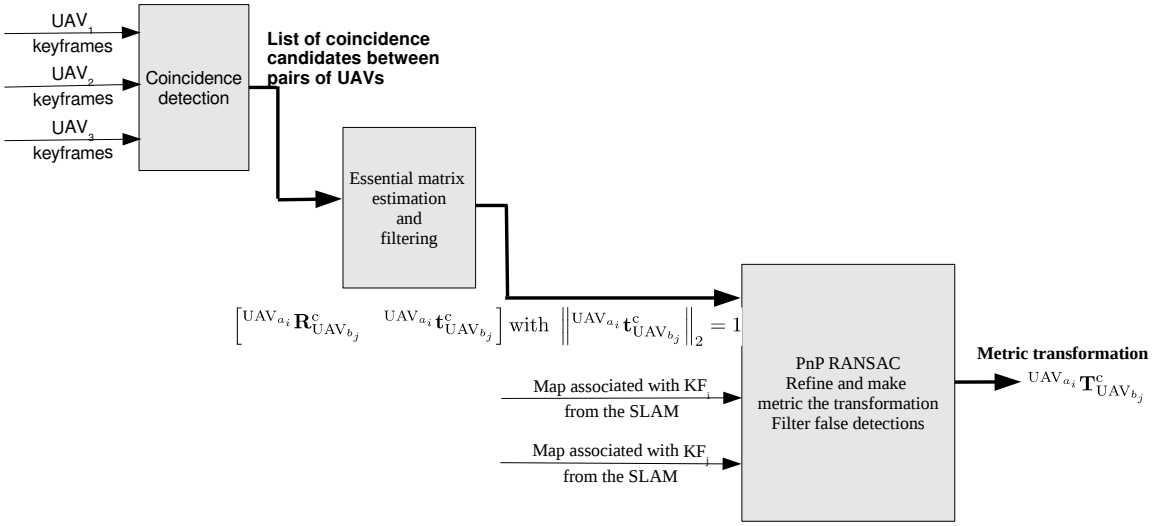


Figure 5.12: Overview of the process to generate an additional constraint in the pose-graph using inter-UAV coincidences.

5.10 Choice of descriptors

The feature matching is an important part of the coincidence detection and SE(3) constraint computation. As each UAV flies in its own world, after the system initialized there is no information about the relative pose between each UAV of the fleet. The roles of feature descriptors are the following:

- Detect candidates for the coincidences,
- Make correspondences between the features to compute geometrical constraints.

The coincidence candidates are found using the Bag-of-Words approach that provide a similarity score for each pair of images based on the feature descriptors. The feature matching is performed by comparing the descriptors. Therefore, it is important that each feature is described uniquely by its descriptor independently of the scale and the orientation. During the development of M-SLAM, three different descriptors were tried: BRIEF, ORB and SIFT descriptors. BRIEF descriptors are not unique enough to ensure a satisfactory detection for the coincidence candidate and provide a poor quality feature matching. ORB descriptors provide satisfactory coincidence candidates but are not descriptive enough to perform the feature matching

to estimate the essential matrix using the EuRoC dataset for the experiments. Finally, we chose to use SIFT descriptors as they provide satisfactory results for the feature matching. The main drawback of SIFT descriptors in comparison to ORB and BRIEF descriptors is that the extraction and comparison of SIFT descriptors are much slower than for ORB and BRIEF descriptors.

5.11 Accuracy of the coincidences

For later data fusion purpose, we should determine a quality value for the coincidences that contains information about how accurate the SE(3) constraint has been estimated. At the end of the M-SLAM research work, we have not been able to find a relation between the accuracy of the estimation of the SE(3) constraint using the ground truth value and the values related to coincidences, namely: the number of inliers before and after each RANSAC, the bags-of-word similarity score and the final PnP reprojection error (total and average per feature). This part remains for further development and requires a deeper and more systemic study.

5.12 Conclusion

We presented the front-end of the M-SLAM algorithm. Our front-end includes a place recognition scheme to detect candidates for coincidences, the estimation of the relative transformation between the keyframes involved in the coincidence using the 2-D information encoded in the images or keypoints and the 3-D map. RANSAC algorithms are used to refined the coincidence constraint - the SO(3) transformation - and are also critical to filter false coincidence candidates. Our method is based on features and we provided observations about the choice for descriptors that can be of great importance to ensure satisfactory detection and filtering.

CHAPTER SIX

MULTI-UAV SLAM: PROCESSING OF THE POSE-GRAPH AND FUSION OF LOCATION INFORMATION

6.1 Introduction

In this chapter, we describe the back-end of the M-SLAM system. We consider that the front-end has gathered the UAV's data and provide to the back-end process a list of filtered coincidences with their metric transformation (the transformation between the camera coordinate frames of the two view points of the scene scenery).

6.2 Objective

The back-end of the M-SLAM builds the multi-UAV SLAM pose-graph and fuses the information about the UAV locations. The tasks of the back-end are the following:

1. Generate the pose-graph including the coincidence constraints,

2. Fuse the location information in the pose-graph,

3. Update the UAV states and provide the relative localization of each UAV to the rest of the fleet.

6.3 M-SLAM pose-graph

The pose-graph represents the localization of the fleet from one UAV point-of-view. As the system is fully distributed, each UAV builds a different pose-graph with all the measurements expressed in its own coordinate frame $\{\text{UAV}_a\}$. The graph consists in nodes and edges. Each edge corresponds to a constraint (a relative motion to transform a pose into another pose) and each node corresponds to the pose of one UAV at one moment in time, when a UAV's SLAM elects a new keyframe. The data fusion algorithm improves the pose estimate for each node using the known constraints at the moment the graph is built. The graph construction of a UAV_a is done through the following steps:

- First, a node is created for each keyframe of the UAV_a. It contains the pose value of the UAV_a in the coordinate frame $\{\text{UAV}_a\}$. The pose comes from the front-end of the M-SLAM algorithm, namely the transformation ${}^{\text{UAV}_a}\mathbf{T}^{\text{i}}_j$,
- Second, edges are generated between each node and represent the relative motion between the consecutive nodes j and k: ${}^{j}\mathbf{T}^{\text{i}}_k$,
- Third, intra-coincidences of UAV_a are added, they are represented by an edge containing the constraint between the two nodes involved in the intra-UAV coincidence m and n: ${}^{m}\mathbf{T}^{\text{i}}_n$,
- Fourth, if a direct or indirect inter-coincidence has occurred between the UAV_a and another UAV of the fleet, such as UAV_b, the nodes of the UAV_b are also added to graph. We use the direct coincidence with the smallest covariance matrix(or indirect coincidence if there are no direct inter-coincidence) ${}^{j}\mathbf{T}_k$ to express the poses of the UAV_b in the coordinate frame $\{\text{UAV}_a\}$. The pose of UAV_b comes from the front-end of the M-SLAM known as the transformation ${}^{\text{UAV}_b}\mathbf{T}^{\text{i}}_j$.

$$ {}^{\text{UAV}_a}\mathbf{T}_j = {}^{\text{UAV}_a}\mathbf{T}_{\text{UAV}_b}\ {}^{\text{UAV}_b}\mathbf{T}^{\text{i}}_j \tag{6.1} $$

$$ {}^{\text{UAV}_a}\mathbf{T}_{\text{UAV}_b} = {}^{\text{UAV}_a}\mathbf{T}^{\text{i}}_k\ {}^{j}\mathbf{T}^{\text{i-1}}_k\ {}^{\text{UAV}_b}\mathbf{T}^{\text{i}^{-1}}_j \tag{6.2} $$

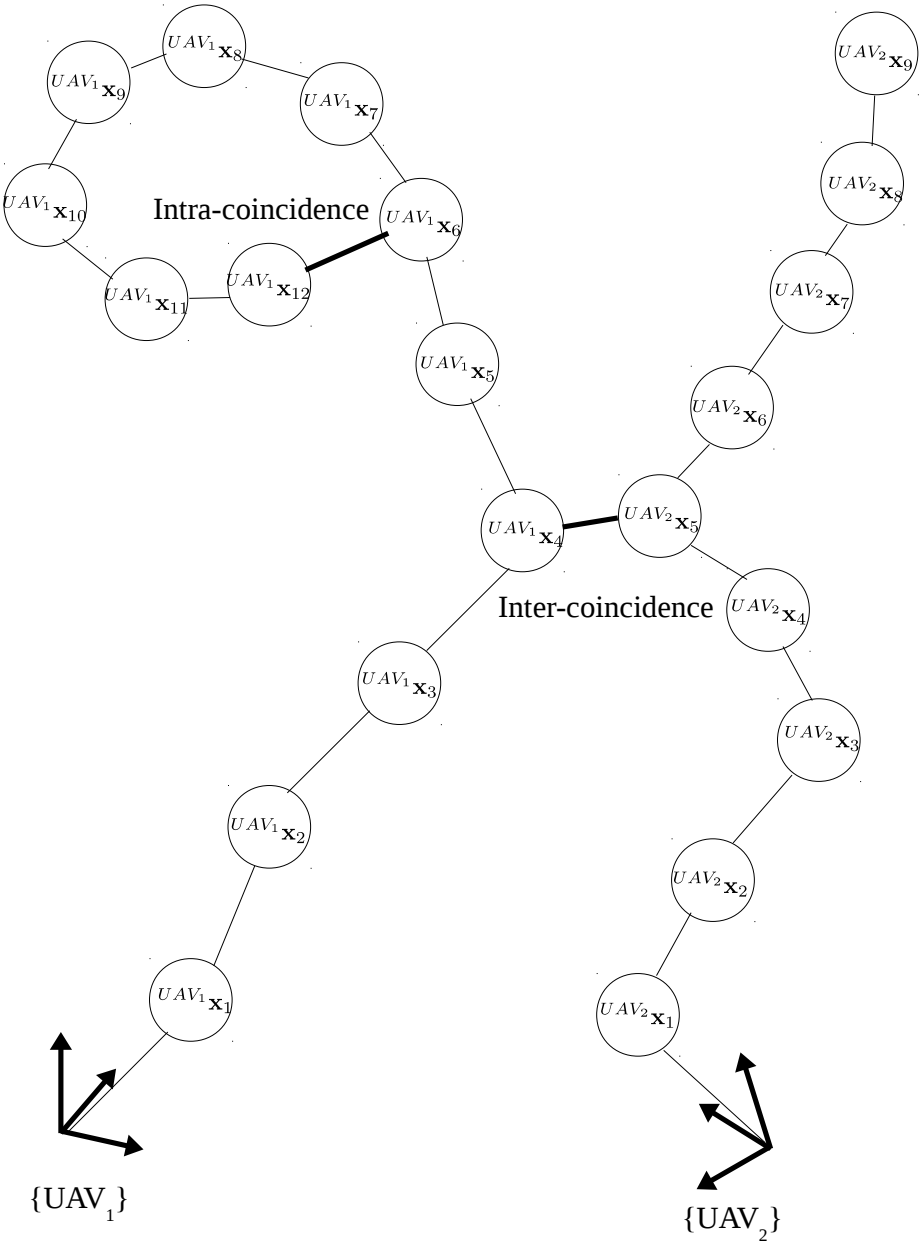


Figure 6.1: An illustration of the pose graph during a two-UAV experiment, ${}^{\text{UAV}_a}\mathbf{x}_j$ represents the pose (or the value of the state vector) at the time the keyframe j was elected.

- Fifth, similarly to step two, edges are added to represent the relative motion between the consecutive nodes of the UAV$_b$ in {UAV$_b$}, and

- Finally, the inter-coincidences are added between the nodes j of UAV$_a$ and the nodes j of UAV$_b$. The inter-coincidence SE(3) constraints ${}^{j}\mathbf{T}_k$ are computed by the front-end.

Figure 6.1 provides an illustration of the pose-graph that can be obtained.

6.4 Significant differences between coincidences and loop-closures in single-robot SLAM

6.4.1 Single robot case

In real-time SLAM algorithms, the loop-closures scheme is always done through the same mechanism. The robot is moving, generating poses throughout its estimated trajectory. For the sake of comprehension, we associate each estimated pose to a node in a graph. At a moment t, a loop-closure is detected between two non-consecutive nodes n_i and n_j. Then, the node n_j is corrected using the pose estimate of n_i and, finally, the correction is propagated to the nodes in between n_i and n_j. Figures 6.2, 6.3 and 6.4 provide an example of the traditional processing of a loop-closure in single robot SLAM. There are three properties that must be underlined:

- n_j is the latest node of the trajectory at time t,
- n_j has a greater error than n_i,
- The correction is propagated to identified nodes: the nodes in between n_i and n_j, the other nodes are not affected.

As a consequence, in single robot systems, when a loop-closure is processed the following pieces of information are always known:

- Between two nodes, which node is relatively wronger than the other,
- What nodes can benefit from the correction,
- The orientation of the loop-closure: The oldest node is used to correct the latest node,
- The corrected node is always the last available node of the trajectory at time t, therefore, loop-closures do not have an effect on the future nodes of the trajectory (generated after time t), the corrected value n_j is simply used as the new starting point for the odometry.

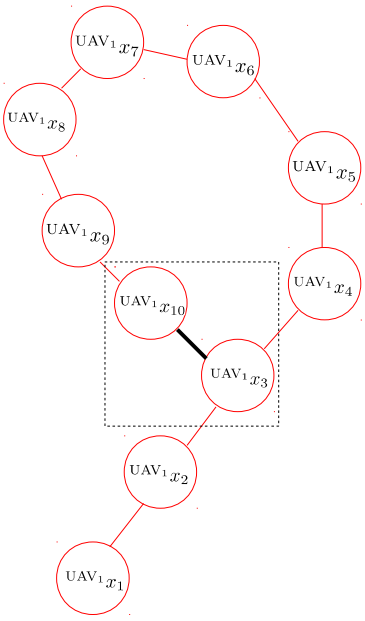


Figure 6.2: Loop-closure in single robot SLAM: Detection of the coincidence.

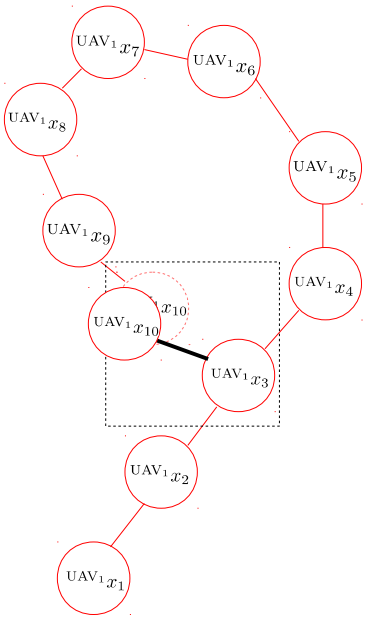


Figure 6.3: Loop-closure in single robot SLAM: Correction of the node.

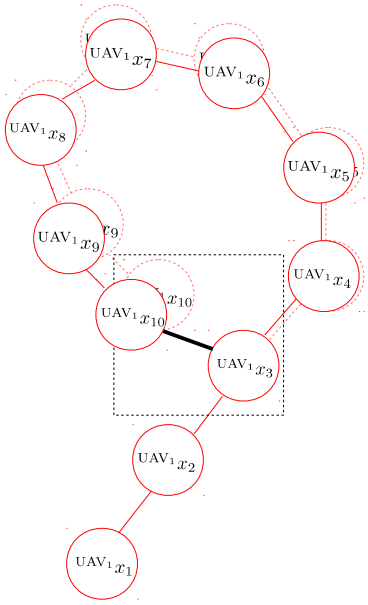


Figure 6.4: Loop-closure in single robot SLAM: Propagation of the correction.

Most of those properties and knowledge are no longer available in a multi robot system as we will discuss in the next subsection.

6.4.2 Multi-robot case

Orientation of the coincidence

In a multi robot system the latest node is not always corrected using a prior node. The correction is done using the node with the smallest error which can be any of the two nodes involved. In inter-coincidences, there is no notion of prior or later nodes as they are provided by two different UAVs and the error can grow differently in each UAV regarding the kind of trajectory or images that have been processed as illustrated in Figures 6.5 and 6.6. It is also valid for intra-coincidences. As inter-coincidences can have been processed prior to the intra-coincidence, it can happen that the latest node have a smaller error than a prior node, Figures 6.7, 6.8 and 6.9 illustrates this mechanism.

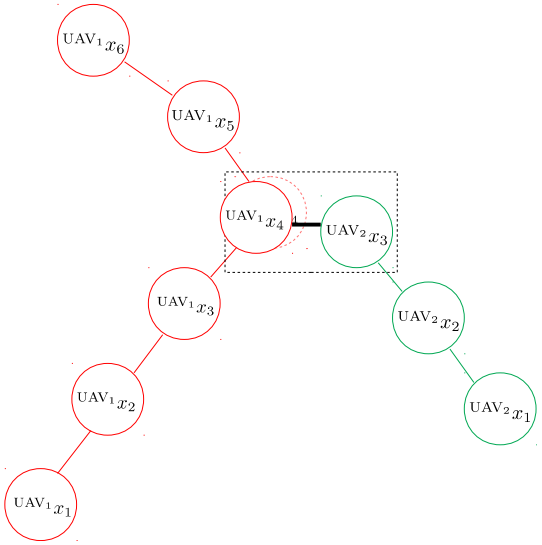


Figure 6.5: The possible orientations in inter-coincidence, correction of the wrongest node, here in the UAV 1 trajectory.

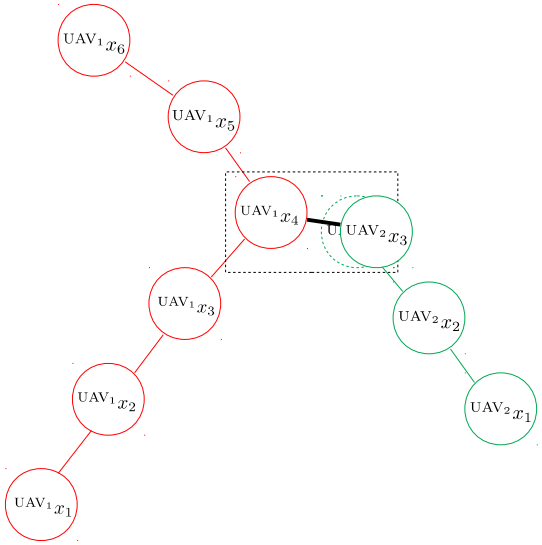


Figure 6.6: The possible orientations in inter-coincidence, correction of the wrongest node, here in the UAV 2 trajectory.

Propagation of the correction backwards

When a node is corrected, the correction is then propagated backwards in the trajectory. The question is: How far should the correction be propagated? In single robot system, the answer is simple, the correction is propagated in between the two nodes involved because the later node is always wronger than the older one and the nodes prior to the older node have already a smaller error and therefore cannot benefit from the correction.

In a multi-robot system, the answer is not that simple because, most of the time, the coincidences do not inherently define until where the correction must be propagated. Figure 6.11 provides an example: until what node should the correction be propagated? Therefore, in a multi-robot system, knowledge about the error on the localization is way more important than in single robot system as they are used to define the end of the propagation of the correction. As discussed in Subsection 6.4.2, the problem of the impacted nodes by the backward correction is encountered both in inter-coincidences and in some intra-coincidences regarding the orientation of the coincidence.

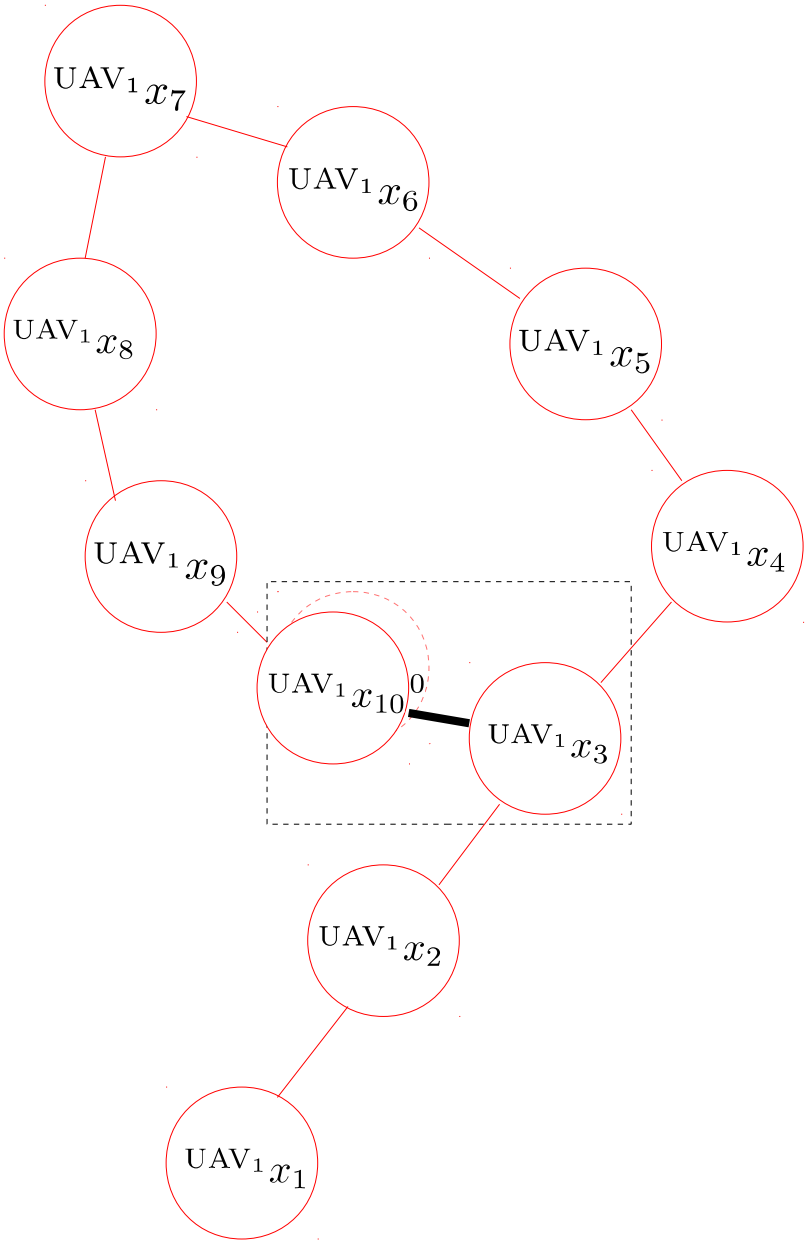


Figure 6.7: In some case, intra-coincidences can be processed like usual single-robot loop-closures.

Effect on the trajectory forwards

In addition to the question of how far should the correction be propagated, the effect of coincidences forward of the corrected node becomes also important in multi-robot system while it has barely no importance in single-robot system. As we can correct nodes in the past, those corrected nodes have an effect on the rest of the trajectory as we re-built the rest of the trajectory using as a new starting point the corrected value of the node. As further nodes can be involved in other

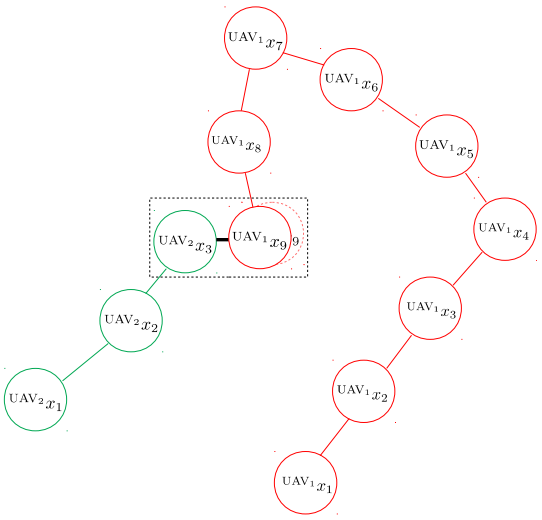


Figure 6.8: Differences between intra-coincidences and loop-closures: An inter-coincidence occurs, the node x_9 of UAV 1 is corrected.

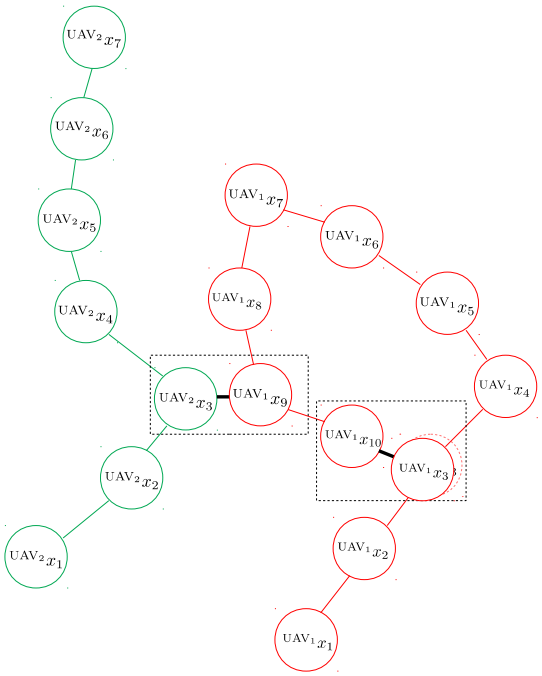


Figure 6.9: Differences between intra-coincidences and loop-closures: Due to the inter-coincidence, the node x_{10} has now a smaller error than the node x_3, the node x_3 is corrected.

coincidences, it is important to take this effect into consideration, Figures 6.12, 6.13, 6.14 and 6.15 provides an example of the entire process for a coincidence including the forward effect. In traditional single-robot SLAM, this forward effect is barely observed for the following reasons:

- Either the system solves a full-SLAM problem, all the measurements and loop-closures are gathered and an optimization scheme solve the whole problem at the end of the experiment,

- Or the system works in real-time (or at least, on the go) and the graph is optimized after each loop-closure. As the corrected node is always the last one measured, further odometry measurements are added using the new corrected value of this node.

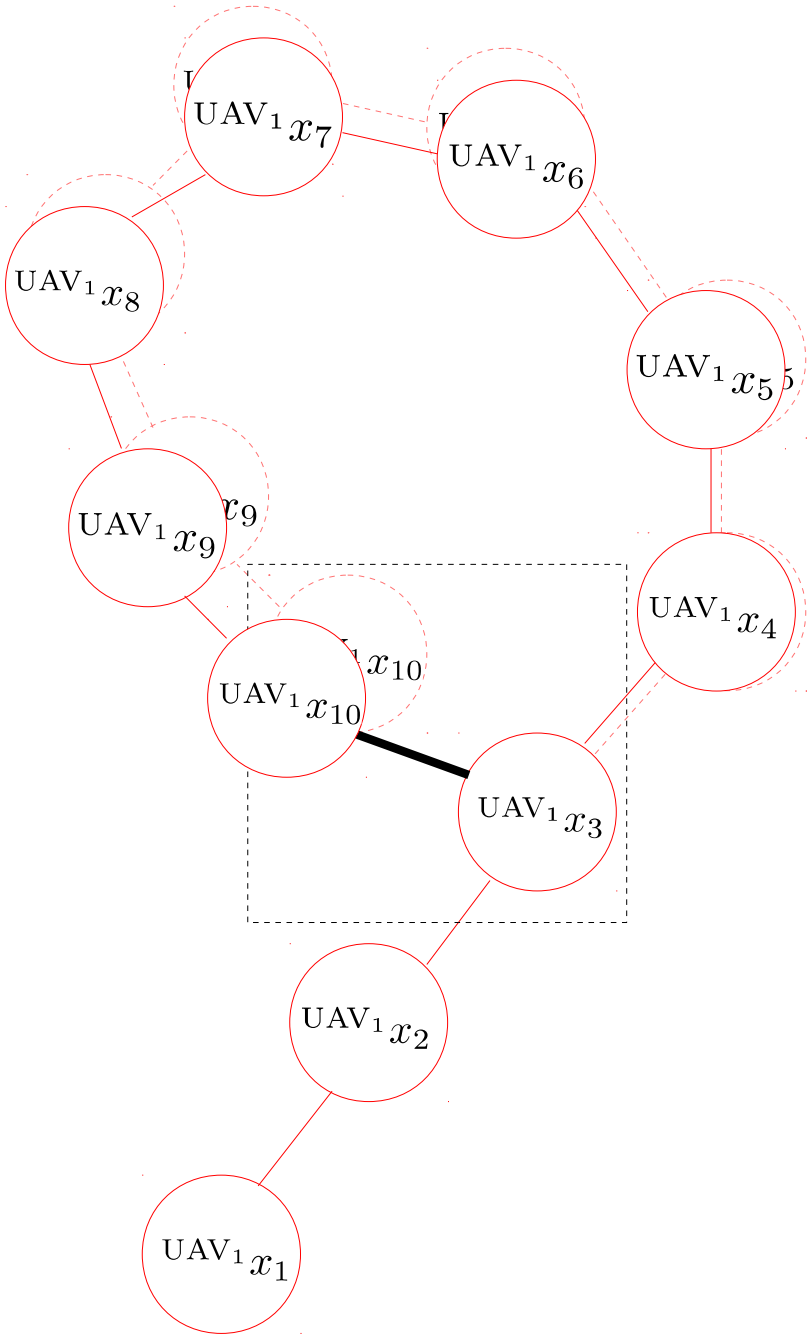


Figure 6.10: Loop-closures, propagation of the correction.

Order for processing the coincidences

The order for processing the coincidences is a non-existent problem in single-robot SLAM because when a loop-closure is detected, it is processed and then the new nodes are added to the system until the next loop-closure.

In a multi-UAV system with coincidences between the UAVs, the order become much important as the most recent coincidence can modify past nodes in the trajec-

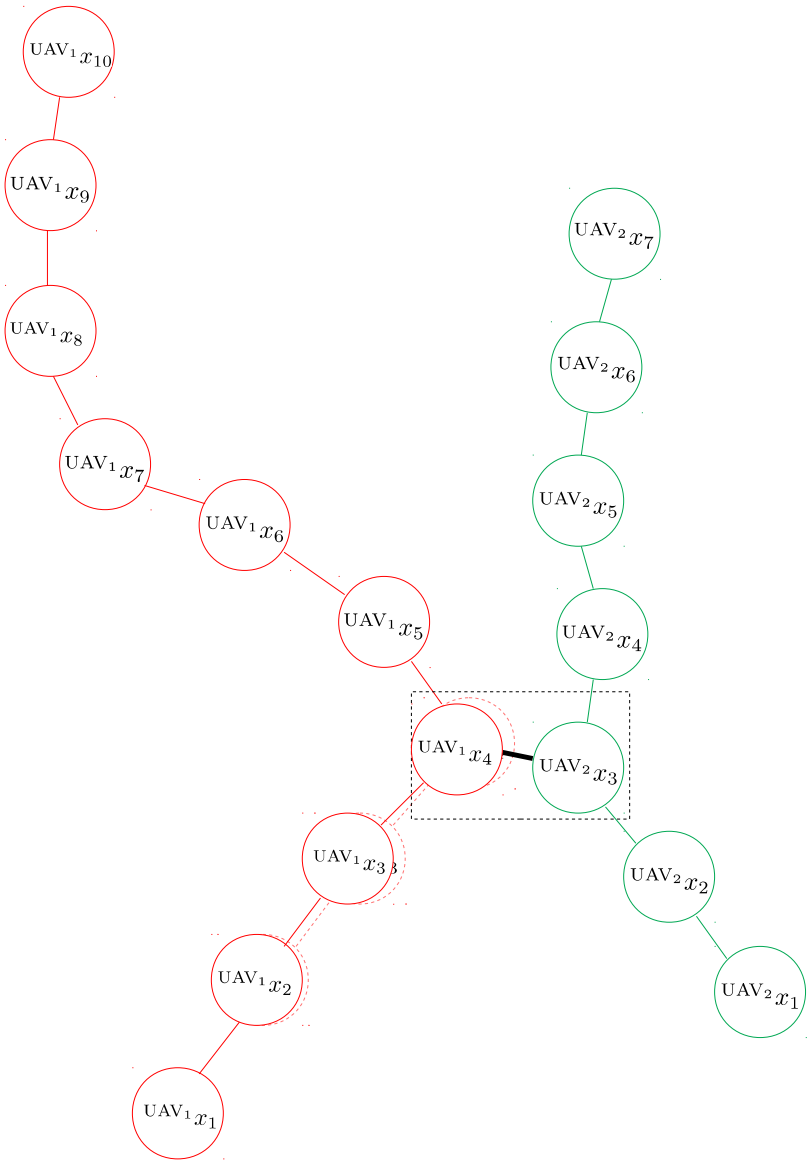


Figure 6.11: Inter-coincidence, propagation of the correction until what node?

tory that could have been involved in previous coincidences. In this context, does it still make sense to process the coincidences on-the-go, when there are detected? We decided to process the coincidences using a different criteria than the moment they are detected. We tried two criterion: i) Processing the coincidences from the most significant correction to the less significant correction, and ii) processing the coincidences from the earliest correction of the trajectories to the latest effect on the trajectories. The first criterion (i) is used to maximize the benefits provided by the most significant node corrections for the further corrections, but due to the forward effect, it can also increase the absolute error in the trajectory afterwards. The second

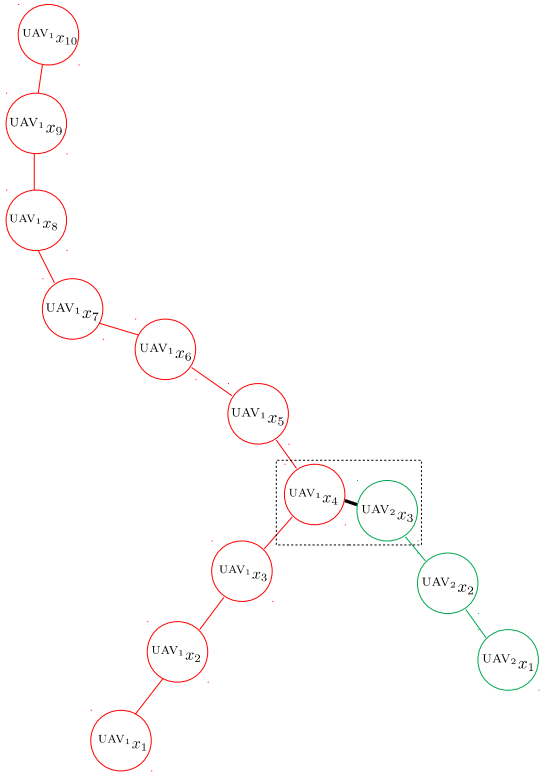


Figure 6.12: Detection of the coincidence.

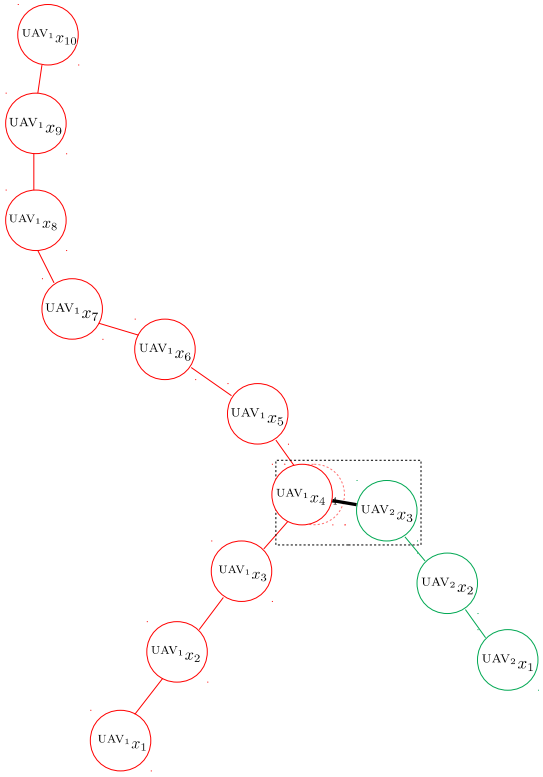


Figure 6.13: Correction of the relative wrongest node.

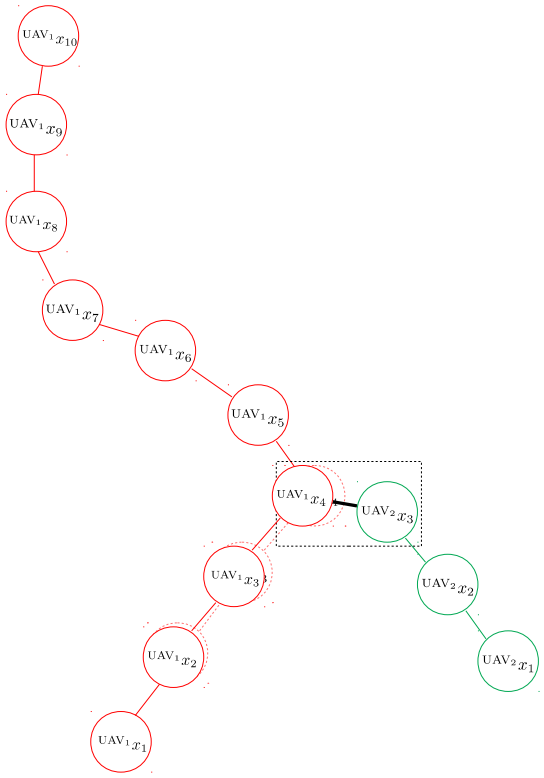


Figure 6.14: Propagation of the correction backwards.

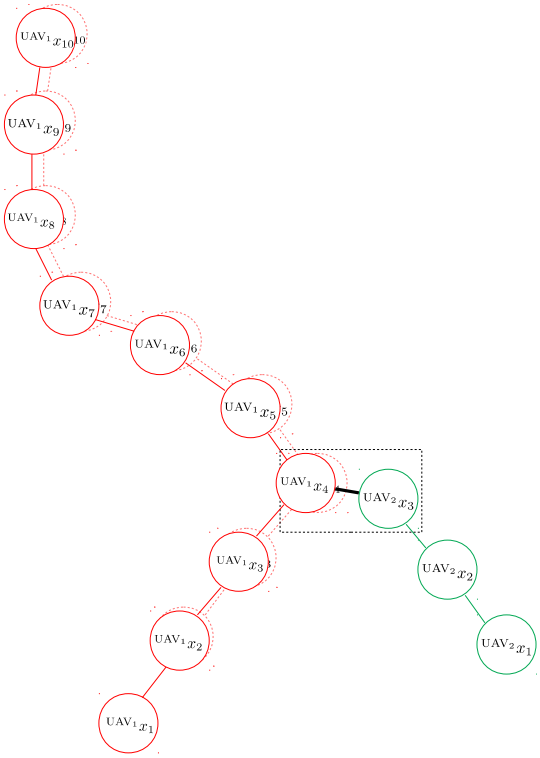


Figure 6.15: Re-computing the rest of the trajectory using the corrected value as a new starting point.

criterion (ii) has the benefits to minimize the forward effect on the trajectories before processing further coincidences, but, as a consequence, the trajectory do not benefit entirely from the most significant correction possible.

6.5 Process of a coincidence

The processing of a coincidence is done in three steps:

1. Correction of one of the node involved in the coincidence,
2. Propagation of the correction backwards,
3. Re-building of the trajectory forwards.

For the sake of notations, we define the operator $(\cdot)$ that applies a 3-D transformation to a pose vector. A pose vector is a 7-element vector that defines a pose in 3-D: Three scalars define the 3-D translation, and four scalars are the coefficients of the quaternion that is used to represent the orientation.

$$\mathbf{x} = \begin{bmatrix} t_x & t_y & t_z & q_w & q_x & q_y & q_z \end{bmatrix}^T \tag{6.3}$$

In the following subsections, we transform a pose vector into another pose vector using 4-by-4 homogeneous matrices that encapsulates both the relative 3-D translation and rotation.

$${}^{i}\mathbf{x}_k = {}^{i}\mathbf{T}_j \cdot {}^{j}\mathbf{x}_k \tag{6.4}$$

From a numerical point of view, we consider that $\mathbf{T}$ encapsulates a rotation that can be represented either by a quaternion or a 3-by-3 rotation matrix, and a 3-by-1 translation vector. We chose to represent constraints (motion to go from one pose at time t to the next pose at time $t+1$ and coincidences) with 4-by-4 homogeneous matrices denoted $\mathbf{T}$, and to represent the poses of the UAVs with 7-by-1 state vectors, the changes in the state vectors defines the trajectory of the UAVs. A 7-by-7

covariance matrix is attached to each state vector to express the inaccuracy of the pose estimates. Those choices were made to simplify the comprehension of the reader as both the constraints and the representation the UAV trajectories are made of poses, which are 3-D rigid body transformation (3-D rotation and translation) and to make easier the incorporation of covariance matrices in the M-SLAM system.

6.5.1 Correction of the node

A coincidence involves two nodes, we determine which node is relatively wronger than the other, therefore information about errors is required. Then, the most accurate node is used to correct the other node. Considering a coincidence between the node ${}^{\mathrm{UAV}_a}\mathbf{x}_k^{\mathrm{i}}$ from UAV_a and the node ${}^{\mathrm{UAV}_b}\mathbf{x}_l^{\mathrm{i}}$, and assuming that the covariance matrix used to represent the errors P_l on the node ${}^{\mathrm{UAV}_b}\mathbf{x}_l^{\mathrm{i}}$ is greater than the covariance matrix $\mathbf{P}_k$ on the node ${}^{\mathrm{UAV}_a}\mathbf{x}_k^{\mathrm{i}}$. We assume that a covariance matrix $\mathbf{P}_i$ is greater than $\mathbf{P}_k$ if the sum of the elements of $\mathbf{P}_i$ is greater than the sum of the elements of $\mathbf{P}_k$. The corrected value of the node ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_l^{\mathrm{i}}$ is given by

$$ {}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_l^{\mathrm{i}} = {}^{\mathrm{UAV}_b}\mathbf{T}_{\mathrm{UAV}_a} \cdot {}^{\mathrm{UAV}_a}\mathbf{x}_m^{\mathrm{i}} \cdot {}^{m}\mathbf{T}_l \tag{6.5} $$

${}^{\mathrm{UAV}_b}\mathbf{T}_{\mathrm{UAV}_a}$ is the transformation between one UAV coordinate frame $\{\mathrm{UAV}_b\}$ and another UAV coordinate frame $\{\mathrm{UAV}_a\}$ and is found by processing coincidences, this point is discussed in 6.3. For the case of intra-coincidences ($k = l$),

$$ {}^{\mathrm{UAV}_b}\mathbf{T}_{\mathrm{UAV}_a} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.6} $$

The initial value of ${}^{\mathrm{UAV}_a}\mathbf{x}_m^{\mathrm{i}}$ is given by the single-robot SLAM running onboard the UAV_a, the value can be modified later on by processing the coincidences gathered in the system. ${}^{m}\mathbf{T}_l^{\mathrm{i}}$ is the measurement of the coincidence, it comes from the

processing of the keyframes and the single-robot SLAM estimates. The covariance matrix of $\mathbf{P}_{\mathrm{UAV}_b \hat{\mathbf{x}}_l^{\mathrm{i}}}$ is computed as follows

$$\mathbf{P}_{\mathrm{UAV}_b \hat{\mathbf{x}}_l^{\mathrm{i}}} = \mathbf{P}_{\mathrm{UAV}_b \mathbf{T}_{\mathrm{UAV}_a}} + \mathbf{P}_{\mathrm{UAV}_a \mathbf{x}_m^{\mathrm{i}}} + \mathbf{P}_{m \mathbf{T}_l^{\mathrm{i}}} \tag{6.7}$$

For the case of intra-coincidences ($i = j$),

$$\mathbf{P}_{\mathrm{UAV}_b \mathbf{T}_{\mathrm{UAV}_a}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.8}$$

6.5.2 Propagation of the correction backwards

When the correction is propagated, first we must determine until where we should propagate the modification. Once again, information about the error on the nodes is critical. Going backwards in the trajectory, we search for the node ${}^{\mathrm{UAV}_b}\mathbf{x}_m^{\mathrm{i}}$ with an covariance matrix $\mathbf{P}_m$ in which all the elements are smaller than the covariance matrix on ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_l^{\mathrm{i}}$ ($m < l$). The nodes ${}^{\mathrm{UAV}_b}\mathbf{x}_n^{\mathrm{i}}$ with $m < n < l$ can benefit from the correction. To compute the new value ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_n^{\mathrm{i}}$ of the nodes ${}^{\mathrm{UAV}_b}\mathbf{x}_n^{\mathrm{i}}$, we define an upper and a lower bound, respectively ${}^{\mathrm{UAV}_b}\mathbf{x'}_n^{\mathrm{i}}$ and ${}^{\mathrm{UAV}_b}\mathbf{x}_n^{\mathrm{i}}$. ${}^{\mathrm{UAV}_b}\mathbf{x}_n^{\mathrm{i}}$ is the current pose of the node (before the propagation of the correction). ${}^{\mathrm{UAV}_b}\mathbf{x'}_n^{\mathrm{i}}$ is computed starting from the corrected node ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_l^{\mathrm{i}}$ and applying the odometry backwards.

We state that the value ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_n^{\mathrm{i}}$ is in between ${}^{\mathrm{UAV}_b}\mathbf{x'}_n^{\mathrm{i}}$ and ${}^{\mathrm{UAV}_b}\mathbf{x}_n^{\mathrm{i}}$.

To compute ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_n^{\mathrm{i}}$, several solutions can be used. We decided to use covariance intersection (with a coefficient of 0.5) that expresses the fact that ${}^{\mathrm{UAV}_b}\hat{\mathbf{x}}_n^{\mathrm{i}}$ should be affected more by the most accurate node and less by the less accurate node.

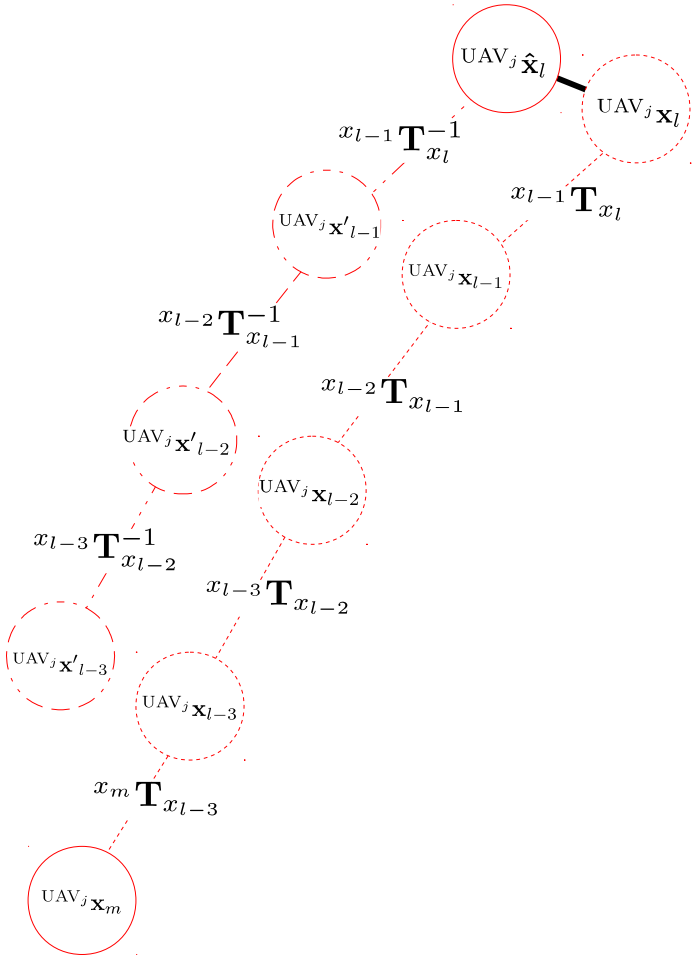


Figure 6.16: Definition of the bounds for the propagation.

$$\mathbf{P}^{-1}_{\text{UAV}_b\hat{\mathbf{x}}^{\text{i}}_n} = \mathbf{P}^{-1}_{\text{UAV}_b\mathbf{x'}^{\text{i}}_n} + \mathbf{P}^{-1}_{\text{UAV}_b\mathbf{x}^{\text{i}}_n} \tag{6.9}$$

$$^{\text{UAV}_b}\hat{\mathbf{x}}^{\text{i}}_n = \mathbf{P}_{\text{UAV}_b\hat{\mathbf{x}}^{\text{i}}_n}(\mathbf{P}_{\text{UAV}_b\mathbf{x'}^{\text{i}}_n}\ ^{\text{UAV}_b}\mathbf{x'}^{\text{i}}_n + \mathbf{P}_{\text{UAV}_b\mathbf{x}^{\text{i}}_n}\ ^{\text{UAV}_b}\mathbf{x}^{\text{i}}_n) \tag{6.10}$$

The new quaternion computed is then normalized in order to ensure that it lies on the unit sphere.

The covariance $\mathbf{P}_{\text{UAV}_b\mathbf{x'}^{\text{i}}_n}$ is computed as follows

$$\mathbf{P}_{\text{UAV}_b\mathbf{x'}^{\text{i}}_n} = \mathbf{P}_{\text{UAV}_b\mathbf{x'}^{\text{i}}_{n+1}} + \mathbf{P}_{n\mathbf{T}^{\text{i}}_{n+1}} \tag{6.11}$$

$\mathbf{P}_{n\mathbf{T}^{\text{i}}_{n+1}}$ is the covariance associated to the transformation to move from $^{\text{UAV}_b}\mathbf{x'}^{\text{i}}_{n+1}$ to $^{\text{UAV}_b}\mathbf{x'}^{\text{i}}_n$

Another method could be to use a non-linear solver such as Ceres in order to minimize the following residuals

$$\mathbf{r}_{mn}({}^{\text{uav}_b}\mathbf{p}_m, {}^{\text{uav}_b}\mathbf{q}_m, {}^{\text{uav}_b}\mathbf{p}_n, {}^{\text{uav}_b}\mathbf{q}_n) = \begin{bmatrix} \mathbf{R}({}^{\text{uav}_b}\mathbf{q}_m)^{-1}({}^{\text{uav}_b}\mathbf{p}_n - {}^{\text{uav}_b}\mathbf{p}_m) - {}^{m}\hat{\mathbf{p}}_{mn} \\ ({}^{\text{uav}_b}\mathbf{q}_n - {}^{\text{uav}_b}\mathbf{q}_m) - {}^{m}\hat{\mathbf{q}}_{mn} \end{bmatrix} \tag{6.12}$$

with

$${}^{i}\hat{\mathbf{p}}_{mn} = \hat{\mathbf{R}}^{-1}({}^{\text{uav}_b}\hat{\mathbf{p}}_n - {}^{\text{uav}_b}\hat{\mathbf{p}}_m) \tag{6.13}$$

$${}^{i}\hat{\mathbf{q}}_{mn} = {}^{\text{uav}_b}\hat{\mathbf{q}}_n - {}^{\text{uav}_b}\hat{\mathbf{q}}_m \tag{6.14}$$

6.5.3 Effect on the trajectory forwards

The existing values of the existing nodes forwards to the coincidence are updated using the new value of the node involved in the loop closures. Starting from the corrected node, the odometry measurements are successively applied to update the pose state of each node as well as its covariance matrix.

$${}^{\text{UAV}_j}\hat{\mathbf{x}}^{\text{i}}_{l+1} = {}^{l+1}\mathbf{T}^{\text{i}}_{l} \cdot {}^{\text{UAV}_j}\hat{\mathbf{x}}^{\text{i}}_{l} \tag{6.15}$$

$$\mathbf{P}_{{}^{\text{UAV}_j}\hat{\mathbf{x}}^{\text{i}}_{l+1}} = \mathbf{P}_{{}^{\text{UAV}_j}\hat{\mathbf{x}}^{\text{i}}_{l}} + \mathbf{P}_{{}^{l}\mathbf{T}^{\text{i}}_{l+1}} \tag{6.16}$$

6.5.4 The roles of the coincidences

The coincidences are used to solve two distinct problems. On one hand, they are used to find the transformation from one UAV coordinate frame to another UAV coordinate frame ${}^{\text{UAV}_a}\mathbf{T}_{\text{UAV}_b}$. Any process that involves at least two different UAVs requires to know this transformation. On the other hand, coincidences are used as additional information in the system to improve the localization estimation of the UAV of the fleet.

6.6 Data size for network load estimation

The network communication is not discussed in this thesis, though we provide an overview of the data load required by the M-SLAM system.

Each time a new keyframe is elected by one of the UAV of the fleet, the following data are transmitted:

- The UAV identifier,
- The current keyframe identifier and/or timestamps,
- The current state vector: a 7-by-1 vector,
- The covariance matrix associated with the state vector: a 7-by-7 matrix,
- The list of descriptors and keypoints and the keyframe image if additional feature extraction is required,
- The 3-D point cloud associated with the keyframe (depth estimation of the features).

The M-SLAM system gather the data broad casted by the fleet and process the input data to search for coincidences and then, update the estimate of the fleet trajectory from each UAV point of view.

6.7 Further work

Several ways of improvement for our back-end are under consideration:

- As mentioned in Section 6.5.2, a nonlinear solver for minimizing the residuals such as Ceres could improve the benefits brought by the coincidences,
- The propagation of uncertainty and the algebra regarding the matrix covariance propagation, particularly when we use the transformation between

several local coordinate frames to express the measurements, require a particular focus, the Lie theory provides interesting mathematical tools that could be applied to solve this question [SDA18] [ZGS18],

- As described in Section 2.2.4, most approaches use either a global or a local Bundle Adjustment scheme to optimize the re projection error, due to our focus on the Ceres solver, we did not try this kind of optimization in the M-SLAM algorithm, it is likely that the use of Bundle Adjustment could significantly improve the location estimates considering the addition of coincidences, this task remains for future work and experimentation.

6.8 Conclusion

We presented the back-end of the M-SLAM algorithm. We discussed deeply the differences between the traditional loop-closure and their extended version for multi-UAV systems: The coincidences. In addition, we provided a method to incorporate coincidences into the location estimates using a Kalman filtering-like method (or covariance intersection). New effects on the UAV location were underlined with regard to single-UAV systems. Finally, several approaches can be considered to improve parts of the M-SLAM system, they are referenced in Section 6.7 and remain for future research.

CHAPTER SEVEN

M-SLAM EXPERIMENTAL RESULTS: TOWARD TECHNOLOGICAL SYSTEM-OF-SYSTEMS

7.1 Introduction

In this chapter, we present the experimental results obtained with the M-SLAM algorithm. We demonstrate and discuss several significant points

- The effect of the order of coincidences,
- The effect on the location estimates using measurements from the ground truth for the coincidences (ideal case) in the M-SLAM algorithm compared with a single-robot SLAM algorithm,
- The effect on the location estimates using real noisy measurements in the M-SLAM algorithm compared with single-robot SLAM algorithm,
- The accuracy of the location of the fleet for each UAV (information that is not available in single-robot SLAM algorithm).

7.2 Presentation format of the results

Most of the following plots compare the error on a pose parameter for each UAV's keyframe (the keyframes can be seen as samples of the UAV trajectories). The coin-

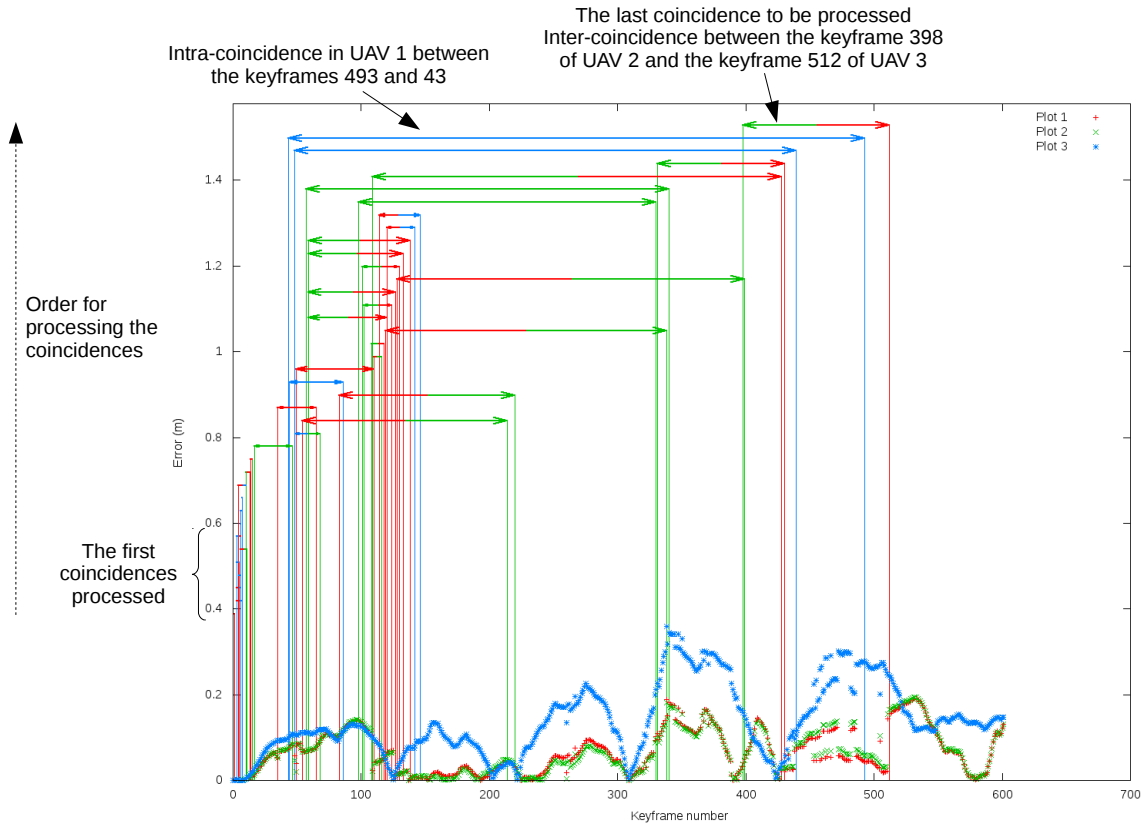


Figure 7.1: An example of plot with explanatory notes.

cidences that occurred during the experimentation are represented with horizontal arrows. The color of the arrows represent the UAVs involved. The order for processing the coincidences is indicated in a bottom-up order. The lowest horizontal arrow represents the first coincidence processed while the highest horizontal arrow represents the latest coincidence processed. The following experimentations were performed using a fleet of three UAVs. For the horizontal arrows, the red color is associated with the first UAV, the green color is associated with the second UAV and the blue color is associated with the third UAV. The Figure 7.1 provides an example with notes about how the plots can be read.

Each UAV navigates independently from the others. We used the EuRoC dataset to emulate our fleet of UAVs: The first UAV used the sensor measurements provided by the sequence V_01, the second UAV used the sensor measurements provided by the sequence V_02 and the third UAV used the sensor measurements provided by the sequence V_03. In order to provide the reader a better overview

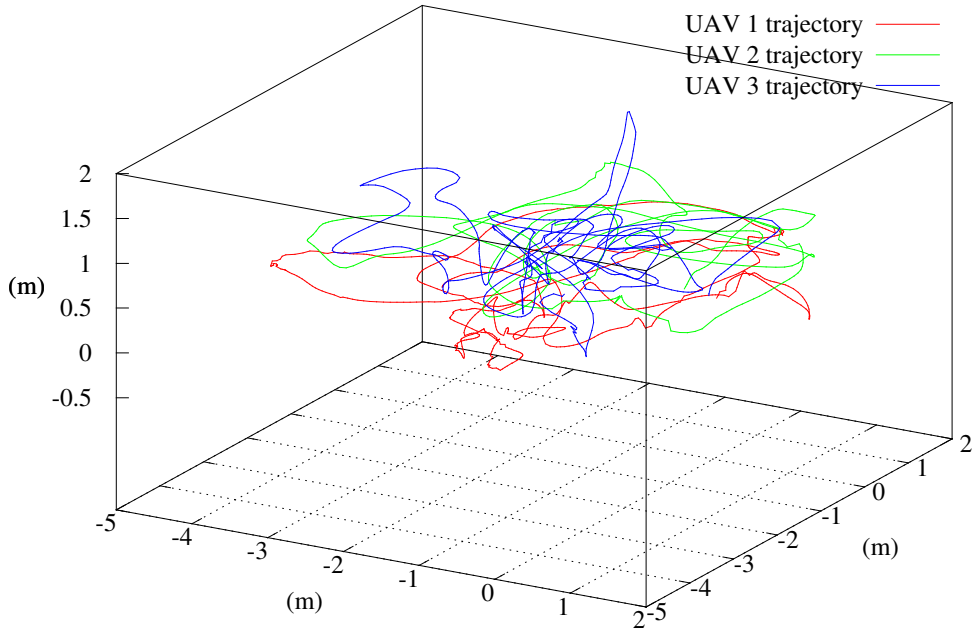


Figure 7.2: Overview of the UAV trajectories

of the experiment, the Figure 7.2 displays the trajectory of each UAV in the UAV 1 coordinate frame.

We denote the x, the first component of the translation vector, y, the second component of the translation vector, and z the third component of the translation vector. The error on the rotation is expressed using Euler angles, the yaw angles is denoted ϕ, the pitch angle is denoted θ, and the roll angle is denoted ψ.

7.3 Effect of the order of the coincidences

As discussed in Section 6.4.2, in multi-UAV systems, it is interesting to process the coincidences using a different order than the chronological order. Changing the order for processing the coincidences can have significant effects on the locations. We tried two sorting criteria to process the coincidences:

1. Processing the coincidences from the most significant correction to the less significant correction,

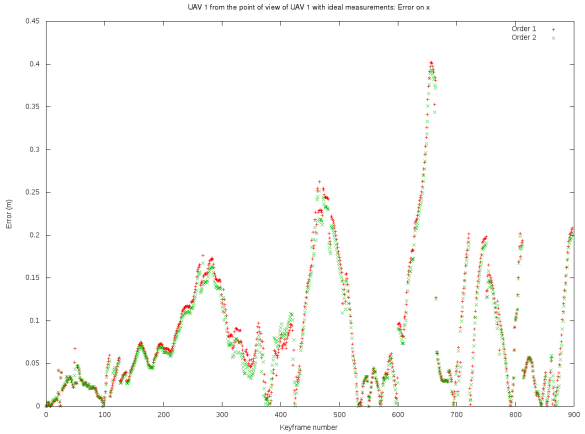
2. Processing the coincidences from the earliest correction of the trajectories to the latest effect on the trajectories.

We compared the error in position by sorting the coincidences using either the first or the second criteria. We also compared the errors using ground truth measurements for the coincidences and real noisy measurements for the coincidences.

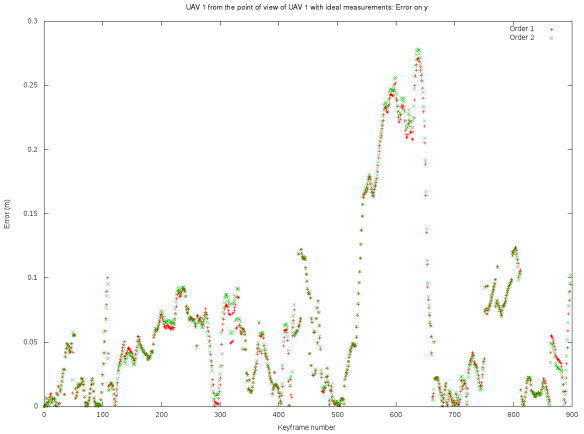
The measurement associated with a coincidence is the transformation ${}^{j}\mathbf{T}_{k}^{\mathrm{i}}$, the transformation between the inertial coordinate frames paired with the j^{th} keyframe (or pose) of the UAV$_a$ and the k^{th} keyframe (or pose) of the UAV$_b$. We explored the processing of the coincidences using two sources for the coincidence measurements: Either the ground truth measurement from the motion capture system Vicon or the measurement from the front-end computed through the method described in Chapter 5. The ground truth measurements are determined using the timestamps of the keyframes involved in the coincidences. The main benefit of using ground truth measurement is to assess the outcomes of the M-SLAM system when the coincidence measurements are very accurate (the Vicon system provides position with a precision below the millimeter range). We also evaluated the outcomes of the M-SLAM system using only the sensors placed on board the UAVs as described in Section 1.2: the inertial measurements and the front monocular camera imagery measured on board the UAVs.

Figures 7.3a, 7.3b and 7.3c display the differences in position error in UAV 1 trajectory regarding the sorting criterion using the ground truth measurements (ideal case) for the coincidences from the UAV 1 point of view. Figures 7.4a, 7.4b and 7.4c display the error on the trajectory of UAV 2 and Figures 7.5a, 7.5b and 7.5c display the error on the trajectory of UAV 3.

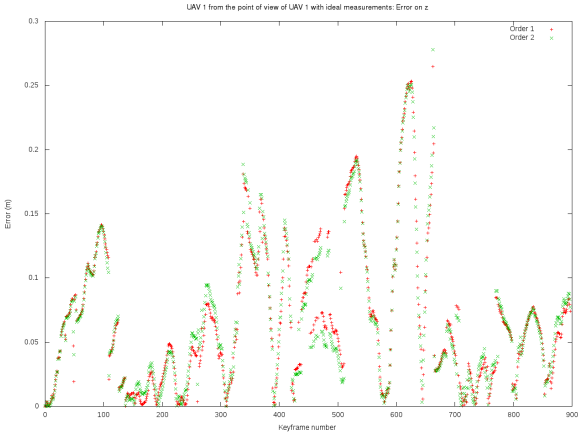
Figures 7.3a, 7.3b, 7.3c, 7.4a, 7.4b, 7.4c, 7.5a, 7.5b and 7.5c show that with ideal measurements for the coincidences, the effect of the order for processing the coincidences is barely visible. Table 7.1 confirms this conclusion as the differences on the RMSE on the position are significantly small.



(a) Error on x.

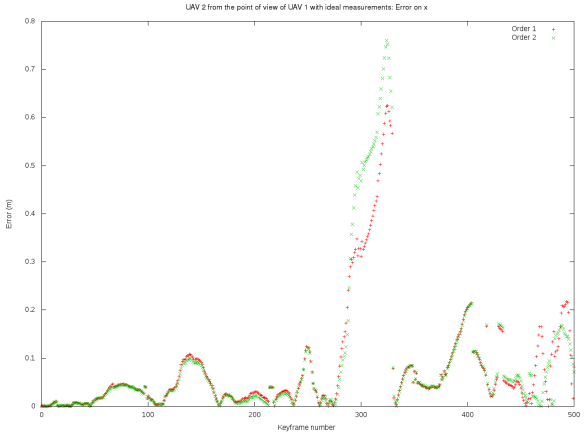


(b) Error on y.

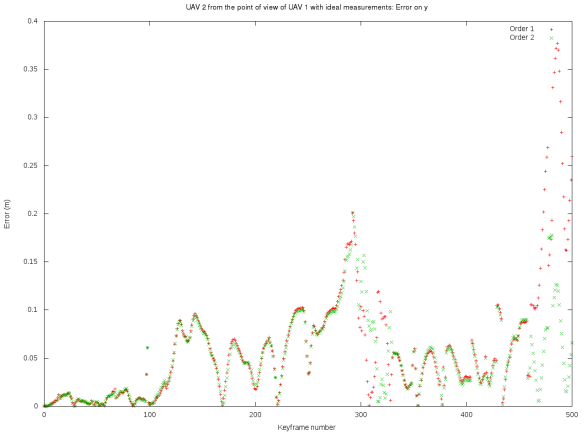


(c) Error on z.

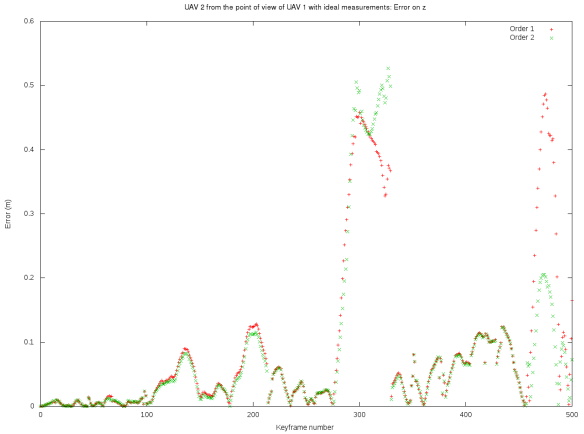
Figure 7.3: Comparison between two different order criterion in UAV 1 trajectory using ground truth measurements.



(a) Error on x.

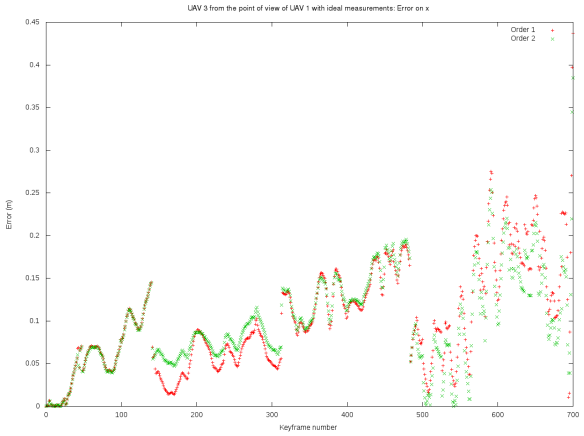


(b) Error on y.

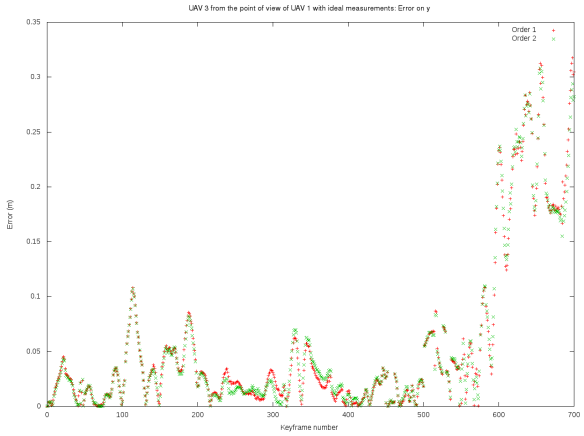


(c) Error on z.

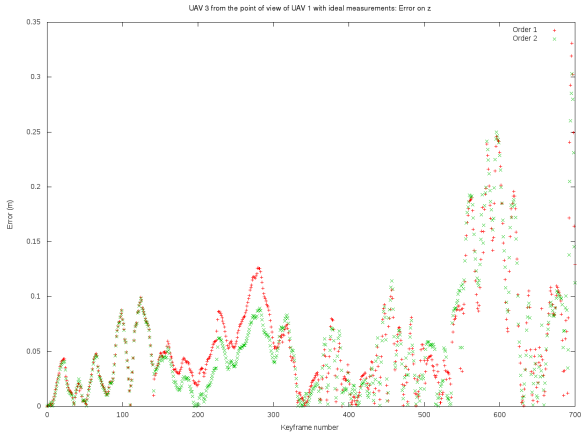
Figure 7.4: Comparison between two different order criterion in UAV 2 trajectory using ground truth measurements.



(a) Error on x.



(b) Error on y.



(c) Error on z.

Figure 7.5: Comparison between two different order criterion in UAV 3 trajectory using ground truth measurements.

RMSE	Criterion 1	Criterion 2	Absolute difference
UAV 1	0.172 m	0.169 m	0.003 m
UAV 2	0.227 m	0.234 m	0.007 m
UAV 3	0.171 m	0.165 m	0.006 m

Table 7.1: RMSE on the position regarding the order criterion for processing the coincidences using ground truth measurements.

RMSE	Criterion 1	Criterion 2	Absolute difference
UAV 1	0.672 m	0.590 m	0.082 m
UAV 2	0.376 m	0.552 m	0.176 m
UAV 3	0.245 m	0.357 m	0.112 m

Table 7.2: RMSE on the position regarding the order criterion for processing the coincidences using the front-end measurements.

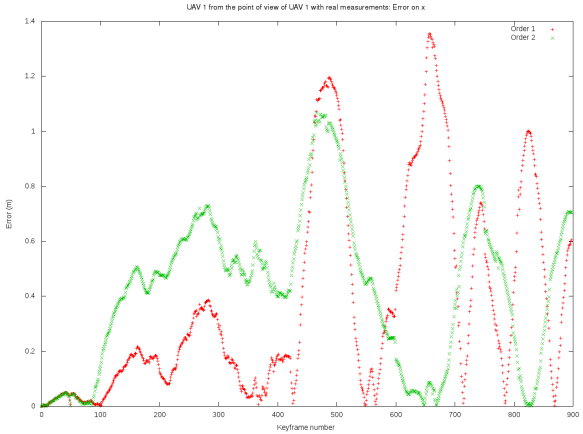
Similarly, we display the error on positions of the trajectories of UAV 1 (Figures 7.6a, 7.6b and 7.6c), UAV 2 (Figures 7.7a, 7.7b and 7.7c) and UAV 3 (Figures 7.8a, 7.8b and 7.8c) regarding the sorting criterion using the front-end measurements (real case) for the coincidences from the UAV 1 point of view.

When the front-end measurements are used, the order for processing the coincidences make a significant difference on the error on the UAV's position as shown in Figures 7.6a, 7.6b, 7.6c, 7.7a, 7.7b, 7.7c, 7.8a, 7.8b and 7.8c. Table 7.2 confirms this conclusion with a difference in RMSE regarding the order criterion a hundred times the difference in RMSE obtained using the ground-truth measurements.

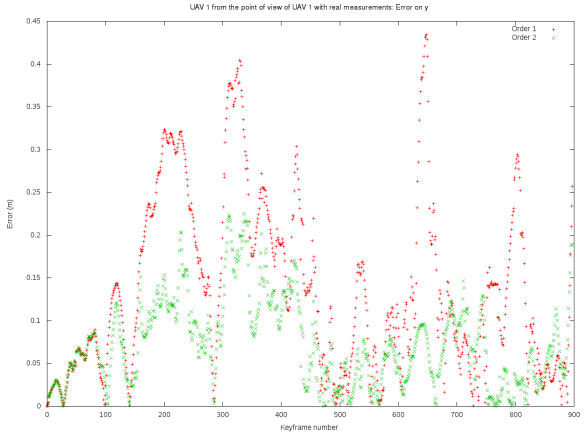
In conclusion, the order for processing the coincidences does not have a significantly effect on the UAV's position error if the measurements of the coincidences are precise enough (below the millimeters of error for the position). If the coincidence measurements are less accurate, the order for processing produces significant change in the location estimate of the fleet, therefore, the criterion to sort the coincidences must be chosen and studied carefully.

7.4 Ideal case, single-UAV versus multi-UAV

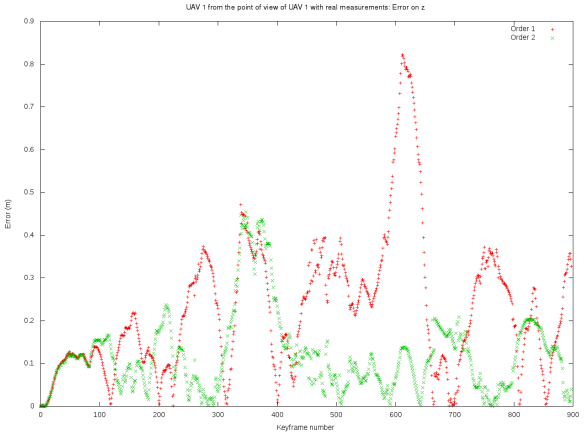
In this section, we compare the location estimate of the UAVs using either a single-robot system or our M-SLAM system (multi-robot).



(a) Error on x.

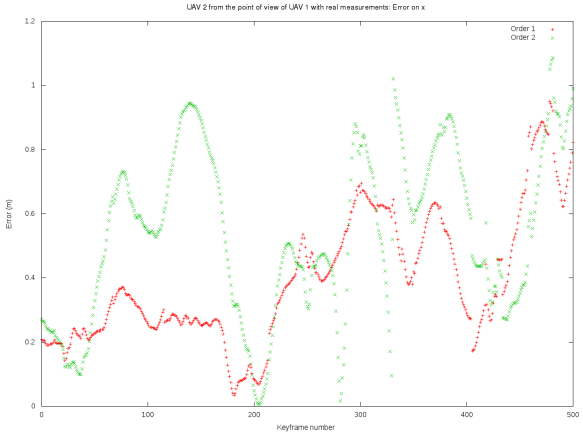


(b) Error on y.

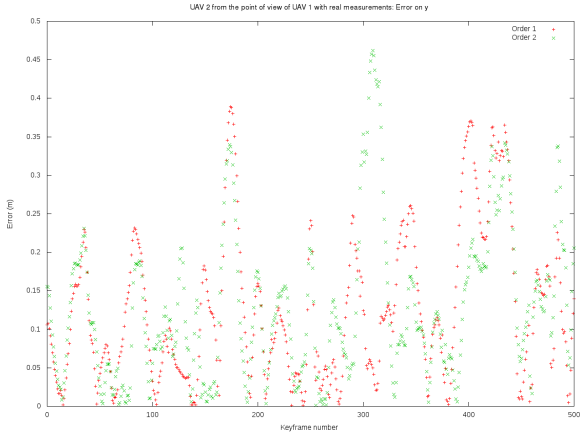


(c) Error on z.

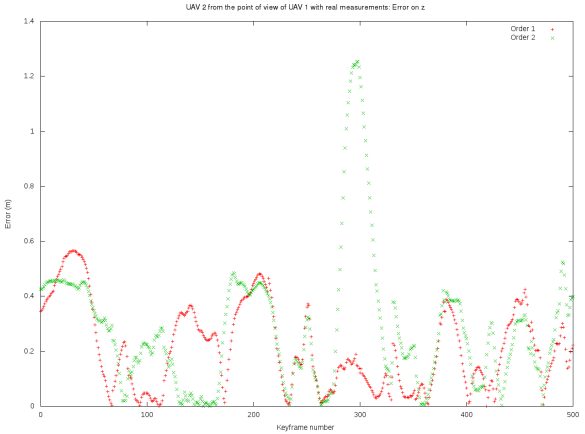
Figure 7.6: Comparison between two different order criterion in UAV 1 trajectory using the front-end measurements.



(a) Error on x.

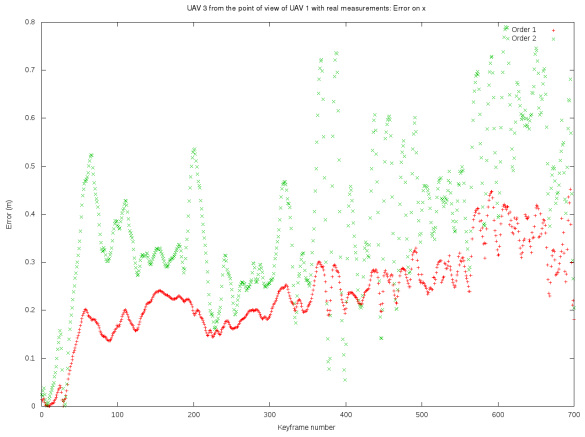


(b) Error on y.

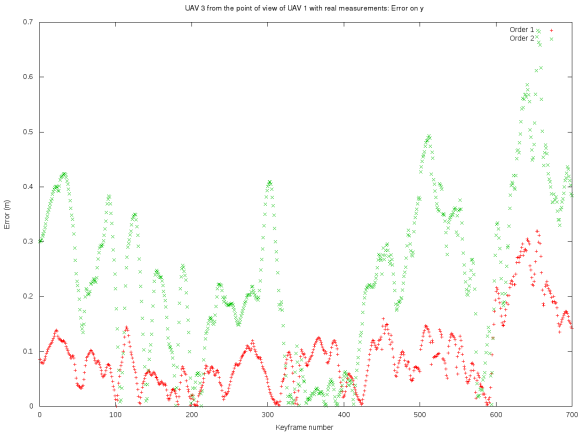


(c) Error on z.

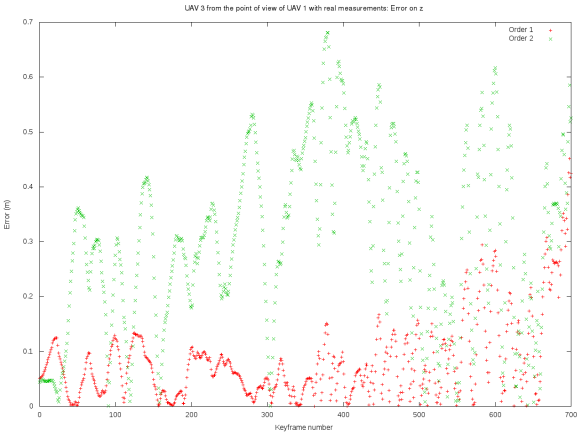
Figure 7.7: Comparison between two different order criterion in UAV 2 trajectory using the front-end measurements.



(a) Error on x.



(b) Error on y.



(c) Error on z.

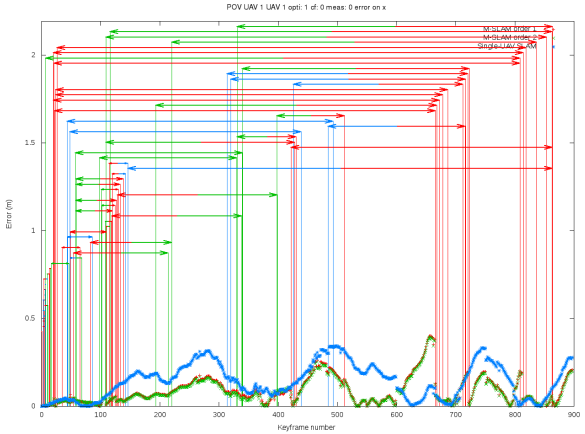
Figure 7.8: Comparison between two different order criterion in UAV 3 trajectory using the front-end measurements.

In the first place, we use the ground truth values for setting the coordinate frames and the coincidences measurements. Therefore, we use precise measurements for the transformations ${}^{\mathrm{UAV}_a}\mathbf{T}_{\mathrm{UAV}_b}$ and ${}^{j}\mathbf{T}^{c}_{k}$ for each coincidence.

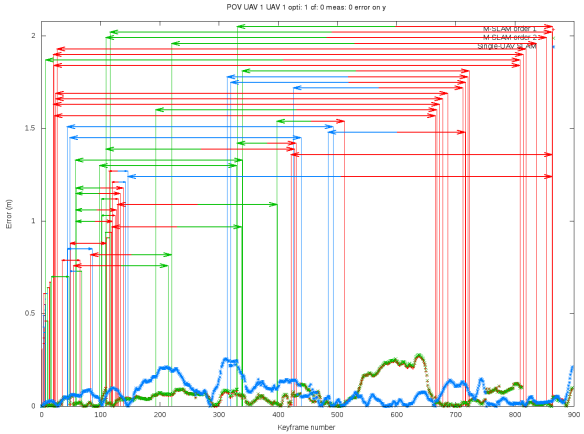
As single-robot system provides the UAV's location in different coordinate frames, in order to be able to compare a single-robot SLAM with the M-SLAM output, we considered each UAV's location from it's own point of view: We compare the location estimate of UAV 1 from the UAV 1 point of view, the location of UAV 2 from the UAV 2 point of view and the location of UAV 3 from the UAV 3 point of view. Figures 7.9a, 7.9b, 7.9c, 7.9d, 7.9e and 7.9f display the absolute error on x, y, z, ϕ, θ and ψ for each keyframe of UAV 1 trajectory from the UAV 1 point of view. Similarly, Figures 7.10a, 7.10b, 7.10c, 7.10d, 7.10e and 7.10f display the absolute error on UAV 2 trajectory and Figures 7.11a, 7.11b, 7.11c, 7.11d, 7.11e and 7.11f display the absolute error on the poses of UAV 3.

Using the M-SLAM system, the positions of the UAVs are improved though peaks in error can be observed. The results for orientation are more unstable, and should be monitored carefully as they can create instability in the system. Using our multi-UAV system, we managed to improve the location estimates in comparison with a single-UAV system. However, our system is not satisfactory for the orientation estimation. An interesting development would be to include an INS-based system like in [KHS17] to process the inertial measurements in between each keyframes, at least for the orientation part. The small distances in between consecutive keyframes would prevent any significant drifts.

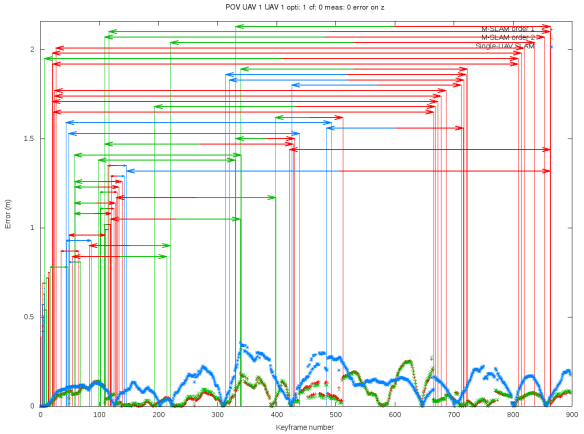
Tables 7.3, 7.4 and 7.5 summarize the plot by displaying the average error the UAV poses. The results obtained show that for the orientation the M-SLAM system is outperforms by the single-UAV system. The estimation of orientations are the main weakness of the M-SLAM system and will be the most important part to improve in further work. Moreover, the improvement in orientation estimation should produce a significant improvement in the position estimation. However, improvements can be observed in the position of UAVs using the M-SLAM system.



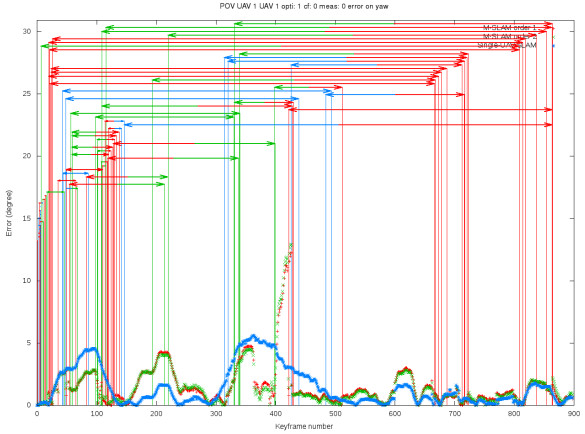
(a) UAV 1 error on x.



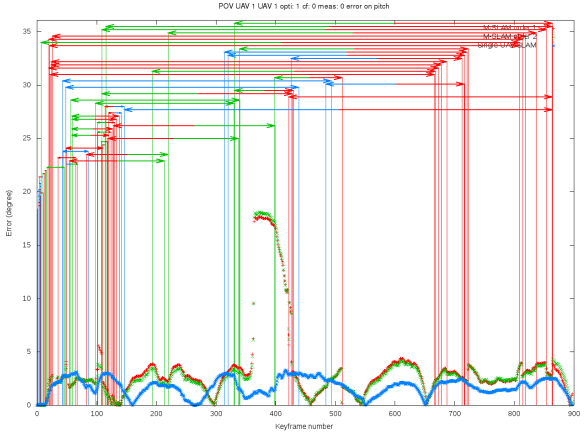
(b) UAV 1 error on y.



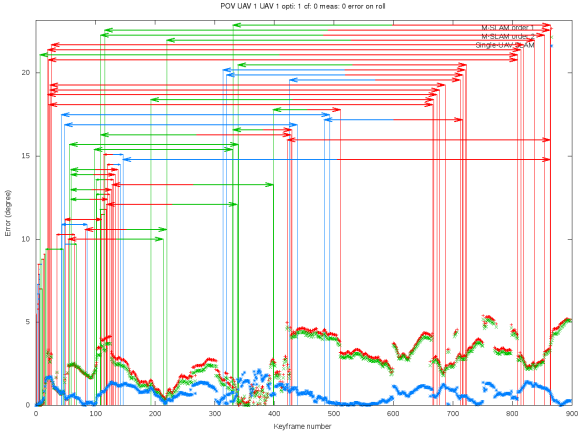
(c) UAV 1 error on z.



(d) UAV 1 error on ϕ.

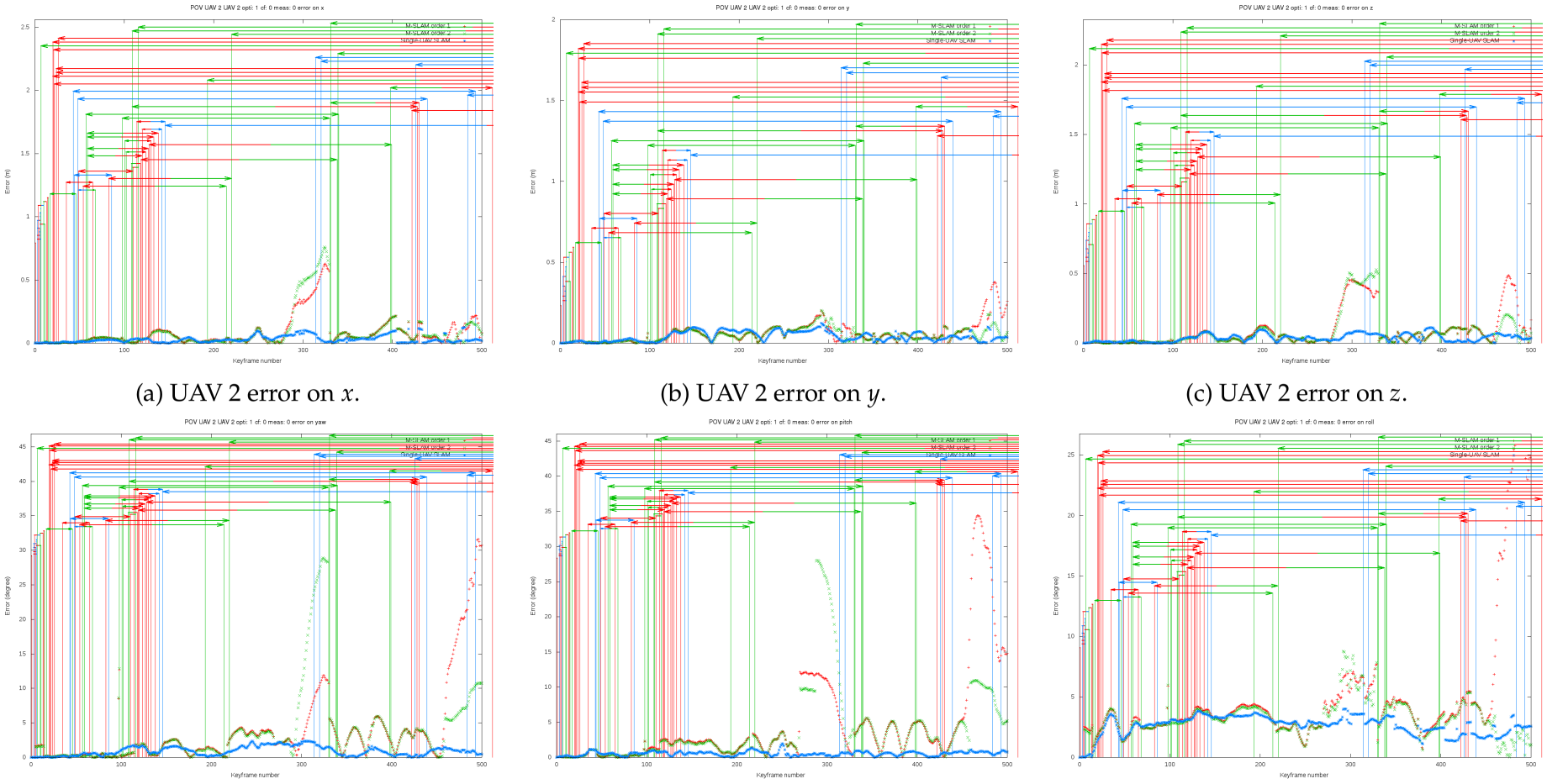


(e) UAV 1 error on θ.



(f) UAV 1 error on ψ.

Figure 7.9: Comparison between the M-SLAM and the single-UAV SLAM systems using ground truth measurements for UAV 1.



(a) UAV 2 error on x.

(b) UAV 2 error on y.

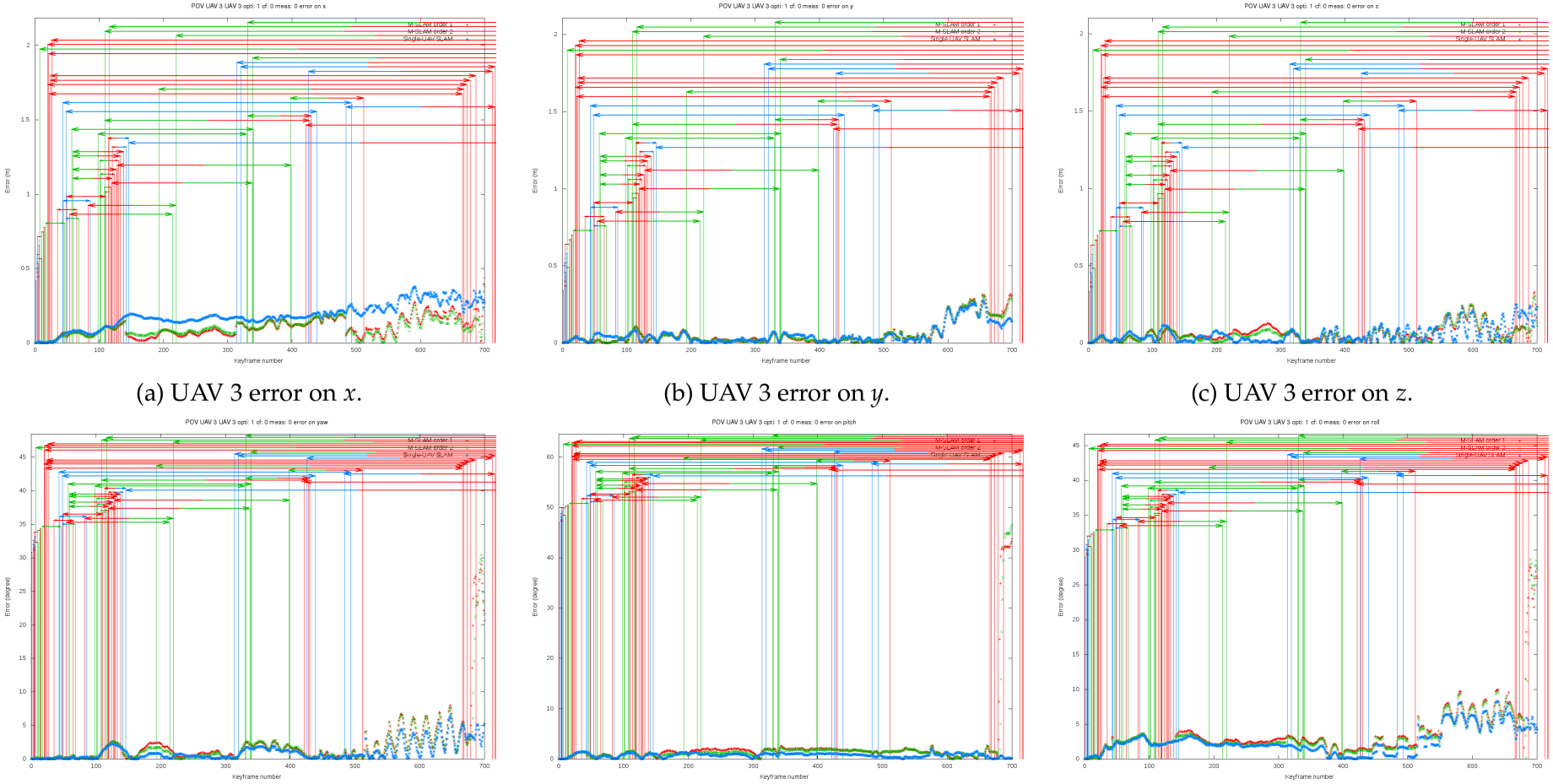
(c) UAV 2 error on z.

(d) UAV 2 error on ϕ.

(e) UAV 2 error on θ.

(f) UAV 2 error on ψ.

Figure 7.10: Comparison between the M-SLAM and the single-UAV SLAM systems using ground truth measurements for UAV 2.



(a) UAV 3 error on x.

(b) UAV 3 error on y.

(c) UAV 3 error on z.

(d) UAV 3 error on ϕ.

(e) UAV 3 error on θ.

(f) UAV 3 error on ψ.

Figure 7.11: Comparison between the M-SLAM and the single-UAV SLAM systems using ground truth measurements for UAV 3.

	Single-UAV system					
Error	x	y	z	ϕ	θ	ψ
UAV 1	$0.144m$	$0.075m$	$0.132m$	1.435°	1.738°	0.728°
UAV 2	$0.016m$	$0.023m$	$0.020m$	0.513°	0.300°	1.471°
UAV 3	$0.142m$	$0.042m$	$0.043m$	0.844°	0.583°	1.961°

Table 7.3: Average error on the pose of the UAVs using a single-UAV system.

	M-SLAM order criterion 1					
Error	x	y	z	ϕ	θ	ψ
UAV 1	$0.089m$	$0.063m$	$0.067m$	1.486°	3.324°	2.707°
UAV 2	$0.046m$	$0.034m$	$0.050m$	2.156°	2.649°	2.830°
UAV 3	$0.080m$	$0.044m$	$0.047m$	1.727°	1.946°	2.963°

Table 7.4: Average error on the pose of the UAVs using the M-SLAM system with order criterion 1 using ground truth measurements.

The position of UAV 1 is significantly improved using the M-SLAM system, the UAV 3 position is also improved or at least similar to the results produced by the single-UAV system. The case of UAV 2 is slightly different, the single-UAV system outperforms the M-SLAM system. However, it is interesting to observe that the error on the UAV 2 position given by the single-UAV system are dramatically small. Obviously, the UAV 2 trajectory is a favorable case for the single-UAV system while the M-SLAM system do not manage to benefit as much as the single-UAV system of the measurements taken by the UAV 2.

As displayed, peaks can appear sometimes in the error. Figure 7.12 outlines this phenomenon.

To understand how the peaks appeared, Figures 7.13a, 7.13b, 7.13c, 7.13d, 7.13e, 7.13f, 7.14a, 7.14b,7.14c, 7.14d, 7.14e and 7.14f provide an overview of the change in

	M-SLAM order criterion 2					
Error	x	y	z	ϕ	θ	ψ
UAV 1	$0.084m$	$0.063m$	$0.068m$	1.466°	3.220°	2.513°
UAV 2	$0.050m$	$0.028m$	$0.044m$	2.044°	2.193°	1.789°
UAV 3	$0.077m$	$0.044m$	$0.044m$	1.597°	1.722°	2.750°

Table 7.5: Average error on the pose of the UAVs using the M-SLAM system with order criterion 2 using ground truth measurements.

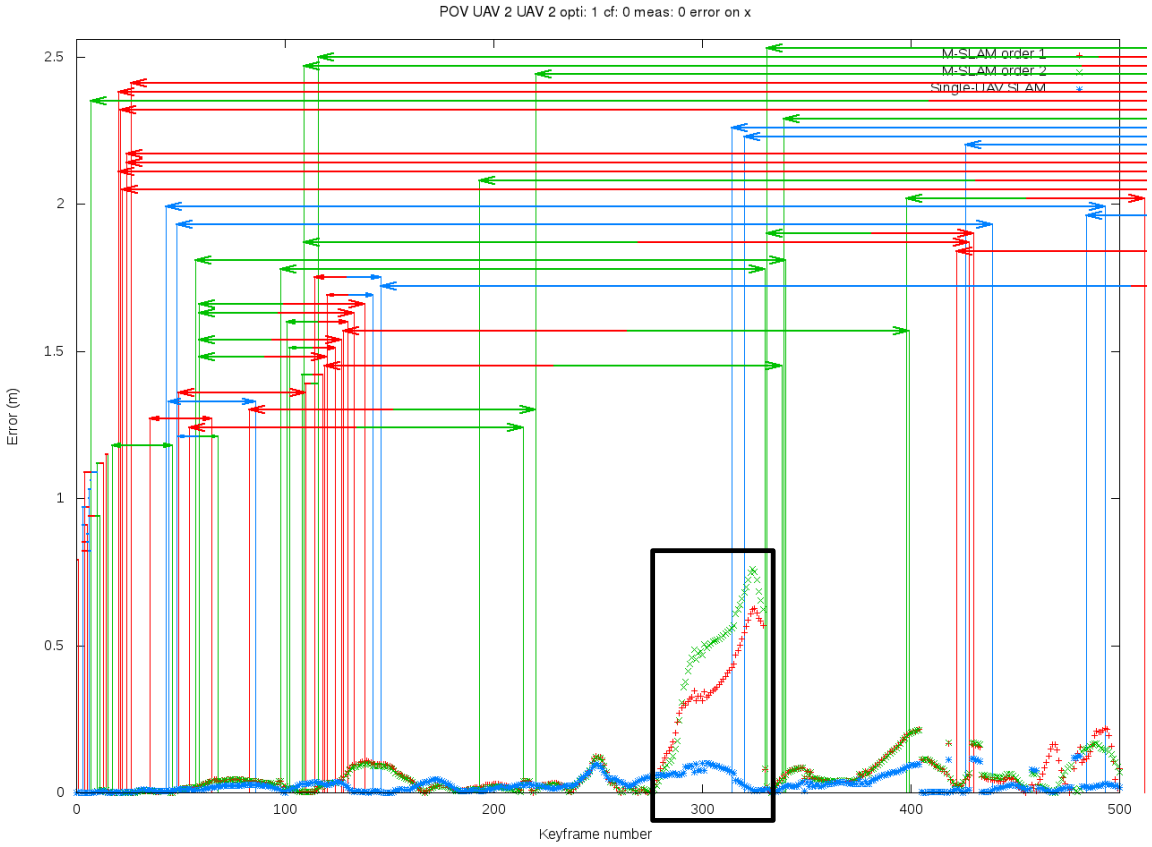
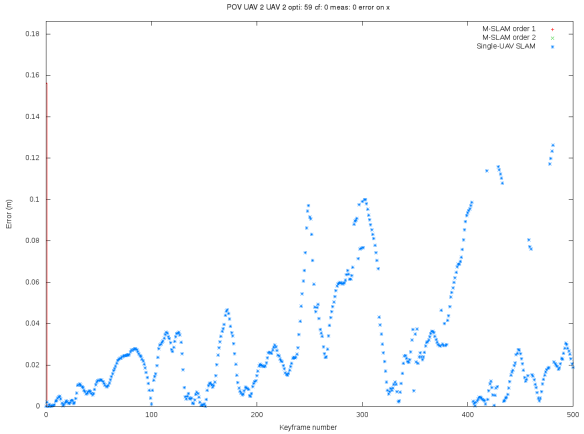


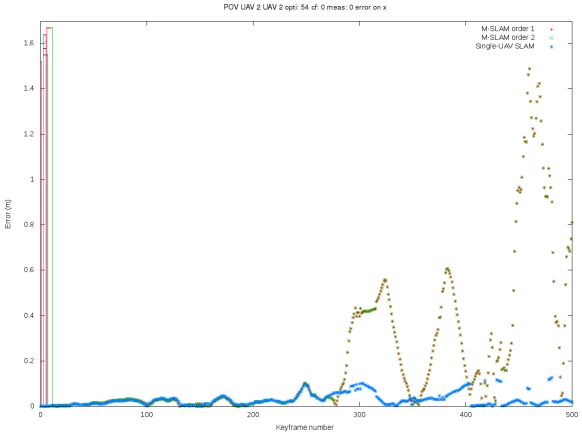
Figure 7.12: An example of peak in the error in the UAV 2 trajectory

the error on x throughout the processing of the coincidences.

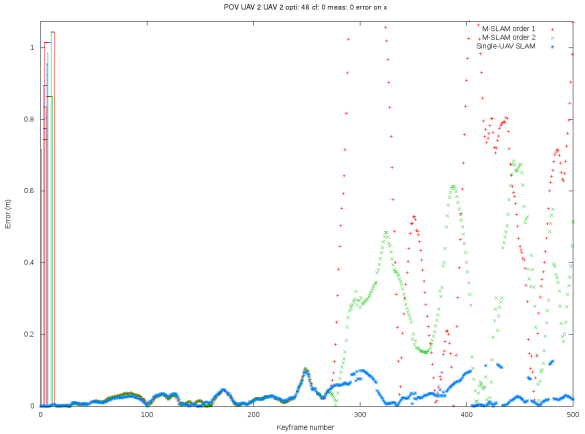
We discovered that those peaks are due to the forward effect (Section 6.4.2). When the coincidences are processed, the trajectory is re-built based on the new value of the corrected node using the relative motion measurements. This strategy is successful for a medium time range (about 250 keyframes) but when the trajectory becomes longer, it can create instabilities (the peak observed in Figure 7.12). To counter-balance those instabilities, two strategies could be considered: Either to trigger coincidences around this moment (it could be part of the navigation and decision algorithm for the multi-UAV exploration), or include a feed-back loop in the system to update the output from the odometry measurements. Those peaks can also be partially explained by the values of the covariance matrices. As VINS-Mono do not output the covariance matrices related to the poses and keyframes, we estimate the value using the ground truth which lead to an underestimation of the covariances. As the covariance values are underestimated, the backward



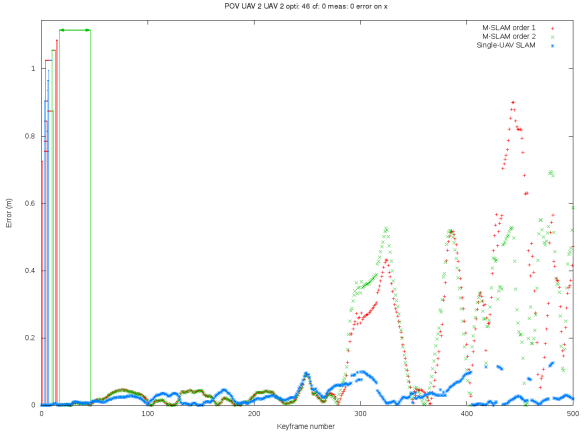
(a) 1 coincidence processed.



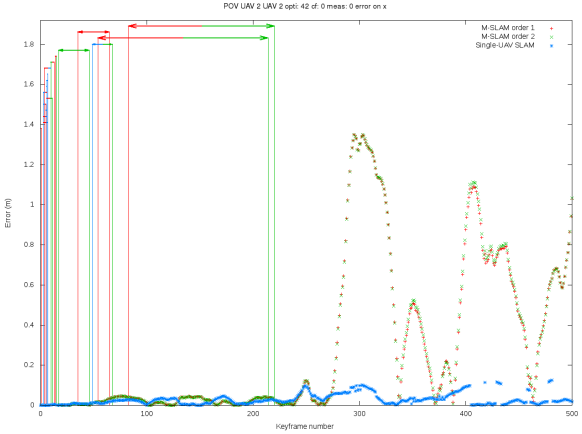
(b) 6 coincidences processed.



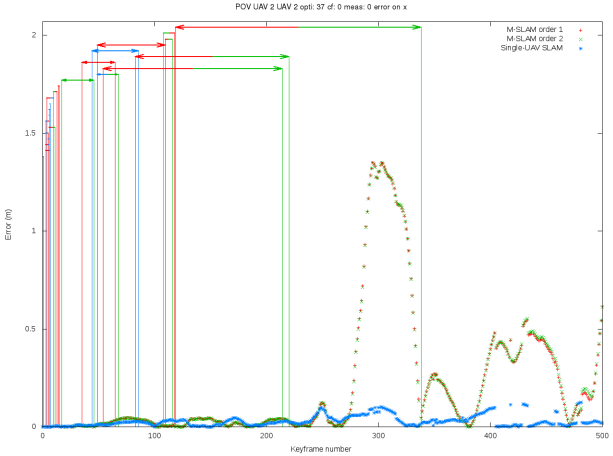
(c) 12 coincidences processed.



(d) 14 coincidences processed.

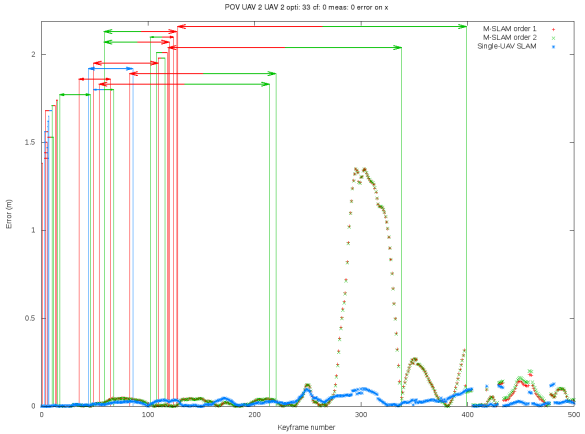


(e) 18 coincidences processed.

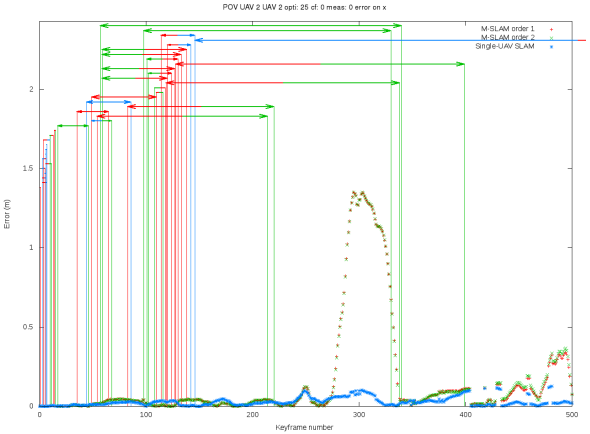


(f) 23 coincidences processed.

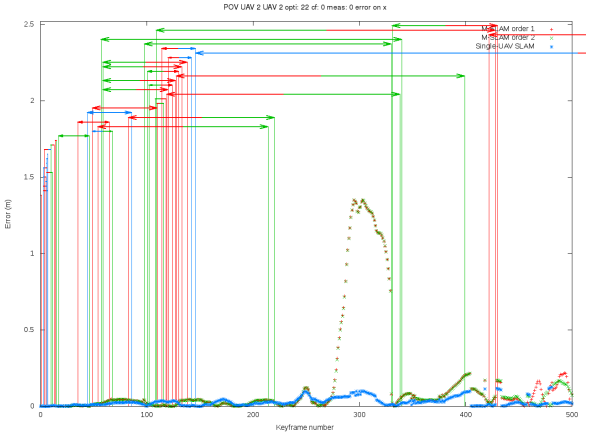
Figure 7.13: Change in the error on x throughout the processing of coincidences (1).



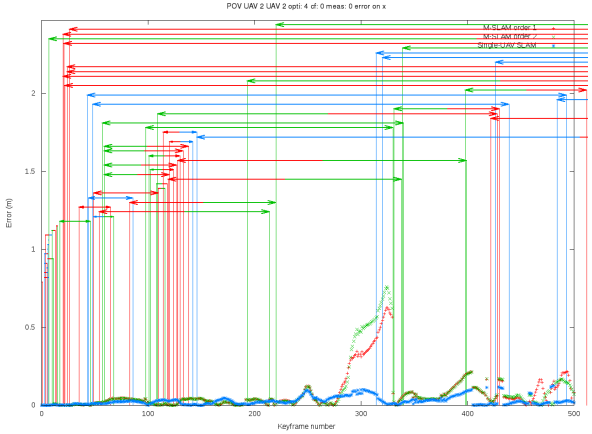
(a) 27 coincidence processed.



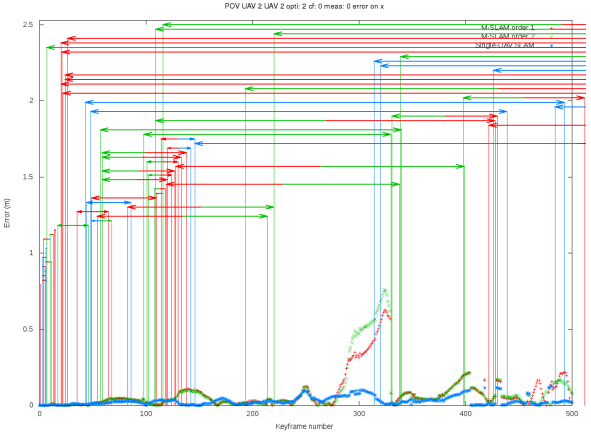
(b) 35 coincidences processed.



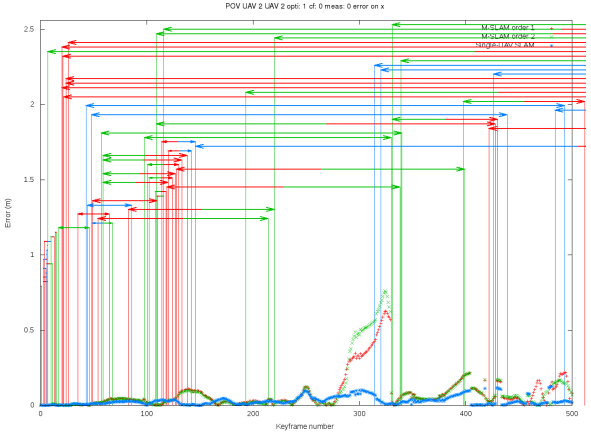
(c) 38 coincidences processed.



(d) 56 coincidences processed.



(e) 58 coincidences processed.



(f) 59 coincidences processed.

Figure 7.14: Change in the error on x throughout the processing of coincidences (2).

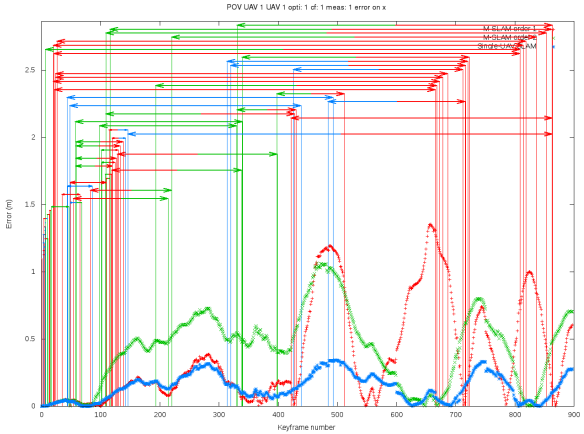
propagation cannot smooth fully the trajectory and therefore, correct entirely the peaks due to the forward effect.

7.5 Real measurements, single-UAV versus multi-UAV

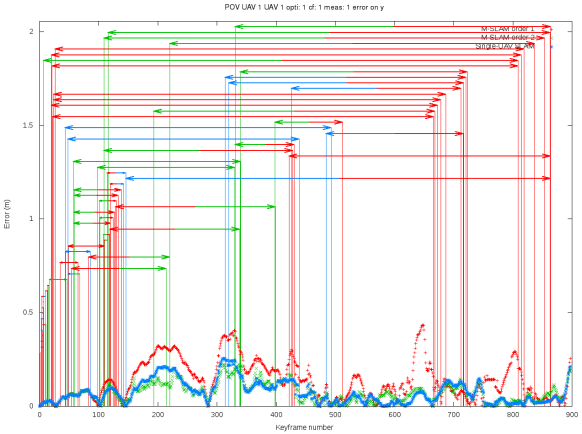
The Section 7.4 provides an overview of the best performance we can expect from our M-SLAM approach using precise measurements from a motion capture system Vicon for the coincidences; but in a real system, the ground truth cannot be used as measurements. In this section, we provide an overview of the results we obtained from the M-SLAM system in comparison to the single-UAV SLAM when we use the measurements computed by the front-end (Chapter 5). Both the coordinate frame transformations ${}^{\mathrm{UAV}_i}\mathbf{T}_{\mathrm{UAV}_j}$ and the transformations ${}^{\mathbf{cam}_k}\mathbf{T}_{\mathbf{cam}_l}$ for each coincidence are computed from real measurements recorded on board the UAVs.

Figures 7.15a, 7.15b, 7.15c, 7.15d, 7.15e and 7.15f display the absolute error on x, y, z, ϕ, θ and ψ for each keyframe of UAV 1 trajectory from the UAV 1 point of view. Similarly, Figures 7.16a, 7.16b, 7.16c, 7.16d, 7.16e and 7.16f display the absolute error on UAV 2 trajectory and Figures 7.17a, 7.17b, 7.17c, 7.17d, 7.17e and 7.17f display the absolute error on the poses of UAV 3.

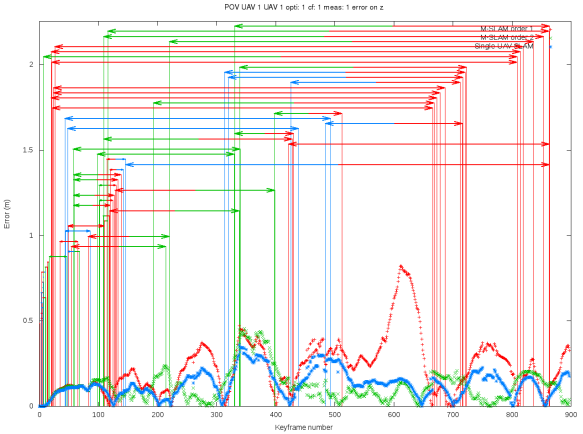
As expected, using the front-end measurements that are less precise than the ground truth measurements for the coincidences, we obtain greater error. Quite surprisingly, except for the x component, the position error is similar to the results obtained using the ground truth measurements. Regarding the orientation part, the estimate are significantly less precise in average than the results obtained with ground truth which confirm that the orientation is more sensitive to the processing of the coincidences than the position. However, we would like to underline that in this experiment there is a critical additional piece of information that is estimated: The transformation between the UAV's coordinate frames. As a matter of fact, the transformations ${}^{\mathrm{UAV}_a}\mathbf{T}_{\mathrm{UAV}_b}$ are not considered in a single-UAV system (because there is only one UAV) and are measured very precisely when we use the ground



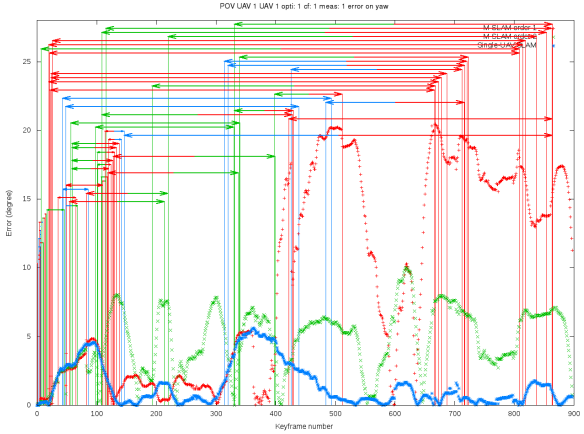
(a) UAV 1 error on x



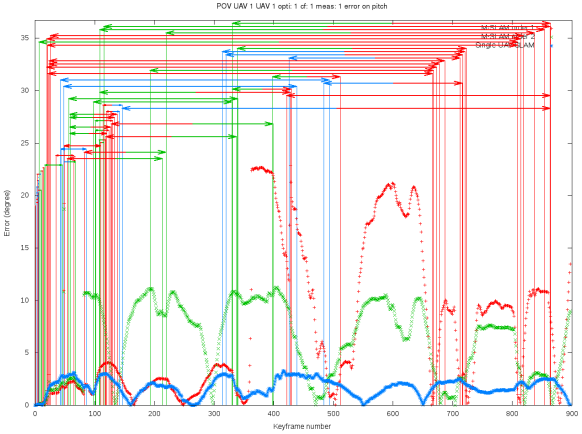
(b) UAV 1 error on y



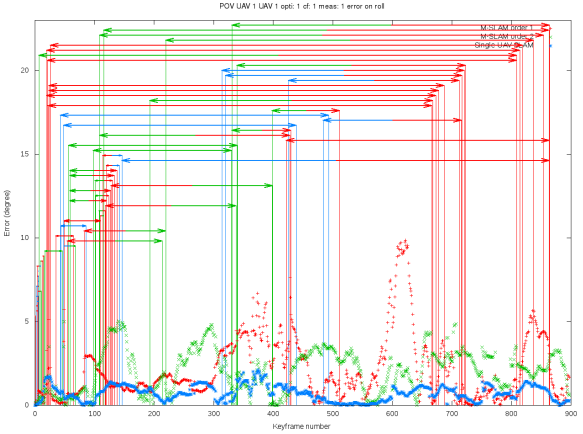
(c) UAV 1 error on z



(d) UAV 1 error on ϕ

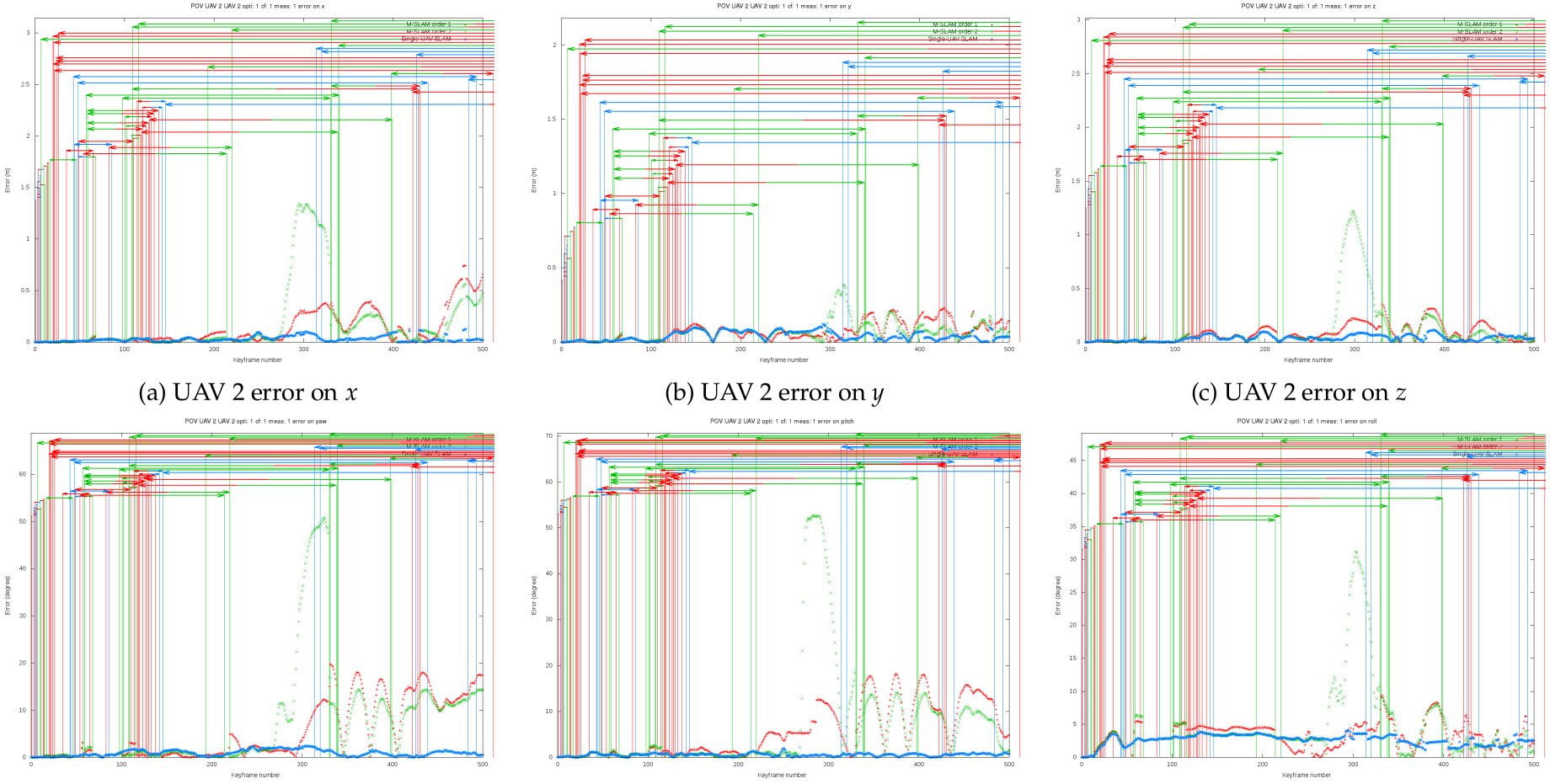


(e) UAV 1 error on θ



(f) UAV 1 error on ψ

Figure 7.15: Comparison between the M-SLAM and the single-UAV SLAM systems using the front-end measurements for UAV 1.



(a) UAV 2 error on x

(b) UAV 2 error on y

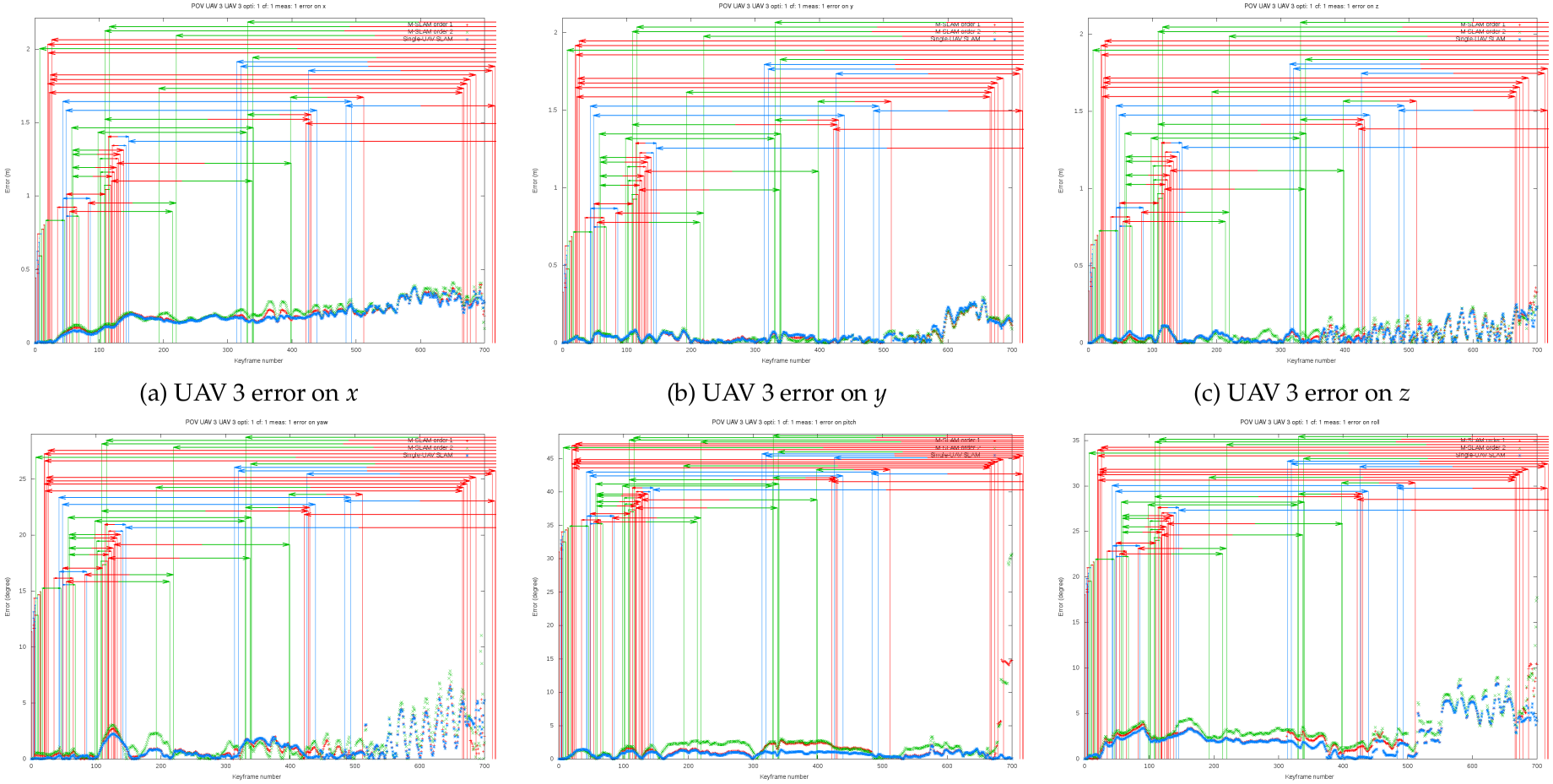
(c) UAV 2 error on z

(d) UAV 2 error on ϕ

(e) UAV 2 error on θ

(f) UAV 2 error on ψ

Figure 7.16: Comparison between the M-SLAM and the single-UAV SLAM systems using the front-end measurements for UAV 2.



(a) UAV 3 error on x

(b) UAV 3 error on y

(c) UAV 3 error on z

(d) UAV 3 error on ϕ

(e) UAV 3 error on θ

(f) UAV 3 error on ψ

Figure 7.17: Comparison between the M-SLAM and the single-UAV SLAM systems using the front-end measurements for UAV 3.

	M-SLAM order criterion 1					
Error	x	y	z	ϕ	θ	ψ
UAV 1	$0.378m$	$0.137m$	$0.227m$	9.138°	7.471°	2.162°
UAV 2	$0.073m$	$0.040m$	$0.044m$	3.025°	3.055°	1.850°
UAV 3	$0.148m$	$0.039m$	$0.043m$	0.870°	1.146°	2.273°

Table 7.6: Average error on the pose of the UAVs using the M-SLAM system with order criterion 1 using the front-end measurements.

	M-SLAM order criterion 2					
Error	x	y	z	ϕ	θ	ψ
UAV 1	$0.424m$	$0.073m$	$0.121m$	4.714°	6.223°	2.093°
UAV 2	$0.104m$	$0.039m$	$0.068m$	4.164°	3.884°	2.622°
UAV 3	$0.161m$	$0.040m$	$0.047m$	1.124°	1.576°	2.681°

Table 7.7: Average error on the pose of the UAVs using the M-SLAM system with order criterion 2 using the front-end measurements.

truth measurements.

Tables 7.6 and 7.7 provide a summary of the results by displaying the average error on the poses.

Using the front-end measurements, the single-UAV outperforms the precision on the pose estimation provided by the M-SLAM system. However, there is an additional piece of information that is estimated in the M-SLAM that is required for each UAV to estimate where are the other UAVs of the fleet.

7.6 Localization of the fleet

7.6.1 Experiment goal

One of the main benefit of the M-SLAM approach is to make possible for each UAV to have an estimate of the location of the other UAV of the fleet, information that is not available in single-UAV SLAM. In this section we focus on the estimation of the trajectory of the fleet for each UAV. This part is critical for the design of distributed exploration algorithm as UAVs cannot have an effective exploration strategy without having information about the pose of the others. In this experiment, we are in

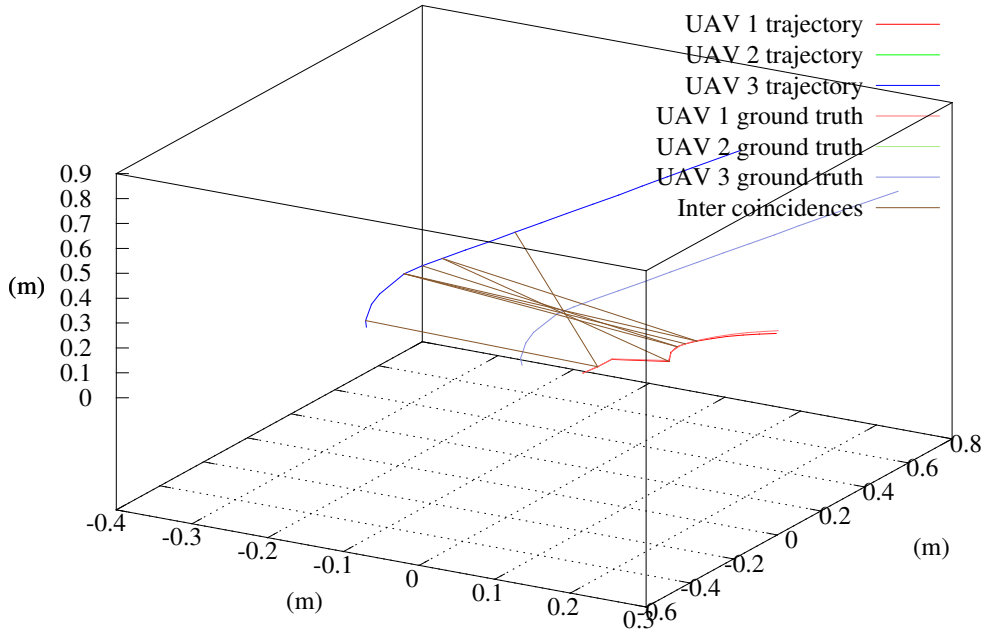


Figure 7.18: UAV 1 point of view, trajectory of the fleet (1)

the shoes of each UAV to understand what information they receive from the other via the inter-coincidence processing.

7.6.2 UAV 1 point of view

We present the estimation of the fleet trajectory from UAV 1 point of view for the first keyframes. After some keyframes, the graphs become so tangled that a static visual representation is not suitable anymore.

When the experiment begun, the UAV 1 and UAV 3 take off roughly at the same and from close position. Figure 7.18 shows what UAV 1 knows after a few keyframes: the UAV 1 pose estimates and the UAV 3 trajectory with regard to $\{\text{UAV}_1\}$. The trajectory of UAV 3 is longer as the UAV flies faster than UAV 1. The UAV 1 trajectory is very close to the ground truth because we are regarding the experiment from UAV 1 point of view, UAV 3 is further to its ground truth because the transformation ${}^{\text{UAV}_1}\mathbf{T}_{\text{UAV}_3}$ is estimated by the M-SLAM system using the detected inter-coincidences, measurements that are noisy.

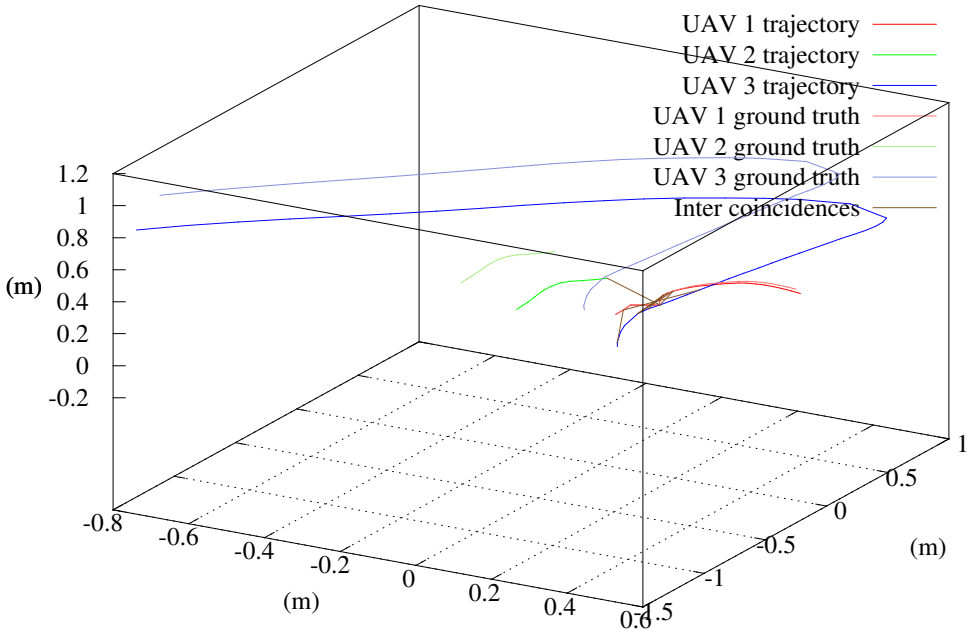


Figure 7.19: UAV 1 point of view, trajectory of the fleet (2)

UAV 2 took off after the other UAVs. Figure 7.19 illustrate the inclusion of UAV 2 into the UAV 1 pose-graph. An inter-coincidence is detected between UAV 2 and UAV 3. As UAV 3 and UAV 1 are already connected, there is an indirect link between UAV 1 and UAV 2. Using this indirect link, the transformation ${}^{\mathrm{UAV}_1}\mathbf{T}_{\mathrm{UAV}_2}$ can be estimated.

In Figure 7.20, the trajectory of UAV 2 with regard to $\{\mathrm{UAV}_1\}$ is improved by the detection of a direct inter-coincidence between UAV 1 and UAV 2. There is also the first intra-coincidence in the UAV 2 trajectory.

In Figure 7.21, the experimentation keep running and the UAV trajectories become longer with the time going. Additional coincidences are detected and processed to improve the pose estimates.

7.6.3 UAV 2 point of view

The case of UAV 2 is interesting as it takes off after UAV 1 and UAV 3.

The moment in the experiment Figure 7.18 was taken, UAV 2 has not taken

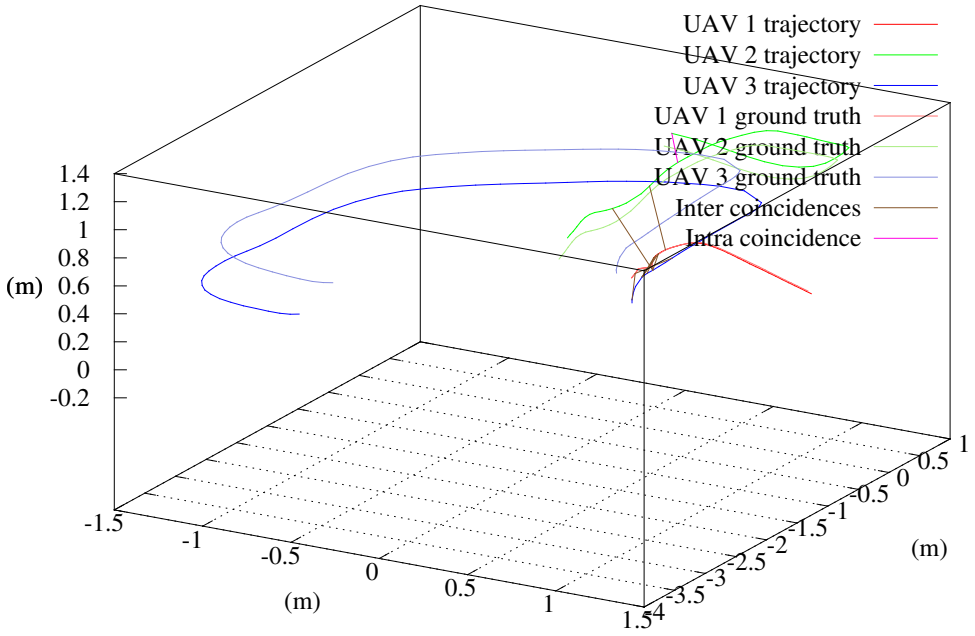


Figure 7.20: UAV 1 point of view, trajectory of the fleet (3)

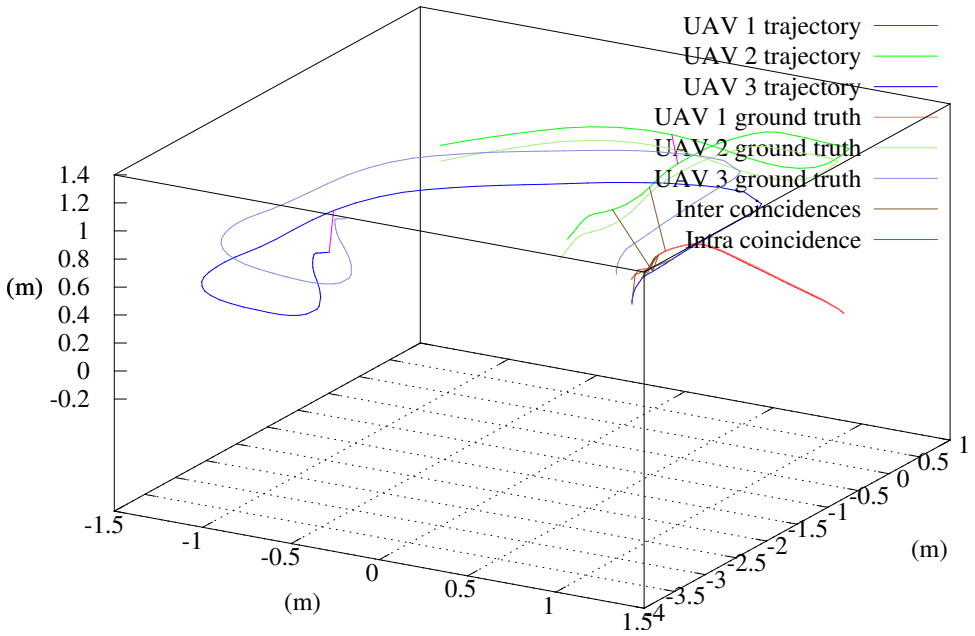


Figure 7.21: UAV 1 point of view, trajectory of the fleet (4)

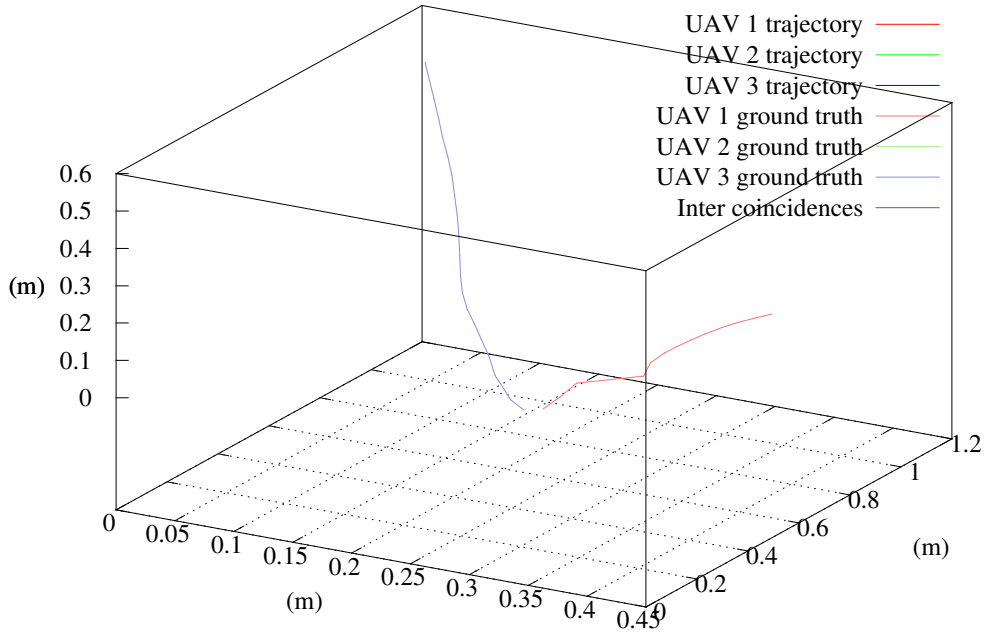


Figure 7.22: UAV 2 point of view, trajectory of the fleet (1)

off yet. Therefore, UAV 2 has no knowledge about the rest of fleet, this explains why only ground truth trajectories are plotted in Figure 7.22. UAV 1 and UAV 3 are flying following the trajectories plotted by their ground truth measurements, however, as UAV 2 has not detected coincidences, it cannot know where the UAVs are.

Similarly to UAV 1, the trajectory of UAV 2 is very close to the ground truth as this we are observing from the UAV 2 point of view. Therefore, the transformations ${}^{UAV_1}\mathbf{T}_{UAV_2}$ and ${}^{UAV_2}\mathbf{T}_{UAV_3}$ are estimated either directly (Figure 7.24) or indirectly (Figure 7.23). Figure 7.25 illustrates the experiment running with he first intra-coincidences detected.

To be consistent in the representation, the plots of the fleet trajectories from each UAV point of view (respectively Figures 7.18, 7.22 and 7.26 for the first freeze, Figures 7.19, 7.23 and 7.27 for the second freeze, Figures 7.20, 7.24 and 7.28 for the third freeze, and Figures 7.21, 7.25 and 7.29 for the fourth freeze) were all taken at the same time with the same number of coincidences processed.

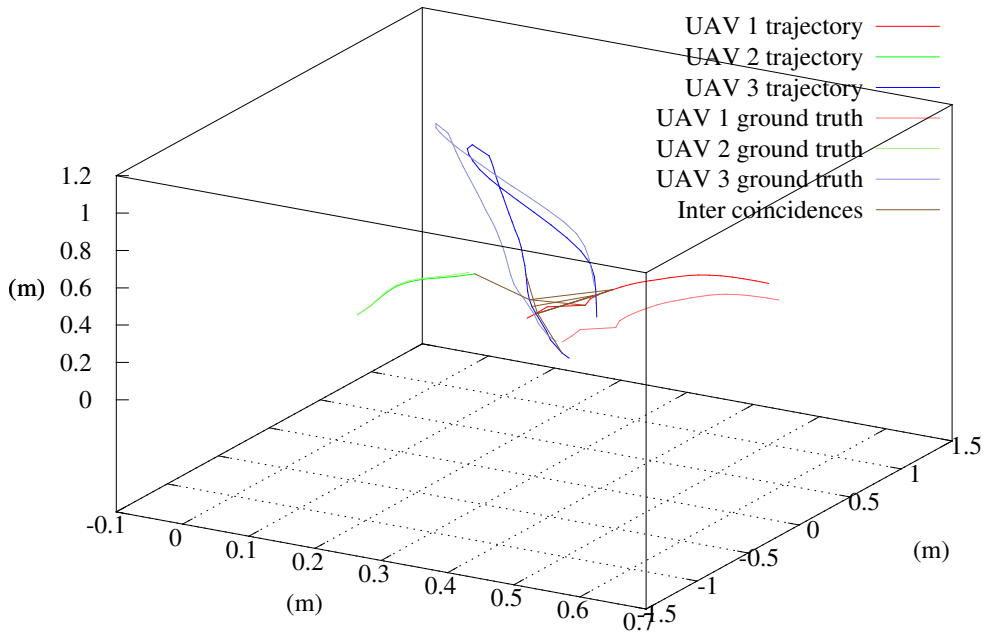


Figure 7.23: UAV 2 point of view, trajectory of the fleet (2)

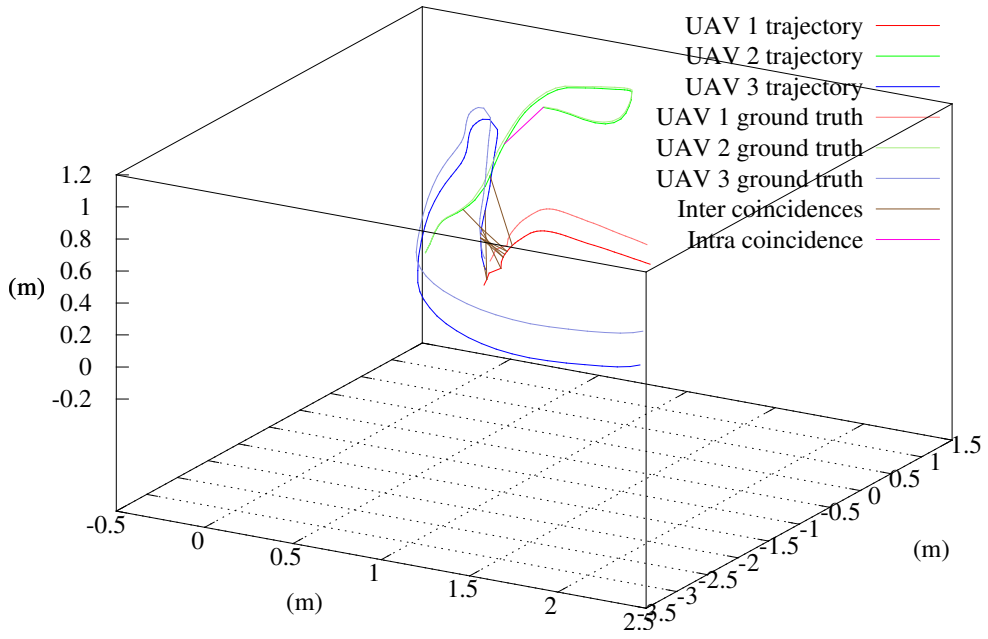


Figure 7.24: UAV 2 point of view, trajectory of the fleet (3)

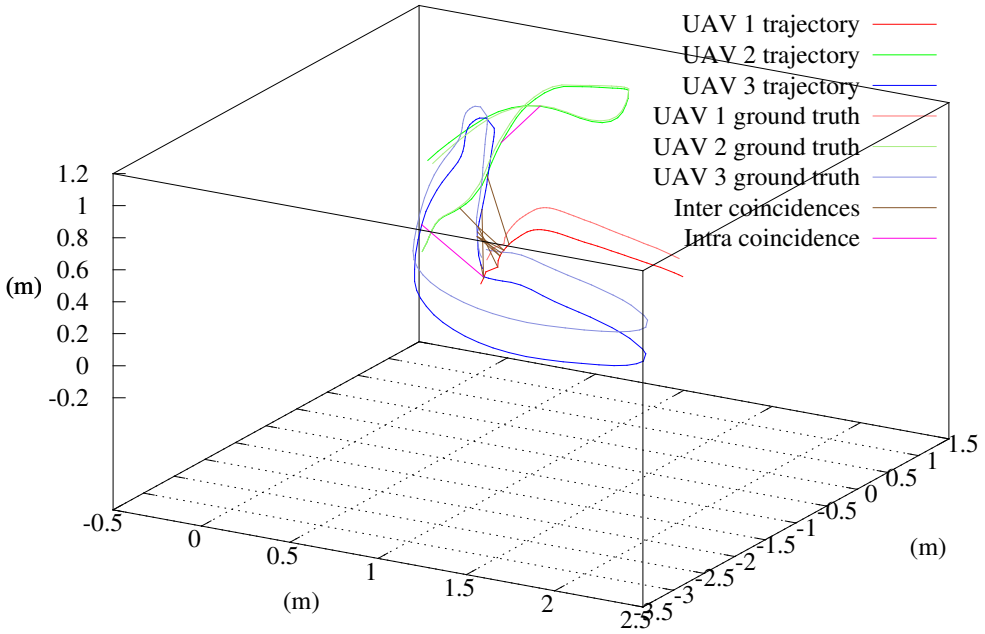


Figure 7.25: UAV 2 point of view, trajectory of the fleet (4)

7.6.4 UAV 3 point of view

Finally, we observe the scene from UAV 3 point of view. Figures 7.26, 7.27, 7.28 and 7.29 illustrate how the trajectories change with the experiment running. We confirm the previous remarks: The UAV 3 poses are precisely estimated with regard to the ground truth because we are observing from the UAV 3 point of view. At the beginning, UAV 1 did not take off and therefore does not share any direct or direct coincidences with UAV 3, that is the reason why the UAV 2 trajectory does not appear on Figure 7.26. Then, there is a good quality inter-coincidence between the UAV 2 and UAV 3 that allow UAV 3 to estimate where is UAV 2 with regard to $\{UAV_3\}$ quite precisely.

Tables 7.8 and 7.9 provide the RMSE of the fleet for each UAV point of view.

The main goal we target is fulfilled as each UAV has information about the location of the other UAVs of the fleet with regard to its own coordinate frame. However, we cannot conclude to best criterion for ranking the coincidences before

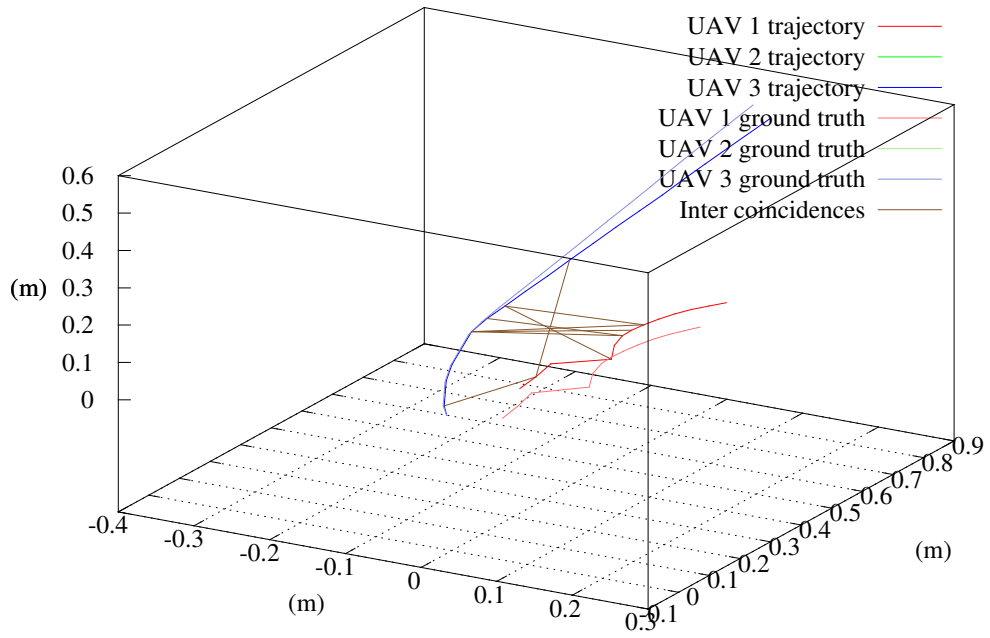


Figure 7.26: UAV 3 point of view, trajectory of the fleet (1)

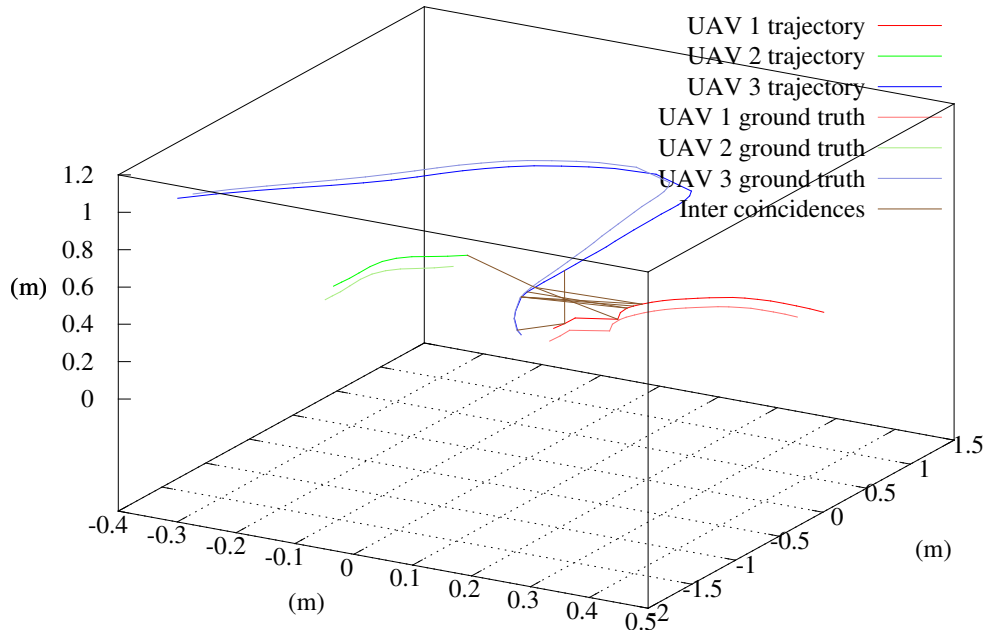


Figure 7.27: UAV 3 point of view, trajectory of the fleet (2)

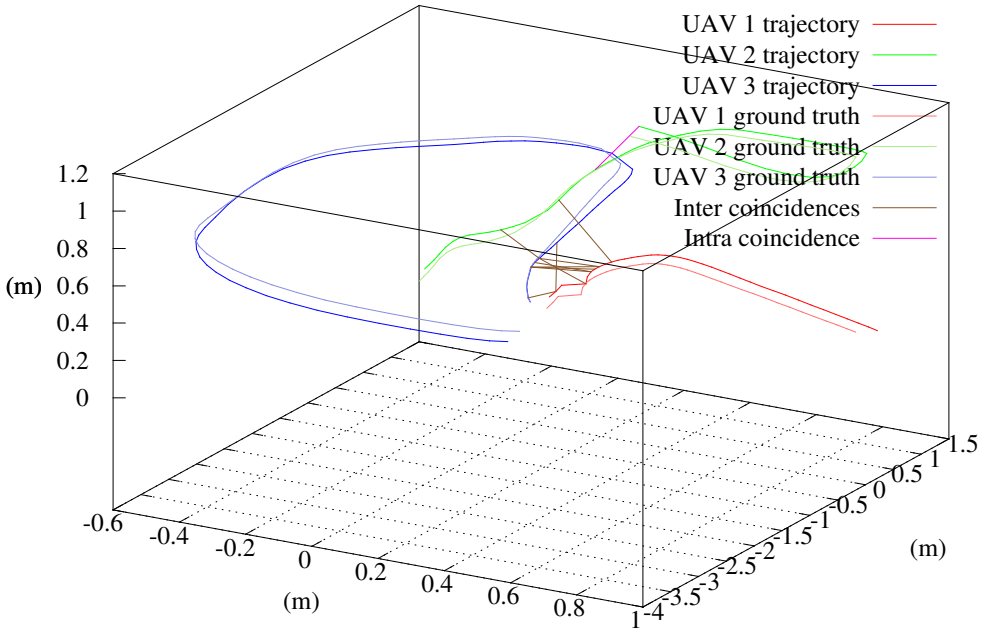


Figure 7.28: UAV 3 point of view, trajectory of the fleet (3)

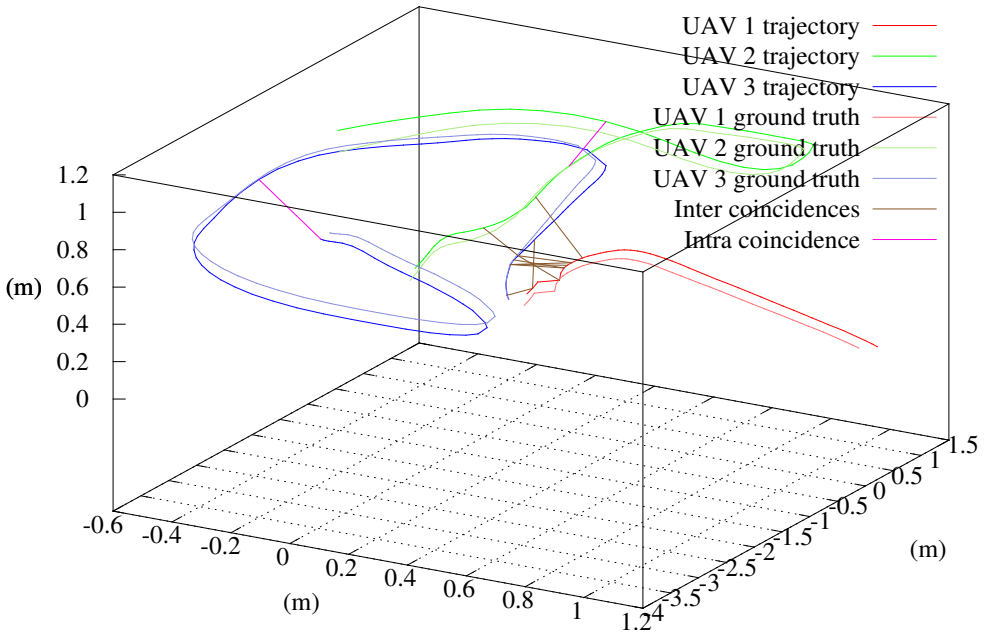


Figure 7.29: UAV 3 point of view, trajectory of the fleet (4)

	UAV 1	UAV 2	UAV 3
PoV UAV 1	0.622 m	0.537 m	0.297 m
PoV UAV 2	0.793 m	0.257 m	0.206 m
PoV UAV 3	0.602 m	0.335 m	0.234 m

Table 7.8: RMSE of the fleet using the front-end measurement and order criterion 1.

	UAV 1	UAV 2	UAV 3
PoV UAV 1	0.537 m	0.721 m	0.606 m
PoV UAV 2	0.724 m	0.476 m	0.213 m
PoV UAV 3	0.509 m	0.459 m	0.251 m

Table 7.9: RMSE of the fleet using the front-end measurement and order criterion 2.

their processing. Though, we can affirm that this order has a significant on the error of the estimates.

CHAPTER EIGHT

CONCLUDING REMARKS AND DIRECTIONS FOR FUTURE RESEARCH

8.1 Contributions

In this thesis, we presented the M-SLAM system that solves some of the critical issues related to the localization of the UAVs. The main application of the system is navigation and exploration of unknown areas by using a fleet of collaborative autonomous UAVs. We provided a distributed localization component considering the inherent challenges due to the independent operations of the autonomous UAVs: They can take off and land from any place and time without constraints.

As the M-SLAM is meant to be included into distributed exploration algorithms and is part the DIVINA challenge team, some specifications were imposed: To produce metric estimates for both the 3-D representation of the environment and the locations, by using only a front monocular camera and an IMU placed on board each UAV for the sensing part.

We proposed an extensive study about the Technological System-of-Systems about autonomous fleet of robots to define and illustrate most of the concepts we used to manage the fleet of UAVs, and about the single robot SLAM algorithms with a specific focus on monocular SLAM, monocular-inertial SLAM and real-time SLAM algorithms.

We created a novel mechanism for metric map and pose estimation in monocular

SLAM using a loosely-coupled fusion with inertial measurements. This work was published in the IEEE International Conference on Multi-sensor Fusion and Integration for Intelligent Systems (MFI) in 2017 [Spa+17].

We designed and implemented the M-SLAM system, which is capable of providing for each UAV, a location estimation of the UAVs of the fleet (including itself) with a minimal set of work hypothesis (no absolute measurements and no a priori knowledge on the positions).

We compared the results provided by the M-SLAM system to the output of a recently published inertial-monocular SLAM regarding the location of a single robot. We also discussed the estimation of the trajectory of the fleet from each UAV point of view, an information that is not available in the usual single-UAV systems.

To create the M-SLAM system, we extended the concept of loop-closure in traditional single-robot SLAM to multi-robot systems.

8.2 Further improvements in the M-SLAM system

During this work, choices had to be made in order to being able to offer a complete system. Therefore, in some components, there is a good potential for further investigation and improvement:

- In our loosely-coupled scheme, we should add a RANSAC module to improve the convergence to the best scaling coefficient,
- In the M-SLAM system, further investigation should be performed to improve the orientation estimation,
- In the M-SLAM system, additional experiments should be done with better estimated covariance matrices, this implies a follow-up work in the field of uncertainty and covariance matrices in the single-UAV SLAM used as the input,
- Find a solution to assess the uncertainty of the detected coincidences,

- Extend and compare the experiments using a fleet of five drones and a loosely-coupled inertial-monocular SLAM as an input, in addition to the current tightly-coupled inertial-monocular SLAM algorithm.

If we consider a longer road-map, we also plan additional engineering work on the ROS package for the M-SLAM that is currently a prototype. As well as testing the M-SLAM system as an input for an exploration algorithm.

8.3 Future research

2018 has been a prolific year for research publications in multi-UAV SLAM and visual-inertial systems. The incorporation of a number of published works into the M-SLAM would be beneficial for additional experiments and comparative analysis.

Many publications would be interesting to be incorporated in the M-SLAM system in order to perform additional experiments and comparisons.

The front-end of the CVI-SLAM, the OKVis and ROVIOLI algorithms and our loosely-coupled approach for metric state estimation are all of great interest to test on our front-end. A deeper study on the choice of descriptors and the filtering of false coincidence candidates could also help to add improvements in our system and save computation time.

For the back-end, several components can be studied or improved. The problem of the propagation of covariance matrices and the propagation of uncertainty more generally would require thorough investigation. The Lie theory [SDA18] can help to bring a satisfactory answer to this problem. The effect of the map merging on the localization is also an phenomenon that could be studied, moreover, now that useful frameworks, such as Maplab [Sch+18], exist. A particular attention can also be given to define an optimization process in the back-end, either minimizing residuals through a nonlinear solver such as Ceres [AM12], or using one of the form of the Bundle Adjustment, would it be global, local or pose-only optimization.

Bibliography

[18a] *The DIVINA challenge team*. Accessed on 5th of December 2018. URL: https://divina.labexms2t.fr/divina_objectives/.

[18b] *The LabeX MS2T center*. Accessed on 5th of December 2018. URL: https://www.labexms2t.fr/.

[18c] *Robot Operating System*. Accessed on 23rd of August 2018. URL: https://ros.org.

[19a] *The Google Maps API*. Accessed on 22nd of February 2019. URL: https://cloud.google.com/maps-platform/.

[19b] *GTSam framework*. Accessed on 25th of February 2019. URL: https://borg.cc.gatech.edu/.

[19c] *Matlab website*. Accessed on 25th of February 2019. URL: https://fr.mathworks.com.

[19d] *Towards Robust and Safe Autonomous Drones, presentation by D. Scaramuzza*. Accessed on 25th of February 2019. URL: https://fr.slideshare.net/SERENEWorkshop/towards-robust-and-safe-autonomous-drones.

[19e] *Open-source code of Vins-MONO*. Accessed on 22nd of February 2019. URL: https://github.com/HKUST-Aerial-Robotics/VINS-Mono.

[Ach+12] M. Achtelik, M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J. Decotignie, K. Doth, F. Fraundorfer, L. Kneip, and D. Gurdan. "Sfly: Swarm of Micro Flying Robots". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Algarve, Portugal, 2012, pp. 2649–2650.

[Alo+15] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley. "Collision Avoidance for Aerial Vehicles in Multi-Agent Scenarios". In: *Autonomous Robots* 39.1 (2015), pp. 101–121.

[AM12] S. Agarwal and K. Mierle. *Ceres Solver*. 2012.

[ANB11] P. F. Alcantarilla, J. Nuevo, and A. Bartoli. "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". In: *British Machine Vision Conference (BMVC)* 34.7 (Bristol, UK, 2011), pp. 1281–1298.

[Ang+08] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat. "Real-Time Visual Loop-Closure Detection". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Pasadena, California, USA, 2008, pp. 1842–1847.

[BF95] J. V. Burke and M. C. Ferris. "A Gauss-Newton Method for Convex Composite Optimization". In: *Mathematical Programming* 71.2 (1995), pp. 179–194.

[BTV06] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded Up Robust Features". In: *European Conference on Computer Vision (ECCV)* (Graz, Austria, 2006), pp. 404–417.

[Bur+16] 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: *The International Journal of Robotics Research* 35 (10) (2016), pp. 1157–1163.

[Cad+16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[Cal+10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. "Brief: Binary Robust Independent Elementary Features". In: *European Conference on Computer Vision (ECCV)*. Springer. Hersonissos, Heraklion, Crete, Greece, 2010, pp. 778–792.

[CC15] A. Concha and J. Civera. "DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, 2015, pp. 5686–5693.

[CKM08] R. Castle, G. Klein, and D. W. Murray. "Video-Rate Localization in Multiple Maps for Wearable Augmented Reality". In: *IEEE International Symposium on Wearable Computers (ISWC)*. Pittsburgh, USA, 2008, pp. 15–22.

[CMM15] N. Chebrolu, D. Marquez-Gamez, and P. Martinet. "Collaborative Visual SLAM Framework for a Multi-Robot System". In: *PPNIV* 2 (2015), p. 4.

[CN08] M. Cummins and P. Newman. "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance". In: *The International Journal of Robotics Research* 27.6 (2008), pp. 647–665.

[Cog+12] M. Cognetti, P. Stegagno, A. Franchi, G. Oriolo, and H. H. Bülthoff. "3-D mutual localization with anonymous bearing measurements". In: *IEEE International Conference on Robotics and Automation (ICRA)*. St Paul, USA, 2012, pp. 791–798.

[Con+16] A. Concha, G. Loianno, V. Kumar, and J. Civera. "Visual-Inertial Direct SLAM". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 1331–1338.

[Del12] F. Dellaert. *Factor Graphs and GTSam: A Hands-on Introduction*. Tech. rep. Georgia Institute of Technology, 2012.

[DS18] J. Delmerico and D. Scaramuzza. "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, Australia, 2018, pp. 2502–2509.

[EKC18] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (2018), pp. 611–625.

[ESC14] J. Engel, T. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *European Conference on Computer Vision (ECCV)*. Springer. Zurich, Switzerland, 2014, pp. 834–849.

[FB87] M. A. Fischler and R. C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Readings in Computer Vision*. Elsevier, 1987, pp. 726–740.

[For+13] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza. "Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Tokyo, Japan, 2013, pp. 3962–3970.

[For+15] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation". In: *Robotics: Science and Systems (RSS)*. Rome, Italy, 2015.

[Fra+07] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. "A Randomized Strategy for Cooperative Robot Exploration". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Roma, Italy, 2007, pp. 768–774.

[Fra+09] A. Franchi, L. Freda, G. Oriolo, and M. Vendittelli. "The Sensor-Based Random Graph Method for Cooperative Robot Exploration". In: *IEEE/ASME Transactions on Mechatronics* 14.2 (2009), pp. 163–175.

[Fra+12] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys. "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Algarve, Portugal, 2012, pp. 4557–4564.

[Gei+13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision Meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* 32 (11) (2013), pp. 1231–1237.

[Gri+10] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.

[GT12] D. Gálvez-López and J. D. Tardos. "Bags of Binary Words for Fast Place Recognition in Image Sequences". In: *IEEE Transactions on Robotics* 28.5 (2012), pp. 1188–1197.

[Guo+16] 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: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 1071–1078.

[Har95] R. I. Hartley. "In Defense of the 8-Point Algorithm". In: *IEEE International Conference on Computer Vision (ICCV)*. 1995, pp. 1064–1070.

[HC14] J. Hu and M. Chen. "A Sliding-Window Visual-IMU Odometer Based on Tri-Focal Tensor Geometry". In: *IEEE international conference on Robotics and automation (ICRA)*. Hong Kong, China, 2014, pp. 3963–3968.

[Hor87] B. K. P. Horn. "Closed-Form Solution of Absolute Orientation Using Unit Quaternions". In: *JOSA A* 4.4 (1987), pp. 629–642.

[Hub64] P. J. Huber. "Robust Estimation of a Location Parameter". In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101.

[HZ03] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[JU04] S. J. Julier and J. K. Uhlmann. "Unscented Filtering and Nonlinear Estimation". In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422.

[Kae+12] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. "iSAM2: Incremental Smoothing and Mapping Using the Bayes Tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[Kas+16] R. Käslin, P. Fankhauser, E. Stumm, Z. Taylor, E. Mueggler, J. Delmerico, D. Scaramuzza, R. Siegwart, and M. Hutter. "Collaborative Localization of Aerial and Ground Robots Through Elevation Maps". In: *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Lausanne, Switzerland, 2016, pp. 284–290.

[KCS11] L. Kneip, M. Chli, and R. Y. Siegwart. "Robust Real-Time Visual Odometry with a Single Camera and an IMU". In: (Dundee, Scotland, UK, 2011).

[KFL01] F. R. Kschischang, B. J. Frey, and H. Loeliger. "Factor Graphs and the Sum-Product Algorithm". In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519.

[KHS17] M. Kok, J. D. Hol, and T. B. Schön. "Using Inertial Sensors for Position and Orientation Estimation". In: *arXiv preprint arXiv:1704.06053* (2017).

[KK16] W. Kocay and D. L. Kreher. *Graphs, Algorithms, and Optimization*. Chapman and Hall, 2016.

[KRD08] M. Kaess, A. Ranganathan, and F. Dellaert. "iSAM: Incremental Smoothing and Mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[KSC18] M. Karrer, P. Schmuck, and M. Chli. "CVI-SLAM: Collaborative Visual-Inertial SLAM". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 2762–2769.

[KTT13] M. Kontitsis, E. A. Theodorou, and E. Todorov. "Multi-Robot Active SLAM with Relative Entropy Optimization". In: *IEEE American Control Conference (ACC)*. Washington DC, USA, 2013, pp. 2757–2764.

[LD13] V. Lui and T. Drummond. "An Iterative 5-pt Algorithm for Fast and Robust Essential Matrix Estimation." In: *British Machine Vision Conference (BMVC)*. Bristol, England, 2013.

[Lee+10] G. H. Lee, M. Achtelik, F. Fraundorfer, M. Pollefeys, and R. Siegwart. "A Benchmarking Tool for MAV Visual Pose Estimation". In: *International Conference on Control Automation Robotics & Vision (ICARCV)*. Singapore, Singapore, 2010, pp. 1541–1546.

[Leu+15] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-Based Visual-Inertial Odometry using Nonlinear Optimization". In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[Lin+18] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen. "Autonomous Aerial Navigation Using Monocular Visual-Inertial Fusion". In: *Journal of Field Robotics* 35.1 (2018), pp. 23–51.

[Liu+16] S. Liu, K. Mohta, S. Shen, and V. Kumar. "Towards Collaborative Mapping and Exploration using Multiple Micro Aerial Robots". In: *Experimental Robotics*. Springer. 2016, pp. 865–878.

[LK81] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: (1981).

[LMF09] V. Lepetit, F. Moreno-Noguer, and P. Fua. "Epnp: An Accurate O(n) Solution to the Pnp Problem". In: *International journal of computer vision* 81.2 (2009), p. 155.

[Loi+16] G. Loianno, Y. Mulgaonkar, C. Brunner, D. Ahuja, A. Ramanandan, M. Chari, S. Diaz, and V. Kumar. "A Swarm of Flying Smartphones". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 1681–1688.

[Lon87] H. C. Longuet-Higgins. "Double and Triple Ambiguities in the Interpretation of Two Views of a Scene." In: *Alvey Vision Conference*. 1987, pp. 1–6.

[Low04a] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[Low04b] G. Lowe. "SIFT-The Scale Invariant Feature Transform". In: *Journal of Computer Vision* 60 (2004), pp. 91–110.

[LS12] T. Lupton and S. Sukkarieh. "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments without Initial Conditions". In: *IEEE Transactions on Robotics* 28.1 (2012), pp. 61–76.

[LTK15] G. Loianno, J. Thomas, and V. Kumar. "Cooperative Localization and Mapping of MAVs using RGB-D Sensors". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, USA, 2015, pp. 4021–4028.

[Mel+13] D. Mellinger, M. Shomin, N. Michael, and V. Kumar. "Cooperative Grasping and Transport Using Multiple Quadrotors". In: *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 545–558.

[MMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (2015), pp. 1147–1163.

[Mon+02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem". In: *Aaai/iaai*. 2002, pp. 593–598.

[Mor78] J. J. Moré. "The Levenberg-Marquardt Algorithm: Implementation and Theory". In: *Numerical Analysis*. Springer, 1978, pp. 105–116.

[MR08] F. M. Mirzaei and S. I. Roumeliotis. "A Kalman Filter-Based Algorithm for IMU-Camera Calibration: Observability Analysis and Performance Evaluation". In: *IEEE Transactions on Robotics* 24.5 (2008), pp. 1143–1156.

[MT17] R. Mur-Artal and J. D. Tardós. "Visual-Inertial Monocular SLAM with Map Reuse". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 796–803.

[MTS17] A. L. Majdik, C. Till, and D. Scaramuzza. "The Zurich Urban Micro Aerial Vehicle Dataset". In: *The International Journal of Robotics Research* 36.3 (2017), pp. 269–273.

[Nes+16] T. Nestmeyer, P. R. Giordano, H. H. Bülthoff, and A. Franchi. "Decentralized Simultaneous Multi-Target Exploration Using a Connected Network of Multiple Robots". In: *Autonomous Robots* (2016), pp. 1–23.

[Nis04] D. Nistér. "An Efficient Solution to the Five-Point Relative Pose Problem". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6 (2004), pp. 756–770.

[NLD11] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. "DTAM: Dense Tracking and Mapping in Real-Time". In: *IEEE International Conference on Computer Vision (ICCV)*. Barcelona, Spain, 2011, pp. 2320–2327.

[PFS14] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, 2014, pp. 2609–2616.

[PHK16] A. Prorok, M. A. Hsieh, and V. Kumar. "Formalizing the Impact of Diversity on Performance in a Heterogeneous Swarm of Robots". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 5364–5371.

[QLS17] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *arXiv preprint arXiv:1708.03852* (2017).

[Qur+16] A. Quraishi, T. Cieslewski, S. Lynen, and R. Siegwart. "Robustness to Connectivity Loss for Collaborative Mapping". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea, 2016, pp. 4580–4585.

[Ram+16] R. Ramaithitima, M. Whitzer, S. Bhattacharya, and V. Kumar. "Automated Creation of Topological Maps in Unknown Environments Using a Swarm of Resource-Constrained Robots". In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 746–753.

[RB02] S. I. Roumeliotis and G. A. Bekey. "Distributed Multi-Robot Localization". In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 781–795.

[Rub+11] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. "ORB: An Efficient Alternative to SIFT or SURF". In: *IEEE International Conference on Computer Vision (ICCV)*. Barcelona, Spain, 2011, pp. 2564–2571.

[Sa+13] I. Sa, H. He, V. Huynh, and P. Corke. "Monocular Vision Based Autonomous Navigation for a Cost-Effective MAV in GPS-Denied Environments". In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. Wollongong, Australian, 2013, pp. 1355–1360.

[Sas+16] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar. "Swarm Distribution and Deployment for Cooperative Surveillance by Micro-Aerial Vehicles". In: *Journal of Intelligent & Robotic Systems* 84.1-4 (2016), pp. 469–492.

[SC18] P. Schmuck and M. Chli. "CCM-SLAM: Robust and Efficient Centralized Collaborative Monocular Simultaneous Localization and Mapping for Robotic Teams". In: *Journal of Field Robotics* (2018).

[Sch+15] M. J. Schuster, C. Brand, H. Hirschmüller, M. Suppa, and M. Beetz. "Multi-robot 6D graph SLAM connecting decoupled local reference filters". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, 2015, pp. 5093–5100.

[Sch+18] 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". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1418–1425.

[SDA18] J. Solà, J. Deray, and D. Atchuthan. "A Micro Lie Theory for State Estimation in Robotics". In: *arXiv preprint arXiv:1812.01537* (2018).

[Sim06] D. Simon. *Optimal State Estimation: Kalman, H infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[SK08] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer, 2008.

[SMD10] H. Strasdat, J. M. M. Montiel, and A. J. Davison. "Real-Time Monocular SLAM: Why Filter?" In: *IEEE International Conference on Robotics and Automation (ICRA)*. Anchorage, Alaska, 2010, pp. 2657–2664.

[SMK15] S. Shen, N. Michael, and V. Kumar. "Tightly-Coupled Monocular Visual-Inertial Fusion for Autonomous Flight of Rotorcraft MAVs". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Seattle, USA, 2015, pp. 5303–5310.

[Spa+17] A. Spaenlehauer, V. Frémont, Y. A. Şekercioğlu, and I. Fantoni. "A Loosely-Coupled Approach for Metric Scale Estimation in Monocular Vision-Inertial Systems". In: *IEEE International Conference on Multi-Sensor Fusion and Integration for Intelligent Systems (MFI)*. Daegu, Korea, 2017, pp. 137–143.

[Ste+16] J. Stephan, J. Fink, V. Kumar, and A. Ribeiro. "Hybrid Architecture for Communication-Aware Multi-Robot Systems". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, 2016, pp. 5269–5276.

[Stu+12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. "A Benchmark for the Evaluation of RGB-D SLAM Systems". In: *IEEE/RSJ International Conference on Intelligent Robot Systems (IROS)*. Vilamoura, Algarve, Portugal, 2012.

[WS11] S. Weiss and R. Siegwart. "Real-Time Metric State Estimation for Modular Vision-Inertial Systems". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China, 2011, pp. 4531–4537.

[WSS11] S. Weiss, D. Scaramuzza, and R. Siegwart. "Monocular SLAM Based Navigation for Autonomous Micro Helicopters in GPS-Denied Environments". In: *Journal of Field Robotics* 28.6 (2011), pp. 854–874.

[YBH15] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad. "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics". In: *Intelligent Industrial Systems* 1.4 (2015), pp. 289–311.

[ZGS18] Z. Zhang, G. Gallego, and D. Scaramuzza. "On the Comparison of Gauge Freedom Handling in Optimization-Based Visual-Inertial State Estimation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2710–2717.

[ZT13] D. Zou and P. Tan. "CoSLAM: Collaborative Visual SLAM in Dynamic Environments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.2 (2013), pp. 354–366.