DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
ESCUELA SUPERIOR DE INGENIERÍA
UNIVERSIDAD DE SEVILLA

Long-Term Localization of Unmanned Aerial Vehicles based on 3D Environment Perception

por

Francisco Javier Pérez Grau

Ingeniero de Telecomunicación

PROPUESTA DE TESIS DOCTORAL
PARA LA OBTENCIÓN DEL TÍTULO DE

DOCTOR POR LA UNIVERSIDAD DE SEVILLA

SEVILLA, 2017

Directores

Dr.-Ing. Fernando Caballero Benítez, Profesor Titular

Dr.-Ing. Aníbal Ollero Baturone, Catedrático

UNIVERSIDAD DE SEVILLA

Memoria para optar al grado de Doctor por la Universidad de Sevilla

Autor:	**Francisco Javier Pérez Grau**
Título:	**Long-Term Localization of Unmanned Aerial Vehicles based on 3D Environment Perception**
Departamento:	**Departamento de Ingeniería de Sistemas y Automática**

V° B° Director:

Fernando Caballero Benítez

V° B° Director:

Aníbal Ollero Baturone

El autor:

Francisco Javier Pérez Grau

A problem well put is half solved.
(John Dewey)

Acknowledgements

First of all, I would like to sincerely thank my supervisors Fernando Caballero and Aníbal Ollero for all the support, time, patience and teaching that I received from them. I appreciate a lot the amusing moments while developing, testing or writing with Fernando, as well as the effort and inspiring contributions from Aníbal towards this dissertation.

I must extend my gratitude to Antidio Viguria, since this work would not have been possible without his guidance and constant advice. He provided me with all the necessary resources to achieve my goals, and has always endeavored to ensure the maximum overlap between this dissertation and the different research projects in which I have participated over the past years.

During my time as a PhD candidate, I had the opportunity to do a research visit at the Australian Centre for Field Robotics (ACFR) in Sydney. I want to thank Ali Haydar Göktoğan for his friendly hosting during my stay, aligning his research interests with those of this dissertation. I am happy to have shared those months with Will Reid at The Mars Lab, and most notably with Dan Wilson who embraced me as a brother so far away from home.

Research activities like the ones carried out towards this dissertation would not have been possible without the great team of people at the Center for Advanced Aerospace Technologies (CATEC). I would like to thank my colleagues for their kindness, help and encouragement. The combination of excellent professional people in a friendly environment has led to many happy moments. In particular, I am grateful to Ángel Petrus and Ricardo Ragel for spending countless hours testing the aerial platform. I am also happy to have shared the indoor testbed with several fellows,

especially Miguel Ángel Trujillo, who gave highly appreciated constructive feedback to this work in order to further improve it, apart from his bright ideas regarding "strings engineering" (Ángel too!).

And last, but certainly not least, I would like to thank my parents for all their love and support. They have enabled me to address the subsequent challenges I have faced throughout my life. Also, my closest friends who have been there whenever needed, in good times and bad. And most of all María, my loving, encouraging and patient life partner, whose faithful support during the final stages of this work, suffering my uncountable working hours at nights or holidays, is so appreciated. Thank you.

Seville, May 2017.

Abstract

Unmanned Aerial Vehicles (UAVs) are currently used in countless civil and commercial applications, and the trend is rising. Outdoor obstacle-free operation based on Global Positioning System (GPS) can be generally assumed thanks to the availability of mature commercial products. However, some applications require their use in confined spaces or indoors, where GPS signals are not available. In order to allow for the safe introduction of autonomous aerial robots in GPS-denied areas, there is still a need for reliability in several key technologies to procure a robust operation, such as localization, obstacle avoidance and planning.

Existing approaches for autonomous navigation in GPS-denied areas are not robust enough when it comes to aerial robots, or fail in long-term operation. This dissertation handles the localization problem, proposing a methodology suitable for aerial robots moving in a Three Dimensional (3D) environment using a combination of measurements from a variety of on-board sensors. We have focused on fusing three types of sensor data: images and 3D point clouds acquired from stereo or structured light cameras, inertial information from an on-board Inertial Measurement Unit (IMU), and distance measurements to several Ultra Wide-Band (UWB) radio beacons installed in the environment. The overall approach makes use of a 3D map of the environment, for which a mapping method that exploits the synergies between point clouds and radio-based sensing is also presented, in order to be able to use the whole methodology in any given scenario.

The main contributions of this dissertation focus on a thoughtful combination of technologies in order to achieve robust, reliable and computationally efficient long-term localization of UAVs in indoor environments. This work has been validated

and demonstrated for the past four years in the context of different research projects related to the localization and state estimation of aerial robots in GPS-denied areas. In particular the European Robotics Challenges (EuRoC) project, in which the author is participating in the competition among top research institutions in Europe.

Experimental results demonstrate the feasibility of our full approach, both in accuracy and computational efficiency, which is tested through real indoor flights and validated with data from a motion capture system.

Keywords: Aerial Robots, Field Robotics, Visual-Inertial Localization, Multi-Sensor Fusion.

Acronyms

2D	Two Dimensional
3D	Three Dimensional
ACFR	Australian Centre for Field Robotics
AMCL	Adaptive Monte Carlo Localization
ANN	Approximate Nearest Neighbours
BA	Bundle Adjustment
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
CATEC	Center for Advanced Aerospace Technologies
CCNY	City College of New York
CDTI	Centro para el Desarrollo Tecnológico Industrial
CES	Consumer Electronic Show
CMOS	Complementary Metal-Oxide-Semiconductor
D&S	Defence&Space
DJI	Dà-Jiāng Innovations Science and Technology Co., Ltd
DoF	Degrees-of-Freedom
EKF	Extended Kalman Filter
ENAC	École nationale de l'aviation civile
EPnP	Efficient Perspective-n-Point
ETH	Eidgenössische Technische Hochschule
EU	European Union
EuRoC	European Robotics Challenges
FAST	Features from the Accelerated Segment Test

FCC	Federal Communication Commission
FKIE	Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie
FP7	Seventh Framework Program
FREAK	Fast Retina Keypoint
GPS	Global Positioning System
HMI	Human Machine Interface
ICAO	International Civil Aviation Authority
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
IR	Infrared
KF	Kalman Filter
LED	Light-Emitting Diode
LIDAR	LIght Detection And Ranging
LM	Levenberg–Marquardt
MAMMOTH	Mars Analog Multi-Mode Traverse Hybrid
Mbps	Megabits per second
MCL	Monte Carlo Localization
NASA	National Aeronautics and Space Administration
NUC	Next Unit of Computing
ORB	Oriented FAST and Rotated BRIEF
P3P	Perspective-Three-Point
PCL	Point Cloud Library
PMD	Photon Mixing Device
PnP	Perspective-n-Point
RF	Radio Frequency
RGB-D	Red,Green,Blue-Depth
RMS	Root-Mean-Square
RO-SLAM	Range-Only Simultaneous Localization And Mapping
ROS	Robot Operating System
RPA	Remotely Piloted Aircraft

RPAS	Remotely Piloted Aircraft System
RSS	Received Signal Strength
RTAB-Map	Real-Time Appearance-Based Mapping
S-PTAM	Stereo Parallel Tracking and Mapping
SBA	Sparse Bundle Adjustment
SLAM	Simultaneous Localization And Mapping
STAIR	STanford Artificial Intelligence Robot
SVD	Singular Value Decomposition
TOA	Time Of Arrival
ToF	Time of Flight
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
UWB	Ultra Wide-Band
VI	Visual-Inertial
WLAN	Wireless Local Area Network

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 The Localization Problem

Imagine you are at home pleasantly watching a movie on a Saturday night, and suddenly the power goes out. Your cell phone is not at hand so you need to go check your fuses in complete darkness. Even though you know exactly how to reach that place in your house, and you think that the steps you make are taking you there properly, you will more likely bump into nearby obstacles such as tables or doors.

Most of us rely on our eyes to help us find our way. By using the information from our eyes and other senses, we get an idea of where we are in the world, and thus can correct the inherent imperfections of our movements. Mobile robots suffer from the same issues when traversing the world. If they only move around without looking at where their movements are taking them, imperfections in their moving mechanisms will get them lost. Just like humans, sensing the environment can help them detect these imperfections and get a better idea of where they are.

Besides mobility, autonomy is another key feature that enables robots to operate effectively in complex environments and perform tasks without explicit human intervention. Sensing the environment allows an autonomous robot to decide the execution of specific actions according to the data that is being gathered. But before doing

so, the robot needs to be capable of self-localizing in its environment. Localization is one of the basic pillars of autonomous mobile robotics. The localization problem is essential for building a mobile robotic system, since accurate pose estimation is required for even the most basic tasks, even holding the position. It is commonly referred to as "the most fundamental problem to providing a mobile robot with autonomous capabilities" Cox (1991). Once this is achieved, the rest of technologies such as navigation and guidance can be implemented in order to determine where the goals are and how to reach them.

1.1.2 Aerial Robots: Definitions and Categories

Nowadays, the aerial vehicles popularly known as "drones" are not unfamiliar to any of us. Though often associated with military activity, there is also keen interest in many civilian applications. Apparently, this is the last example of military technology transferred and made available for civilian use, such as the Internet or Global Positioning System (GPS). These platforms have brought a revolution in almost every field, largely due to the decreasing cost of technology and the fact that they have distinct functional advantages with respect to manned aviation. Industry experts indicate that this technology has been the most dynamic growth sector of the aerospace industry in the past decade Cavoukian (2012). They are often present in the news on television, newspapers, radio or social networks, used by domestic law enforcement, the private sector or amateur enthusiasts. Factors like the ability of rapidly exploring large and/or inaccessible areas, the reduction of material costs as well as personnel costs, the process automation and the reduction of working times make them suitable for many uses in industrial, governmental and academic fields. They are already being used in a variety of applications, and many more areas will benefit by their use Jenkins and Vasigh (2013). Among these areas, we can find wildfire mapping Merino et al. (2012), agricultural monitoring Saari et al. (2011), disaster management Maza et al. (2011), power line surveys Wang et al. (2010), law enforcement Puri (2005), telecommunication Zhan et al. (2011), weather monitoring Revercomb et al. (1996), aerial imaging/mapping Nex and Remondino (2014), television news coverage

CNN (2016), sporting events Pedersen and Cooke (2006), movie-making Lin and Yang (2014), environmental monitoring Acevedo et al. (2013), oil and gas exploration Hausamann et al. (2005), etc.

Common acronyms often used are UAV, UAS, RPA or RPAS; however, they do not mean the same. "Drone" is the popular denomination in the media, and started out as a military term; nevertheless, it is not commonly applied by specialized personnel in research activities. For ease of clarity, we will briefly discuss the meanings of the main acronyms in order to properly introduce the specific category used throughout this thesis. According to the International Civil Aviation Authority (ICAO) Cary (2011), the main terminology is explained as follows:

- UAV (Unmanned Aerial Vehicle): an aircraft which is intended to operate with no pilot on board.
- UAS (Unmanned Aircraft System): it is the UAV and its associated elements which are operated with no pilot on board, i.e. communication link, ground control station, etc.
- RPA (Remotely Piloted Aircraft): an aircraft where the flying pilot is not on board (it is a subcategory of UAV).
- RPAS (Remotely Piloted Aircraft System): it is the RPA and its associated remote pilot station, the required command and control links and any other system elements as may be required, at any point during flight operation.

To summarize, the UAV (or UAS) is any aircraft (or system) in which the pilot is not physically on board the platform. In the case of RPA (or RPAS), express reference is made to the existence of a pilot who remotely operates the aircraft; whereas the definition of UAV (or UAS) leaves the option of carrying out the flight, or parts of it, as a fully autonomous operation. Hence all RPAS are UAS, but not all UAS are RPAS. That said, since this work focuses on the autonomous operation of aerial vehicles, we will refer to them as UAV or UAS, depending on the reference to the aerial platform or the complete system. In the scope of this dissertation, we will also refer to UAVs as aerial robots.

1.1.3 Localization of UAS

Although UASs seem to be in vogue today, they have been the target for extensive research activities for the past decades Howard and Kaminer (1995). Outdoor localization has greatly benefited from the use of GPS coupled with inertial measurements George and Sukkarieh (2005). Hence, obstacle-free outdoor operation can be generally assumed for UAS, due on the existence of several mass-produced models from different vendors, especially those from Dà-Jiāng Innovations Science and Technology Co., Ltd (DJI). These platforms are capable of offering mature performances in terms of autonomous waypoint navigation in a wide variety of scenarios, such as aerial surveys Siebert and Teizer (2014), cultural heritage Bolognesi et al. (2015) or precision agriculture Mesas-Carrascosa et al. (2014) among others. Figure 1.1 shows an example of outdoor inspection in an open area based on a DJI platform carried out in a refinery.

Figure 1.1: GPS-based aerial survey in a refinery.

Despite this, UASs are starting to play a major role in applications such as inspection Raja and Pang (2016), search and rescue Cui et al. (2016) or security surveillance Lee et al. (2015), which usually require the UAV to fly in dense environments, at low altitudes or indoors. In such cases, GPS signals are often shadowed or not available. This results in the unavailability of reliable position estimations. Relying exclusively on GPS for positioning might also pose safety issues in terms of signal jamming or

platform hijacking, resulting in a potentially vulnerable system Shepard et al. (2012). The introduction of aerial robots collaborating with humans in different applications is also a strong motivation for procuring reliable autonomous systems that can operate safely in their functioning in any situation.

Autonomous operation of UAS in GPS-denied areas has recently been tackled by platform vendors such as the aforementioned DJI with their commercial product Guidance DJI (2016), or Intel Intel (2015) with their live demonstrations at the Consumer Electronic Show (CES). They both show primary positioning abilities and/or collision avoidance capabilities, but these are not mature enough to be safely introduced in real and highly dynamic scenarios, and work reliably during long periods of time. For example, the safety guidelines from DJI's Guidance include a number of usage notices that restrict its operation depending on lighting conditions, the aspect of surrounding surfaces or the presence of moving objects. Hence, there is still a need for reliability in several key enabling technologies to achieve a safe and robust autonomous aerial robotic system.

In order to compensate the aforementioned issues involving GPS-based localization, or lack of robustness in the currently available solutions for GPS-denied scenarios, other methods must be used. This has been an active research topic for the past decades. Ground robots moving in Two Dimensional (2D) environments have demonstrated good performances in terms of localization either indoor Marder-Eppstein et al. (2010) or on streets and highways Montemerlo et al. (2008); Thrun (2011). However, aerial robots move and operate in Three Dimensional (3D) environments, which implies a higher computational complexity. Apart from the space dimensionality, UAS are exposed to additional challenges since the on-board capabilities are limited in terms of payload or computational resources. Besides, most aerial robots cannot assume the existence of reliable localization systems as it is usually assumed in ground robots.

1.2 Localization of UAS in GPS-denied Areas

As previously stated, GPS-based localization is impractical in indoor environments due to the high attenuation of satellite signals. Therefore, alternative technologies

have been proposed and developed in the past years in order to achieve robust indoor localization for UAS applications Liu et al. (2007); Mautz and Tilch (2011).

Motion capture systems are one of the most common forms of performance capture, and are also widely applied to precise indoor localization. These systems are based on a network of cameras strategically positioned in order to have overlapping fields of view to enclose the operating area. There are different types of motion capture depending on the type of camera and the need of specific markers to be captured. Passive optical motion capture uses retro-reflective markers that are tracked by infrared (IR) cameras. These markers need to be rigidly attached to the UAV so the cameras can detect them. The system requires a calibration procedure before it can be used. This is the most common method used in the industry, providing fast (over 100Hz) and accurate (sub-millimeter precision) data and being able to monitor large volumes. However, the hardware is usually expensive since high-frame rate cameras are usually required. Another important drawback is that direct line of sight between the cameras and the UAV is essential; if the markers are blocked by another object, tracking will be affected. The most well-known systems for aerial robotics research are developed by Vicon Cory and Tedrake (2008); Michael et al. (2010); Ruiz et al. (2013); Lupashin et al. (2014), like the one shown in Figure 1.2 which is located at the Center for Advanced Aerospace Technologies (CATEC). There are relatively more affordable systems such as those developed by OptiTrack Marconi et al. (2012); Orsag et al. (2013). Nevertheless, this type of localization solution is not cost-effective for scaling to larger or other indoor scenarios, and the line of sight requirement could be difficult to meet in cluttered environments.

In order to achieve a robust and scalable UAV localization solution, and to operate truly autonomously, it seems reasonable to compute the pose estimations on-board the aerial robot, unlike the aforementioned off-board motion capture systems. The use of different sensors on-board the UAV can be applied to accomplish fully autonomous operations in indoor environments. These localization methods can generally be subdivided into two categories: landmark-based or map-based methods.

Figure 1.2: Vicon-based indoor testbed at CATEC.

1.2.1 Landmark-based approaches

Landmarks are features in the environment that an on-board sensor can detect. Sensor readings are analyzed for the existence of landmarks, and if detected, they are matched with *a priori* known information of the environment to determine the location of the UAV. Landmarks can be classified into active or passive landmarks.

Active landmarks, also called beacons, are landmarks that actively send location information. In the same way as GPS satellites work, radio transmitting devices can be used while another on-board device receives the signals sent by the beacons in order to determine its position. Due to the availability of low-energy sensors and radio frequency circuitry, there is a research trend on radio-based localization systems Gu et al. (2009). This technology relies on measuring the characteristics of received radio waves at the receiver end. The Received Signal Strength (RSS) or the Time Of Arrival (TOA) are often used in order to estimate the distance between two devices, since both are correlated with such distance. The use of Wireless Local Area Network (WLAN) access points effectively exploits the infrastructure in place, making it an easier solution to adopt. However, it does not provide enough accuracy to achieve safe autonomous operation of UAVs Khalajmehrabadi et al. (2017).

Ultra-Wide-Band (UWB) is a wireless communication technology which has attracted interest from the research community as a promising solution for precise target localization and tracking González et al. (2009); Tiemann et al. (2015); Guo et al. (2016). It is particularly well suited for short-distance indoor applications, using several fixed sensors placed at known positions in the environment, and a mobile sensor on-board the UAV. Hence, the position estimation of the on-board device can be obtained by trilateration with an accuracy of the order of that from the devices (generally a few centimeters). At the same time, this setup offers a relatively low-cost solution that can be implemented in almost any scenario, with the advantage of not requiring a direct line of sight between each pair of sensors. Besides, the data association problem is trivially solved by attaching the sensor identification to the range measurement information. However, the distance estimations suffer from attenuation across materials and multi-path propagation. Moreover, these sensors are poorly suited to constitute a full localization system, since the data provided is a simple measurement of the distance between two of them. This makes their integration in localization and/or mapping applications especially difficult due to the lack of bearing information, thus leading to multiple location hypotheses.

On the other hand, landmarks are passive if they do not actively transmit signals. Then, the robot needs to actively look for them in order to acquire position measurements. Techniques using passive landmarks rely on the successful detection of those landmarks from sensor readings, which obviously depends on the sensor type. Among the wide variety of sensing options available, probably the most extended approaches in the aerial robotics literature are based on cameras, due to the amount of information provided versus their low weight and cost. When three or more landmarks are detected by the camera, different techniques can be used to compute its location, such as triangulation, trilateration or multilateration, depending on the type of landmark and sensor readings.

Passive landmarks can be either artificial or natural. Artificial landmarks are particularly designed to be recognized by robots, and placed at specific locations in the environment which are known in advance. Several vision-based localization methods based on artificial landmarks have been proposed in the literature, using simple planar

markers Fiala (2004); Rekleitis et al. (2006). While these approaches significantly reduce the setup cost and difficulty of the localization system compared to motion capture systems, specific issues related to single camera-based approaches arise, such as limitations in terms of camera image quality or lighting.

As motion capture systems, solutions based on both active and passive landmarks usually require an associated infrastructure to be installed in the environment and properly calibrated. Other landmark-based approaches do not rely on existing infrastructure, but on the successful detection of features in the environment by on-board sensors. These landmarks are commonly referred to as *natural landmarks*, and are not specifically engineered to be used as localization means for robots Cesetti et al. (2010). They are already part of the environment, such as doors or windows in the case of indoor scenarios. Again, probably the most extended approaches for UAVs are based on cameras, due to the amount of information provided, their affordability, availability and low weight. Vision-based approaches for robot localization are very popular, even though they entail an important associated processing complexity Gupte et al. (2012).

In this sense, it is very common to find approaches that make use of monocular vision, sometimes fused with inertial sensors and altitude sensors (like barometers, ultrasonic sensors or lasers), in order to estimate the aerial robot localization based on Simultaneous Localization And Mapping (SLAM) Davison et al. (2007); Pinies et al. (2007); Weiss et al. (2011); Achtelik et al. (2012); Mur-Artal et al. (2015). These approaches work very well when we repeatedly visit the same area, but usually fail at high-speed aircraft motions in unknown scenarios. Optical flow approaches Honegger et al. (2013); Mebarki et al. (2016) make use of the same sensors but aim to estimate only linear velocities. These approaches have demonstrated to work very well; however, velocity integration for localization (odometry) quickly diverges, making these methods unusable for long-term localization.

At the cost of higher computational requirements, stereo-vision systems provide direct depth measurements in a general 3D environment for a given camera baseline Kitt et al. (2010); Geiger et al. (2011); Schmid et al. (2013). Both SLAM and pure odometry approaches demonstrate good results at short-term localization and in long trajectories Paz et al. (2008) even without loop closing Oleynikova et al. (2015).

Many of the algorithms developed for stereo-vision odometry and SLAM can be applied to 3D cameras based on pattern projection, i.e. Red,Green,Blue-Depth (RGB-D) cameras Endres et al. (2012); Kerl et al. (2013). These sensors have recently become a very popular option due to their low weight, low cost and the amount of information provided; apart from RGB images they directly provide depth images of the scene in front of the sensor, saving the burden of 3D reconstruction computation (as in stereo-vision systems). Besides, they exhibit another important advantage with respect to classic stereo-based approaches: depth estimation does not depend on the presence of distinct visual features in the scene in order to estimate the depth.

The availability of RGB-D cameras has made dense 3D point clouds available, which were previously only accessible using much more expensive sensors like Time of Flight cameras or scanning 3D laser rangefinders. There are several state-of-the-art open-source algorithms that provide localization estimations based on RGB-D cameras, such as RGBD-SLAM Endres et al. (2012), City College of New York (CCNY) RGB-D Dryanovski et al. (2013) or Real-Time Appearance-Based Mapping (RTAB-Map) Labbe and Michaud (2014) (see Figure 1.3). Nevertheless, they are rarely employed for on-line computation during the UAV flight, since these sensors typically generate voluminous data that typically cannot be processed in its entirety in real time (e.g. the Microsoft Kinect sensor produces over 9 million 3D points per second). The fact that the cited SLAM algorithms are continuously building their maps on-line greatly increases their computational requirements, which limit their usability and robustness given the usual on-board restrictions in terms of payload capacity and computational resources when working with UAVs.

Point cloud based odometry and localization is another significant research area. Although computationally expensive, recent advances in *kd*-trees Nuchter et al. (2007) and Approximate Nearest Neighbours (ANN) Marden and Guivant (2012) enable faster computation times in the presence of medium and dense point clouds. The problem of aligning a pair of point clouds is commonly known as registration. Its output is usually a transformation matrix representing the rotation and translation that would have to be applied on one of the clouds in order to be perfectly aligned with the other. In order to do this, several techniques use a landmark-based approach,

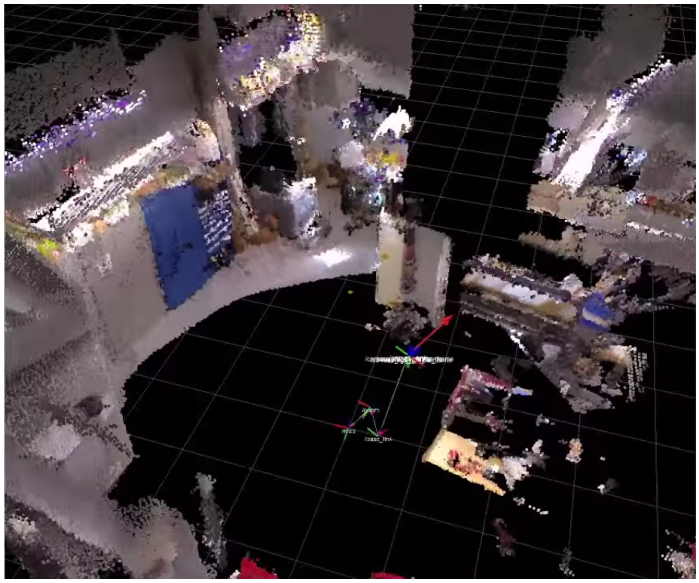

Figure 1.3: 3D localization and mapping using a hand-held RGB-D camera and RTAB-Map.

geometrically computing an estimate of the transformation based on the recognition of distinct features in the environment, occurring naturally or artificially placed Rusu et al. (2009). The factors contributing to the successful performance and integrity of these methods is the reliable acquisition and extraction of features from sensor data, and the ability to efficiently recognize and associate such features. This can be challenging depending on the spatial resolution and noise present in the depth sensor measurement, apart from the computational cost of analyzing dense 3D data. Another family of approaches is based on data correlation, attempting to utilize whatever sensor data are available to compute the transformation. This eliminates the need to decide what constitutes a feature, and uses a maximum likelihood alignment to find the best fit between two sets of data points. Iterative Closest Point (ICP) is a method capable of providing a computationally efficient pose estimations in complex, unstructured environments. ICP has the advantages of locally solving the problem of matching and localization, being generic and potentially used for real-time applications Pomerleau et al. (2013).

In general, vision-based odometry and localization systems for aerial robots are not reliable enough in the long term due not only to cumulative drift, but also external factors such as poor illumination, lack of texture, occlusions or moving objects. All these have a significant impact on the robustness and reliability of most state-of-the-art algorithms. In any case, the aforementioned approaches demonstrate fairly good results in the short term; however, they could quickly diverge depending on the environment.

Autonomous systems intended to operate over long periods of time usually perform some sort of loop closing in order to recognize revisited places. This allows reducing the localization uncertainty at the cost of adding computational complexity Sünderhauf and Protzel (2011); Lowry et al. (2016). However, the problem of loop closing in order to distinguish revisited places is usually framed as a classification task, rather than a robot localization task Angeli et al. (2008). Reliable place recognition can be challenging in large-scale environments or sites with repetitive structures that might exhibit similar scenes in different areas. This is a very delicate issue since a single wrong loop closure can result in a devastating failure of the localization system, which could lead to an undesired flight termination.

1.2.2 Map-based approaches

The other large family of localization techniques relies on previously built maps of the environment. Some UAS applications are usually carried out in known environments, e.g. logistic services or post-disaster assessments Ezequiel et al. (2014). In these cases, the aerial robot is required to carry out specific trajectories or reach certain places. This could be also a requirement for path planning in order to specify goals in a predefined coordinate system.

These approaches use features such as the lines or planes that describe walls in hallways or offices to build a map of the environment. Sensor data can be matched with such features in order to determine the UAV location. Probably the most common map representations are occupancy grids, which were first introduced several decades ago Moravec and Elfes (1985). These grids are a probabilistic approach to represent

the environment in discrete cells that indicate their probability of being occupied by an obstacle. A great advantage is that they do not rely on specific predefined features, and are able to represent unknown areas. However, an important drawback of this approach is its large memory requirement, but recent developments such as OctoMap Hornung et al. (2013) provide efficient data structures particularly suited for robotics applications in the constrained equipment usually found on-board UAVs Nieuwenhuisen et al. (2014); Droeschel et al. (2016).

One of the approaches based on a predefined model of the environment is commonly known as teach-replay Chen and Birchfield (2006); Royer et al. (2007), which is accomplished in two stages: first the robot is manually piloted along the desired path as in a teaching phase, and an accurate 3D map of the environment is built, along with the robot motion from this learning path; afterwards, this map is used to locate the robot when it repeatedly visits the same path. However, this approach is somewhat simplistic and limited, since it only enables a robot to follow a predetermined trajectory.

Monte Carlo Localization (MCL) is another approach that makes use of a known map of the environment, and is one of the most popular algorithms used for robot navigation in indoor environments Thrun et al. (2001). It is a probabilistic localization algorithm that makes use of a particle filter to estimate the pose of the robot within the map, based on sensor measurements. Important benefits include the possibility of accommodating arbitrary sensor characteristics, motion dynamics and noise distributions. In order to obtain a reliable localization result, a certain number of particles will be needed. The larger the environment is, the more particles are needed. Actually, each particle can be seen as a pseudo-robot, which perceives the environment using a probabilistic measurement model. At each iteration, the virtual measurement takes large computational costs if there are hundreds of particles. For that reason, there is a variant of MCL called Adaptive MCL (AMCL) Fox (2001). The term "adaptive" comes from the fact that the number of particles is adjusted dynamically: if there is high uncertainty about the robot pose, the number of particles increases; if the pose is well known, such number decreases. Experimental approaches of either MCL or AMCL exhibit some limitations when it comes to aerial robots. Due to the aforementioned

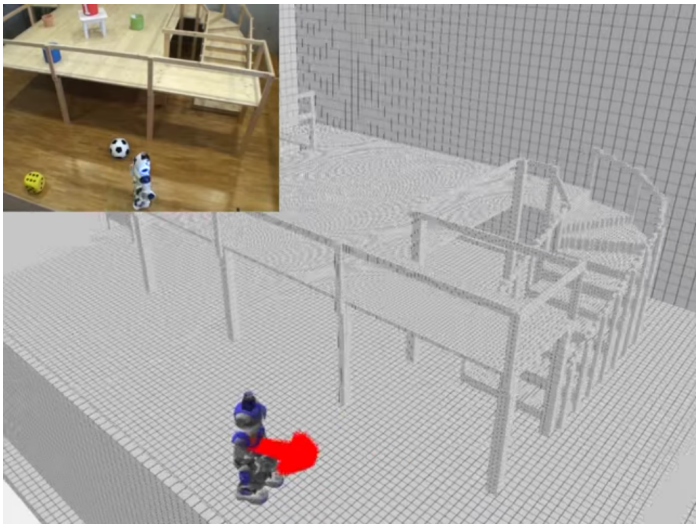

Figure 1.4: 6D localization for humanoid robots based on AMCL.

computational requirements, most of the existing approaches are meant for wheeled robots moving in a 2D environment, requiring a 2D laser scanner for map building and localization. Other authors presented an extension of this approach for 6D localization based on 2D laser scanner Hornung et al. (2010), but it is meant for 2D motion of humanoid robots in a 3D environment (see Figure 1.4), which makes it not suitable for UAVs.

1.3 Contributions

The aforementioned approaches are promising in that they can all provide solutions to the localization problem, but important drawbacks are present using each approach alone. Infrastructure-based positioning systems achieve reliable performances, but they can be expensive and might require laborious setup and calibration processes. Vision-based approaches based on natural landmarks demonstrate good results relying only on on-board equipment, but are not robust enough when applied to UAVs and especially for long-term operation (i.e. long flight time). Map-based methods are robust solutions for long-term localization, but often demand a high computational cost and they need to previously build an accurate representation of the environment.

The main contribution of this work focuses on the combination of technologies in order to achieve long-term autonomous operation of UAVs in indoor environments, taking advantage of their respective benefits to overcome their main limitations. This is accomplished by fusing data from different sensors in order to improve the performance of the overall system. In particular, a visual odometry algorithm based on stereo or RGB-D cameras and a localization algorithm based on UWB sensor beacons have been merged into an enhanced MCL algorithm which relies on a previously built multi-modal map that includes 3D occupancy data and the location of the UWB beacons. Specific contributions in each field are listed below, along with relevant related publications.

- Research and development of a robust visual odometry approach suitable for 3D sensors, which provides a reliable short-term pose estimation. Validation of the approaches using both aerial and ground robots.
  - F.J. Perez, J. Gil, G. Binet and A. Viguria, "Validation of 3D Environment Perception for Landing on Small Bodies using UAV Platforms", 13th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2015), ESA/ESTEC, Noordwijk, The Netherlands, 2015. Link.
  - W. Reid, F. J. Perez-Grau, A. H. Göktoğan and S. Sukkarieh, "Actively articulated suspension for a wheel-on-leg rover operating on a Martian analog surface", 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden, 2016, pp. 5596-5602. doi: 10.1109 / ICRA.2016.7487777
  - F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria and A. Ollero, "Semi-Autonomous Teleoperation of UAVs in Search and Rescue Scenarios", 2017 International Conference on Unmanned Aircraft Systems (ICUAS), Miami, FL, USA, 2017. Accepted for publication.
- Research and development of multi-modal sensor fusion methods that combine the aforementioned visual odometry with other sources of 3D measurements, namely radio-range sensing and point clouds, for long-term localization. The

noise and outliers from radio measurements are filtered thanks to the odometry estimations, while the odometry drift is bounded thanks to radio-based measurements and point cloud matching.

  - F. J. Perez-Grau, F. R. Fabresse, F. Caballero, A. Viguria and A. Ollero, "Long-term aerial robot localization based on visual odometry and radio-based ranging", 2016 International Conference on Unmanned Aircraft Systems (ICUAS), Arlington, VA, USA, 2016, pp. 608-614. doi: 10.1109 / ICUAS.2016.7502653
  - F. J. Perez-Grau, F. Caballero, A. Viguria and A. Ollero, "Multi-Sensor 3D Monte Carlo Localization (MCL) for Long-Term Aerial Robot Navigation", International Journal of Advanced Robotics Systems (IJARS). Accepted for publication.

- Research and development of a multi-modal map building algorithm that exploits the synergies between radio-based distance estimations and point clouds from 3D imaging sensors, which requires a minimum setup.

  - F. J. Perez-Grau, F. Caballero, L. Merino and A. Viguria, "Multi-Modal Mapping and Localization of Unmanned Aerial Robots based on Ultra-Wideband and RGB-D sensing", 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Under review.

- Development of a modular and extensible software architecture for safe and reliable autonomous navigation of aerial robots in GPS-denied environments, validated during extensive field testing throughout different experiment campaigns and demonstrations in the context of national and European Union (EU) funded research projects.

  - F. J. Perez-Grau, R. Ragel, F. Caballero, A. Viguria and A. Ollero, "An Architecture for Robust UAV Navigation in GPS-denied Areas", Journal of Field Robotics (JFR), Special Issue on High Speed Vision-Based Autonomous UAVs. Accepted for publication.

- Highly efficient implementation of all the algorithms in order to make them suitable for real-time on-line localization in the usually constrained equipment that can be mounted on-board a UAV.
  - Video showing on-line localization for the autonomous operation of a UAV using the proposed architecture and algorithms. EuRoC Challenge 3 – Team GRVC-CATEC – Stage IIb (Showcase).

It is important to point out the strong experimental focus of this dissertation, whose main contributions have been focused not only on advancing in the state-of-the-art, but also on implementing and validating different approaches in real-world setups. Proof of the potential impact of this work regarding its current technology development and future transfer to the industry is its recent recognition within the 1st EU Drone Awards, organized by the European Young Innovators Forum at the European Parliament in January 2017. A special innovative prize in the category "Best Drone-based Solution" was awarded to the application of indoor localization of UAVs to logistic operations in aircraft manufacturing plants EYIF (2017), where the system developed within this dissertation is used.

1.4 Thesis Framework

The research leading to these results has received support from the Spanish Centro para el Desarrollo Tecnológico Industrial (CDTI) INNPRONTA 2011-2014 program within Perigeo project, and the European Community's Seventh Framework Program (FP7) project EuRoC (FP7-ICT-608849) in the period 2014-2017.

This dissertation has been extensively validated within the context of the European Robotics Challenges (EuRoC)[1], an EU FP7 project whose main motivation is to bring innovative technologies from research labs to industrial end-users. In order to do that, a series of challenges in a public competition format were presented, and one of them (*Challenge 3*) is related to the demonstration of high-level semi-autonomous operation of a UAV for an inspection task. The goal is to enable unskilled workers

[1] http://www.euroc-project.eu

to perform complex inspection missions with the aid of UAVs. This is difficult to achieve as the complexity of UAVs requires expert piloting skills. In this context, a framework has been developed in order to perform localization and state estimation of the UAV without external positioning systems such as GPS or motion capturing systems, as well as autonomous local obstacle avoidance, local path planning, following of structures or homing of the aerial robot.

The author is part of the challenger team GRVC-CATEC, which is currently competing among top European research institutions, and has successfully demonstrated accurate localization of the UAV platform within the project.

1.5 Thesis Outline

The different chapters of this document describe the evolution of the developed system along the years of work towards this dissertation. There was an incremental strategy based on the accomplishment of intermediate development objectives, all of them aiming at the overall goal of achieving long-term localization of UAVs in GPS-denied areas, keeping in mind safety and robustness as flagship features for the successful widespread use of autonomous aerial robots. Following this incremental approach, the chapters are structured as follows:

- Chapter 2 discusses several sensing modalities that can be used throughout the different developments.

- Chapter 3 details the incremental works towards a vision-based odometry algorithm using images, 3D point clouds and inertial measurements.

- Chapter 4 presents the evolution of our approach to combine visual odometry with radio-based measurements from UWB beacons and a 3D map of the environment.

- Chapter 5 describes a 3D map building method in order to be able to use the previously described localization system in any environment.

- Chapter 6 introduces an overview of the whole framework in which this work has been tested and validated for autonomous navigation of UAVs in EuRoC project.
- Chapter 7 includes conclusions, lessons learned and future lines of work.

Figure 1.5 shows a pictorial representation of the thesis outline, presenting how chapters are related with each other.

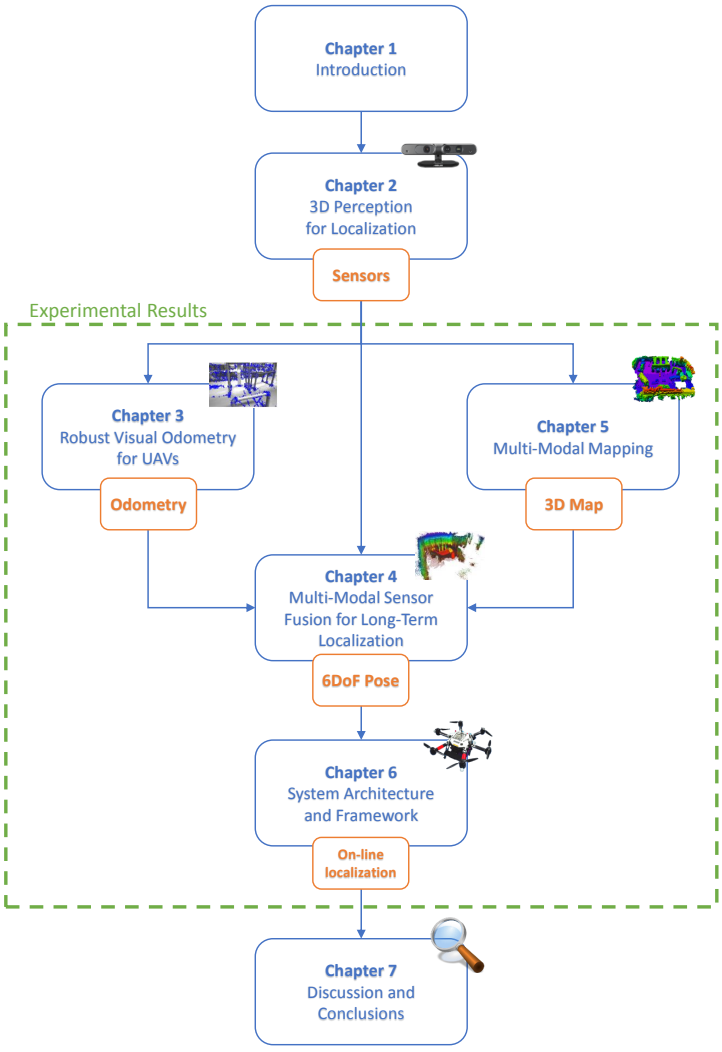

Figure 1.5: Thesis outline.

Chapter 2

3D Perception for Localization

The world around a UAV is 3D; yet traditional approaches for UAV localization and navigation are based on cameras, which are able to acquire only two-dimensional images that lack depth information. This limitation greatly reduces their ability to perceive and understand the complexity of their environment. Depth information is of great value in the analysis of scenes with 3D objects.

Over the past few decades, a significant evolution has taken place with regards to the sensing approaches that have been employed to gather 3D data, from simple ultrasonic range sensors that provide a few bytes of information about its environment, to spinning laser scanners that generate high-quality 3D representations of the world. Unfortunately, the latter sensors are very expensive and out of reach for many robotics projects, and their weight also limits their integration in small platforms. With the advent of new, low-cost 3D sensing hardware, and continued efforts in advanced data processing, 3D perception has gained a lot of importance in robotics, as well as other fields.

Generally speaking, 3D measurement sensors make use of electromagnetic energy in order to derive depth information of an object. Depending on the technology used to extract this information, several methods can be observed. For the purpose of this dissertation, technologies are subdivided into optical and non-optical sensing, depending on the nature of the waves on which the depth measurement relies:

- Optical sensing refers to extracting depth information from light pulses or waves. There is a wide variety of 3D optical techniques, and their classification is not unique. Some examples are stereo vision, laser time of flight or laser triangulation.

- Non-optical sensing includes other frequencies in the electromagnetic spectrum, such as radio- and micro-waves. These 3D sensing techniques typically determine distances to objects by measuring the time required for a pulse or wave energy to bounce back from the objects. Examples are acoustic sensors (e.g. ultrasonic) or radar.

This chapter focuses on the sensing modalities that are relevant in the context of this dissertation belonging to each of these two categories, discussing their appropriate use on-board aerial robots for accurate localization.

2.1 Optical Sensing

A thorough review of optical methods and sensors for 3D measurements is presented in Sansoni et al. (2009). In particular, this section focuses on 3D imaging sensors able to produce depth images, one of the simplest and most convenient ways of representing and storing depth measurements. In a depth image, each pixel value represents the distance to an object, with respect to a common reference frame, as depicted in Figure 2.1. Such representations are similar to gray-scale images, except the distance information replaces the intensity values.

Optical methods for generating depth images are generally classified in two categories:

- Passive methods: the reflectance of the objects and the illumination of the scene are used to derive shape information. These methods only require ambient lighting, but the less well-defined features an object may have, the less accurate the depth estimation will be. Examples of passive optical techniques are stereo vision, photogrammetry or shape from focus.

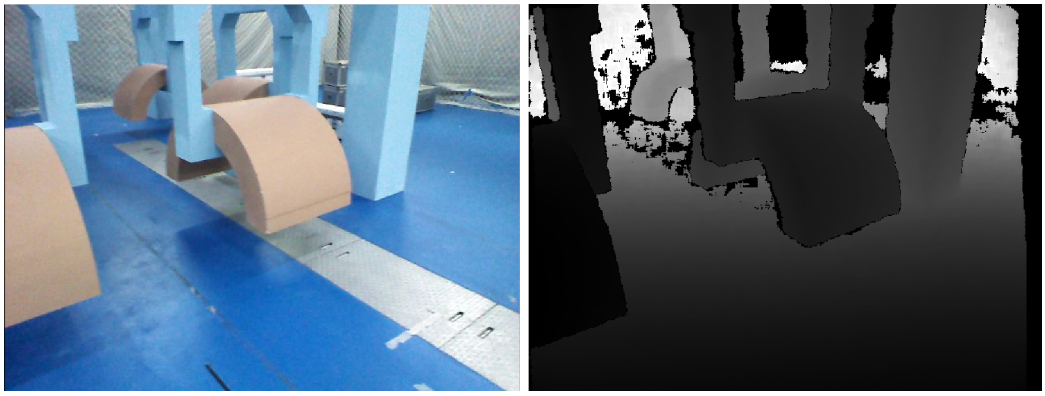

Figure 2.1: Sample color image (left) and its associated depth image (right).

- Active methods: suitable external lighting sources are used to illuminate the objects in the scene, and then determine their relative distances from the sensor. Generally, a known projected light pattern is used to illuminate the scene. However, if the objects exhibit varying surfaces, or the edges of objects need to be imaged, active lighting can produce measurement inaccuracies. Examples of active optical techniques are laser triangulators, structured light or time of flight.

2.1.1 Passive Imaging

As discussed in Section 1, cameras are probably the most popular sensing option for environment perception on-board UAVs. Low weight and cost are the two main properties that facilitate their integration and widespread use. However, obtaining 3D information directly from monocular sensing setups usually demands a high computational cost. Techniques such as shape from focus Nayar and Nakagawa (1994) or texture gradients Bajcsy and Lieberman (1976) are able to estimate depth but are not suited for real-time on-line operation. Other techniques such as shape from shading Horn and Brooks (1989) need external light sources and thus involve a setup that cannot be longer considered a “passive” system.

When considering high-speed depth computation, one of the most extended passive 3D systems is stereo imaging, which makes use of (at least) two cameras to capture two separate images of a scene from two different viewpoints.

A setup of two cameras mounted at a fixed distance is needed, so the cameras concurrently capture the same scene. Depth information can be obtained by examining the relative positions of objects in the two perspectives. Objects which are closer to the cameras will have a greater difference in apparent position between the two perspectives. Relative depth of points in the scene can then be computed since the depth of each point is inversely proportional to the difference in the distance of the corresponding points and their camera centers. This can be then used to generate a disparity map that visually provides 3D information. Disparity refers to the difference in image location of an object present on the two cameras. By triangulation, this disparity yields the object's depth. Each separate camera must be calibrated, and their relative location must be known so that triangulation methods can be used. Since the positional and optical parameters of the two cameras must be accurately calibrated, many products use dual setups that are pre-calibrated, relieving the system developer of such tasks.

As Figure 2.2 shows, a physical point P is taken up in the scene observed by two perspective cameras, namely a left (C^l) and a right (C^r) camera. If the corresponding pixel of this point is found in both images, its 3D position can be computed with the help of the triangulation principle. The cameras have focal length f and are located at a fixed distance B (*baseline*). If a matching pair of the point P is found in both images, shown as x^l and x^r, then it is possible to derive the depth Z by applying the principle of similar triangles:

$$\frac{Z}{B} = \frac{Z - f}{B - (x^l - x^r)} \Rightarrow Z = \frac{f \cdot B}{x^l - x^r} \tag{2.1}$$

where $x^l - x^r$ is the disparity value for point P. This equation also assumes that both cameras are correctly calibrated in order to remove any distortion in the images.

The main advantage of stereo-vision with respect to other range measuring devices is that it achieves high resolution and simultaneous acquisition of the entire depth

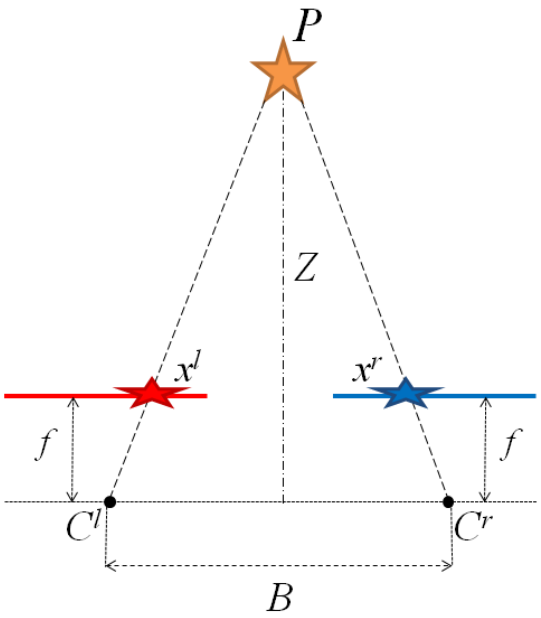

Figure 2.2: Stereo vision principle.

image without energy emission or moving parts. Since it is a passive method, no further equipment (e.g. specific light sources) and no special projections are required. Besides, cameras are a popular choice because they are small, lightweight and relatively cheap. Cameras are able to capture complete images in microseconds, hence they can be used as mobile sensors or operate in highly dynamic environments.

Nevertheless, stereo-vision systems present important issues of consideration. The major problem is the identification of common points within the image pairs, i.e. the solution of the well-known *correspondence problem* Scharstein et al. (2001). Depending on the strategy used to perform the matching, this computation might be costly. Moreover, if the scene is poor in texture and contrast information (due to homogeneous regions or poor illumination), stereo-based systems rapidly drop their performance, because the pixel values have little variation and the algorithm is not able to distinguish and match common points. Besides, depth data could be noisy, as it relies on the natural texture on the observed surfaces, and ambient lighting. Sensitivity to illumination and low accuracy on distant objects also limit the usability of these systems, since longer distances would require larger baseline (the distance between

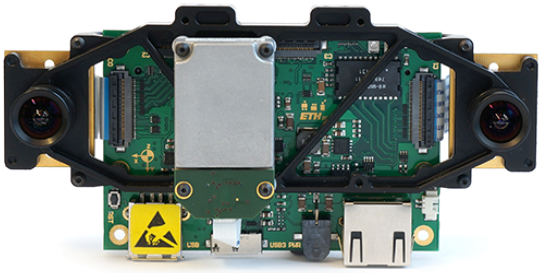

Figure 2.3: Skybotix's VI-Sensor Schneith (2014).

the cameras). Given the restrictions in camera sizes that can be mounted on-board a small UAV, applicable baselines can handle a relatively limited depth range.

The stereo-vision sensor used in some stages of this work is the Visual-Inertial (VI-) Sensor, shown in Figure 2.3, which provides fully time-synchronized and factory calibrated IMU- and stereo-camera data streams. In particular, stereo-vision has been used as the main odometry source in some of the experiments included in Chapters 3 and 6.

2.1.2 Active Imaging

LIDAR

Probably the most common sensors based on active optical methods for obtaining depth information are LIght Detection And Ranging (LIDAR) instruments. They are based on the well-known time of flight principle, i.e. the distance to an object is measured from the time it takes a light pulse to travel from the emitter until it returns reflected from the object. Hence an active light source is required, often in the IR frequency range. Since all electromagnetic energy travels at the speed of light c, in free space, the relationship between the distance Z and the round-trip travel time t is given by

$$Z = \frac{c \cdot t}{2} \tag{2.2}$$

LIDAR systems provide high precision measurements at high data rates; however, their high cost and relatively high weight, or the non-simultaneous points acquisition

(points are grabbed one after another, not simultaneously), limit their widespread use in some applications involving UAV platforms. For these reasons, this dissertation has not focused on this type of active imaging technique.

ToF cameras

Light pulses are not the only method to measure time of flight. By using a modulated signal, distances can be calculated by estimating the phase difference between the emitted and the reflected signal. The distance Z is then obtained as follows:

$$Z = \frac{c}{2 \cdot f_m} \cdot \frac{\phi}{2\pi} \tag{2.3}$$

where c is again the speed of light, f_m is the modulation frequency of the emitted signal and ϕ is the estimated phase shift between the emitted and received signal.

Due to the periodicity of the modulation signal, Equation 2.3 is only valid for distances smaller than $\frac{c}{2 \cdot f_m}$. The modulation frequency f_m of the emitted signal determines the "ambiguity-free" distance range of the sensor.

The last decade has seen an increasing trend in the development of 3D cameras. Time of Flight (ToF) cameras are a relatively new type of sensor that delivers 3D imaging at a high frame rate, simultaneously providing intensity data and range information for every pixel. Near-IR modulated light waves are emitted by several Light-Emitting Diodes (LEDs) and reflected by the objects on the scene back to the imager. Conventional imaging sensors consist of multiple photo-diodes arranged in a matrix. Normally, these diodes provide a gray-scale or color image of the scene. In contrast to normal cameras, a Photon Mixing Device (PMD) sensor additionally acquires a distance value for each pixel simultaneously to the intensity (gray) value. Despite this remarkable progress, it is still made with standard Complementary Metal-Oxide-Semiconductor (CMOS) technology. Therefore, the pixels in these cameras are often called "smart pixels" Xu et al. (1998).

Compared to other technologies that obtain 3D information, ToF cameras allow the acquisition of depth images without any scanning mechanism and from just one point of view. They register dense depth along with intensity images at a high frame

rate. They are small, low-weight and compact, since no mobile parts are needed. They have low costs compared to LIDAR systems, and a lower power consumption with respect to classical laser scanners. Another important characteristic of these devices is their ability to operate in environments where other systems would be unable to do so. For example in untextured scenarios, where the operation of stereo-vision systems would be very difficult due to the lack of representative features on the images to solve the *corresponding problem*.

Important disadvantages of these systems are their high sensitivity to noise, especially in outdoor scenarios due to interference with direct sunlight, and their low-depth measurement accuracy. This is due mainly to the way in which ToF cameras grab data, which consists in the scene being bombarded with near-IR light, capturing a whole surface included into the emitted light cone in one single shot. This differs from laser rangefinders, which acquire points sequentially with very high accuracy. These devices also provide very low resolution (no more than a few thousands of tens of pixels) compared to current standard cameras.

Nevertheless, when it comes to mounting a sensor on-board a UAV platform, additional considerations and requirements must be taken into account. Especially important are the weight, size and power consumption of the sensor, since the payload capacity of small UAVs is very limited. Due to these reasons, the preferred sensing modality for obtaining dense 3D depth images is based on another type of active imaging technique known as structured light.

Structured light

Sensors based on structured light simplify the solution of the *correspondence problem* introduced in stereo-vision techniques. They replace the second camera in the stereo setup by a light source which projects a known pattern of light on the scene. If the scene is simply a planar surface, the pattern acquired by the first camera will be similar to the projected pattern. When the surface is non-planar, the geometric shape of the surface distorts the projected pattern. The shape of this surface can then be obtained based on the information from the distortion of the projected pattern. Still, some correspondences between both patterns must be solved. Different projection

Figure 2.4: RGB-D camera (ASUS's Xtion PRO LIVE) Asus (2017).

patterns have been proposed such as binary codes or color-coded stripes Geng (2011). The most usual pattern is the projection of a grid, in which an easier correspondence problem has to be solved. In this case, we only need to identify, for each point of the captured pattern, the corresponding point of the projected pattern.

Apart from ToF cameras, a new type of sensors commonly referred to as RGB-D cameras can provide both visual texture information and per-pixel depth information simultaneously. They have two cameras: the first is usually a conventional webcam that records color video, and the second is an IR camera that records a non-visible structured light pattern generated by the IR projector (see Figure 2.4).

Microsoft's Kinect was probably the first affordable RGB-D camera widely used in robotics research Boudjit et al. (2008). The per-pixel depth-sensing technology that is used in consumer RGB-D cameras was developed and patented by PrimeSense Garcia and Zalevsky (2008). The depth acquisition technology is named *Light Coding*; it has an IR light source to project a complex pattern of dots into the scene. This pattern is perceived by an IR camera and the distance to each dot is computed by triangulation to build a 3D model of the scene. The color information of this model is gathered by an RGB camera.

Regardless of texture and illumination condition, an RGB-D camera can directly obtain 3D information, unlike stereo-vision due to the lack of representative features in the images. Above all, they are small, low-weight and compact, since no mobile

Figure 2.5: Orbbec's Astra Orbbec (2017).

parts are involved. Besides, they have much lower cost and lower power consumption when compared to LIDAR systems and ToF cameras.

One important limitation of RGB-D cameras is that they can only operate reliably indoors, since the projected pattern is overwhelmed by exterior lighting conditions. Another drawback is the measurement noise when compared to LIDAR sensors, and their limited working range (up to 10m). Nevertheless, RGB-D sensors have seen widespread adoption in robotics research due to their ability to generate reliable 3D data at a fast frame rate at low cost Han et al. (2013).

Regarding on-board requirements for the UAV platform, subsequent models of RGB-D cameras from other vendors exhibit great advantages with respect to Microsoft Kinect: they are significantly smaller, easier to integrate and do not require an external power supply. These aspects make it much more portable and suitable for small aerial vehicles. Examples of these other cameras are ASUS's Xtion PRO LIVE (shown in Figure 2.4), and more recently Orbbec's Astra (see Figure 2.5) which exhibits a longer depth range (up to 10m instead of 4m). These two cameras have been used in the experiments of this dissertation, the Xtion PRO LIVE in Chapter 3 and the Astra in Chapters 4, 5 and 6.

Table 2.1 summarizes the main specifications of the previously discussed RGB-D cameras.

Table 2.1: RGB-D cameras comparison

Specifications	Kinect	Xtion PRO LIVE	Astra
Size (mm)	305 x 63 x 76	180 x 35 x 50	165 x 30 x 40
Weight (g)	1320	540	300
Range (m)	0.8 - 4	0.8 - 3.5	0.6 - 8
Depth Image Size	640 x 480	640 x 480	640 x 480
RGB Image Size	640 x 480	640 x 480	640 x 480
Frames per second	30	30	30
Field of View (°)	57 x 43	58 x 45	60 x 49.5
Power	External	USB	USB

2.2 Non-optical Sensing

As stated earlier in this chapter, non-optical sensing refers to the use of pulses or waves not included in the visible or the IR spectrum for obtaining 3D measurements. This section focuses on microwaves, in particular on UWB, which is the technology that has been used in this work.

UWB is a high data rate, low power short-range wireless technology that is generating a lot of interest in the research community and the industry, as a high-speed alternative to existing wireless technologies such as IEEE 802.11 WLAN, Home Radio Frequency (RF) and HiperLANs Lad (2004). Even though UWB has been around for more than 40 years, a substantial change occurred in 2002, when the Federal Communication Commission (FCC) issued a report allowing the commercial and unlicensed deployment of UWB with a given spectral mask for both indoor and outdoor applications in United States. This wide frequency allocation initiated a lot of research activities from both industry and academia, focusing in recent years on consumer electronics and wireless communications.

UWB transmits binary data, using low energy and extremely short duration impulses or bursts (in the order of picoseconds) over a wide spectrum of frequencies. It delivers data over 15 to 100 meters and does not require a dedicated radio frequency,

so is also known as carrier-free, impulse or base-band radio. UWB systems use carrier-free, meaning that data is not modulated on a continuous waveform with a specific carrier frequency, as in narrowband and wideband technologies.

UWB technology has the following significant characteristics Lad (2004):

- **High data rate**: it can handle more bandwidth-intensive applications like streaming video, than either 802.11 or Bluetooth, reaching data rates of roughly 100 Megabits per second (Mbps), with speeds up to 500 Mbps. The maximum speed for 802.11a is 54 Mbps, while for Bluetooth it is about 1 Mbps.

- **Low power consumption**: it constantly transmits short impulses, instead of transmitting modulated waves like most narrowband systems do, and hence does not need conversion between frequencies, local oscillators, mixers, and other filters.

- **Interference immunity**: due to low power and high frequency transmission, UWB's aggregate interference is vaguely detected by narrowband receivers. This makes it suitable for coexistence with narrowband radio systems operating in the same spectrum without causing undue interference.

- **High security**: since UWB systems' noise is very low, they are inherently covert and extremely difficult for unintended users to detect.

- **Low complexity, low cost**: the most attractive of UWB's advantages are its low system complexity and cost. Traditional carrier based technologies modulate and demodulate complex analog carrier waveforms. Due to the absence of carrier in UWB, the transceiver structure can be very simple. Besides, recent advances in silicon process and switching speeds make UWB systems also low-cost.

- **Resistance to jamming**: UWB spectrum covers a huge range of frequencies. That is why its signals are relatively resistant to jamming, because it is very difficult to jam every frequency in the UWB spectrum at the same time. Therefore, there is a wide frequency range available even in the case that some frequencies are jammed.

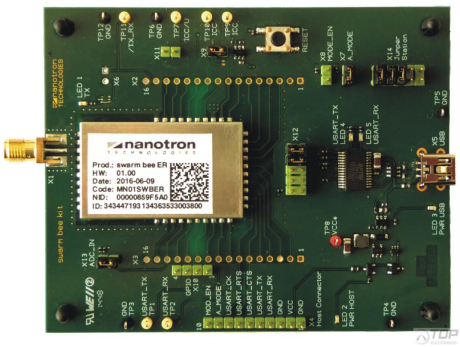

Figure 2.6: Nanotron's swarm ER Nanotron (2017).

- **Scalability**: UWB systems are very flexible because their common architecture is software re-definable, so that it can dynamically trade-off high-data throughput for range Chong et al. (2006).

UWB signaling is especially suitable for position and ranging applications due to its low energy, high bandwidth and fine temporal resolution properties. It allows accuracies of a few centimeters in ranging, as well as low-power and low-cost implementation of communication systems Gezici et al. (2005). The process involves exchange of signals between nodes, and measurement of parameters to estimate distances.

Distance measurements between two radio-based sensors are usually based on the energy (signal strength) or travel times (time of arrival) of signals between the devices. To determine the distance from energy measurements, the characteristics of the channel must be known; therefore this technique is very sensitive to the estimation of those parameters. In that sense, the measurements of travel times is preferred. If the two sensors have a common clock, the sensor receiving the signal can determine the time of arrival of the incoming signal that is time-stamped by the emitter sensor. In the absence of a common clock between the sensors, round-trip times can be measured in one of the sensors to estimate the distance between the two.

The UWB sensors used in this work are manufactured by Nanotron Technologies within their *swarm* product family, which provide accurate and relatively low-cost location capabilities. These UWB sensors provide distance measurements with an accuracy of 10 cm. A sample board is shown in Figure 2.6.

2.3 Conclusions

This chapter summarizes different 3D sensing modalities that are considered relevant for the purpose of localization of aerial robots in GPS-denied areas. They have been classified according to how they obtain depth information, whether it is a simple point-to-point distance estimation or a full depth image. Different sensors have been discussed, highlighting their main advantages and limitations regarding their applicability to small UAVs. Taking into account these factors and on-board requirements, the most interesting selected devices to be tested are stereo cameras, RGB-D cameras and UWB sensors.

Chapter 3

Robust Visual Odometry for UAVs

The first step of the work carried out towards this dissertation was obtaining localization estimations through the analysis of data provided by a 3D imaging sensor. During the past decades, different techniques have been proposed in order to tackle this estimation problem, generally by processing sensor data acquired at subsequent time instants. Vision-based odometry addresses the problem of estimating the motion of a robot by only using information from vision sensors Scaramuzza and Fraundorfer (2011).

One of the most popular techniques is known as registration, which is the process by which two data sets are brought into alignment. In particular, when dealing with 3D imaging sensors, it involves aligning 3D point clouds. Both passive and active optical sensors, such as stereo cameras or RGB-D cameras, can provide dense 3D point clouds at a high frequency. Then, subsequent point clouds can be matched in order to deduce the transformation between them and, consequently, the 6 Degrees-of-Freedom (DoF) motion of the sensor.

Apart from that, many state-of-the-art approaches are based on extracting interest points from RGB images, and matching them with those extracted in previous frames. By using the 3D information associated with such points, the transformation between frames can be estimated, and hence the motion of the robot Fraundorfer and Scaramuzza (2012).

In the presented approach, registration was first implemented, using only 3D point clouds to estimate the robot localization. Then, this processing pipeline was optimized

by adding information from the acquired RGB images. Afterwards, the matching was improved and also combined with information from inertial sensors. The following sections explain in detail the different approaches developed towards this dissertation, providing experimental results to validate them and discuss their applicability.

3.1 Registration

3.1.1 Depth-only Registration

Registration of 3D point clouds tries to find the rigid transformation that aligns them, so that they can be placed in a common coordinate system. Its output is usually a transformation matrix representing the rotation and translation that would have to be applied on one of the clouds in order to be perfectly aligned with the other.

Since no assumptions can be made about the shape of the objects in the scene, the chosen algorithm must be able to handle free-form surfaces. Automatic registration of free-form surfaces is usually performed using the Iterative Closest Point (ICP) algorithm Besl and McKay (1992). ICP is a method capable of providing a computationally efficient pose estimate in complex, unstructured environments. It has the advantages of locally solving the problem of matching and localization, being generic and potentially used for real-time applications Pomerleau et al. (2013).

The ICP algorithm takes as input two point clouds, the current reading and the reference, and tries to align them by iteratively computing the transformation between them. To be more accurate, the algorithm performs an iterative descent procedure which seeks to find a rigid geometric transformation that can be applied to the reading cloud such that it is most closely aligned with the reference cloud. Each point in the reading cloud can then be associated with its closest point in the reference cloud. The closeness of the alignment is most often based on a cost function such as the sum of mean square distances between point associations. At each iteration, the algorithm computes correspondences by finding closest points, and then minimizes the mean square error in position between the correspondences. The goal is obviously to optimize the alignment and therefore find a rigid transformation which minimizes

the cost function. Such transformation is a good estimate of the pose change between the acquisition times of the two clouds.

One major drawback of all ICP variants is that a good initial estimate of the transformation is required. Since it involves an iterative descent algorithm, an initial estimate is needed in order to converge to the global minimum; otherwise, ICP might get stuck in a local minimum. This is why, in general, ICP is only suitable if the distance between the point clouds is already small enough at the beginning. This insufficiency is often handled by estimating an initial transformation with methods or algorithms that converge to the global minimum but with lower precision. Following this procedure, ICP is then used to improve the result by performing a fine registration Salvi et al. (2007). In other scenarios, no guess of the initial transformation is needed because the difference between the measurements is sufficiently small, since the point clouds are generated at such a high rate (up to 30 Hz) that this precondition is met.

The simplest concept based on Besl and McKay (1992) is illustrated in Algorithm 1, which is often referred to as *Standard ICP*. The algorithm requires the reading point cloud A and the reference point cloud B in order to estimate the transformation T between them. Additionally, a rough estimate of the initial transformation T_0 can be considered. If this is not provided, and the true transformation between the point clouds is small enough, the initial transformation can be set to the identity matrix.

In every iteration, each of the N points of the reference point cloud B is transformed with the current transformation estimation T, and matched with its corresponding point from the reading point cloud A. Matches are rejected if the Euclidean distance between the pairs exceeds d_{max}, since points that are matched with a closer distance are less likely to be outliers. A weight value w_i is used to perform this rejection, setting its value to 0 or 1 accordingly. After solving the optimization problem and obtaining a new transformation T, the next iteration starts. The algorithm usually converges if the difference between the estimated transformation of two subsequent iterations becomes small enough, or if a predefined number of iterations is reached.

The *Standard ICP* algorithm is a point-to-point approach, which means that it tries to align all matched points exactly by minimizing their Euclidean distance. This does not take into account that an exact matching is usually not feasible, because

Input : Point clouds: $A = a_1, ..., a_M$ and $B = b_1, ..., b_N$ and Initial Transformation T_0
Output : Transformation T which aligns A and B
while *not converged* **do**
  for *each point* a_i *in cloud* A **do**
    $m_i = \text{FindClosestPointInA}(T \cdot b_i)$
    if $\|m_i - T \cdot b_i\|_2 \leq d_{max}$ **then**
      $w_i = 1$
    else
      $w_i = 0$
    end
  end
  $T = argmin_T \sum_i w_i \cdot \|m_i - T \cdot b_i\|_2^2$
end

Algorithm 1: Standard ICP

noise and/or different sampling in the two point clouds lead to pairs without perfect equivalence. To overcome this issue, some variants of ICP take advantage of surface normal information Chen and Medioni (1992) using a point-to-plane metric, which minimizes the sum of the squared distance between a point and the tangent plane at its corresponding point. Unlike the point-to-point metric, which has a closed-form solution, the point-to-plane metric is usually solved using standard non-linear least-squares methods. Although each iteration of the point-to-plane ICP algorithm is generally slower than the point-to-point version, better convergence rates have been found using the former. Moreover, when the relative rotation between the two input clouds is small, the non-linear least-squares optimization problem can be approximated with a linear one, thus computation times are lower Low (2004).

Yet another variant uses the normals from both the reading and the reference point clouds; this is usually known as plane-to-plane metric. In Segal et al. (2009), the authors propose a generalization of the ICP algorithm which takes into account the locally planar structure of both datasets, obtaining better and more robust results than using the aforementioned metrics.

Available implementations of ICP variants are included in the Point Cloud Library (PCL) Rusu and Cousins (2011), an open-source development library which contains state-of-the-art algorithms for filtering, feature estimation, surface reconstruction,

registration, model fitting or segmentation. Our first steps in this context have made use of such implementations, developing and testing the rest of steps described in this section.

Another issue that is worthy of consideration is that, in general, by continuously aligning consecutive 3D point clouds from the on-board sensor, the localization estimation is prone to error accumulation over time. In order to overcome this limitation, our approach considers building continuously an overall map as the point clouds are being registered. Then, subsequent point clouds are compared back to this map, instead of using the immediately previous point cloud. Thus, the relative pose is referred to the overall map that is being built, assuming that the initial pose of the robot is known. This persistent comparison back to the same map prevents the growth of errors in the pose estimation, as it usually happens in dead-reckoning approaches. However, memory usage increases as the map grows. In order to prevent this, downsampling configurations for this map have been developed consequently.

The developed processing pipeline includes several mechanisms to optimize the efficiency of the algorithm. First, the input point clouds are filtered in order to reduce the number of points, and hence decrease the computation time of each iteration. In particular, a voxel grid filter is employed, which downsamples the data by taking a spatial average of the points in the cloud. It divides the 3D space into voxels, or tiny boxes, of a specific configurable size. Then, in each voxel, all the points present are approximated with their centroid. This approach is a bit slower than approximating them by the center of the voxel, though it represents the underlying surface more accurately.

Another implementation involving efficiency in the processing pipeline is related to the matching step that takes place at each iteration of the algorithm. This matching between the filtered clouds using the current transformation parameters is based on a *kd*-tree, which produces an optimal search at a lower computation complexity.

Experimental Results

Depth-only registration results were obtained in the context of the project Perigeo, whose main goal was to investigate a wide range of space technologies using UAVs.

Figure 3.1: Scaled scenario of a landing site for testing depth-only registration.

One of these technologies was autonomous navigation for missions involving entry, descent and landing on asteroids using optical sensors; in particular, a Flash LIDAR prototype that is able to provide depth maps at video rate. To reproduce such mission, a scaled scenario was designed in order to recreate a descent trajectory through the use of a UAV.

The approach was to produce a section of an asteroid, representing the landing site and surroundings of asteroid Itokawa JAXA (2017), as it can be seen in Figure 3.1. The model had actual dimensions of 1.775m x 1.775m, thus achieving a scale factor of 1:45 with respect to the original asteroid section (80m x 80m). The Flash LIDAR sensor was emulated by an on-board RGB-D camera facing down, which provided the 3D point clouds used as data source for depth-only registration, as shown in Figures 3.2 and 3.3.

The experiments have been performed at the indoor testbed of the Center for Advanced Aerospace Technologies (CATEC). This facility is used to develop and test a wide range of technologies related to autonomous systems. The useful volume where tests can be conducted is a cube of 15x15x5 meters. The testbed houses an indoor localization system based on 20 Vicon cameras. It only needs the installation of passive markers on the objects to locate and/or track. This system is able to provide the position and attitude of each object in real-time with millimetric precision, and with very low latency in communications.

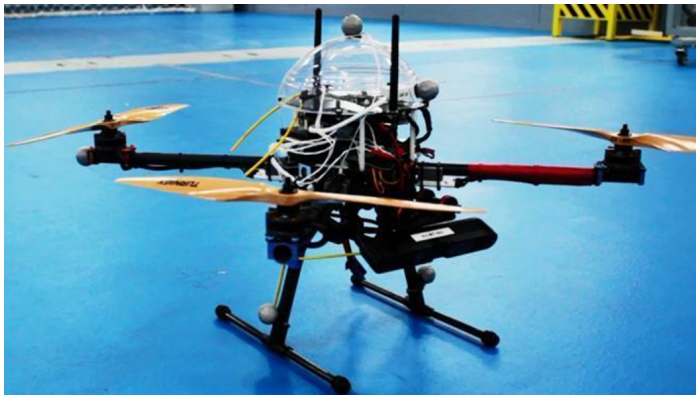

Figure 3.2: UAV used for testing depth-only registration.

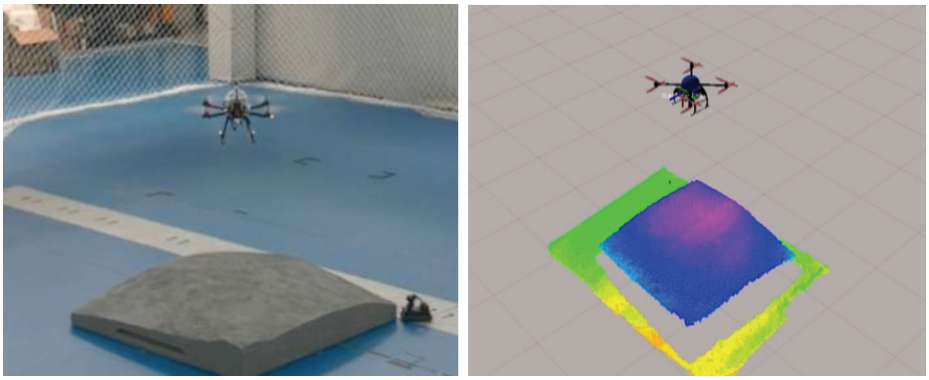

Figure 3.3: UAV flying over the asteroid scaled model and sample 3D data.

The UAV conducted two different trajectories representing two separate experiments:

- Descent trajectory: the UAV followed a downscaled (1:45) version of the nominal entry, descent and landing trajectory towards the actual landing site of a previous mission on the asteroid.

- Hovering trajectory: a different approach for validating space navigation algorithms is to explore an area of interest at a constant height. This trajectory can be seen as a hovering scenario for a lander and has the advantage of being easily implemented on a small multi-rotor UAV.

Ground-truth localization data for comparison were acquired by the motion capture system of CATEC's indoor testbed. In both experiments, the UAV automatically followed a predefined list of waypoints using the motion capture system to close the control loop, and then the point clouds were processed off-line in order to obtain the localization estimations shown in the figures. Only the initial ground-truth pose of the UAV was used in order to properly initialize the registration algorithm.

Figure 3.4 shows the results of the descent trajectory in position (x, y, z) and orientation $(roll, pitch, yaw)$, including both the ground truth values (gtX, gtY, gtZ, gtROLL, gtPITCH and gtYAW) and the registration algorithm outputs (estX, estY, estZ, estROLL, estPITCH and estYAW). Errors in position are relatively small (<10cm), as well as in orientation (<0.2°) during the whole trajectory. In the case of descent trajectories, initial errors in the point cloud matching process tend to converge. The overall map that the algorithm is continuously building usually represents the same surface of the small body. As the UAV gets closer to the asteroid model, the point clouds describe this surface with more detail, leading to better results in the matching process.

Even though the results are already promising, there is still significant room for improvement. The overall map that the algorithm continuously builds is initialized at the beginning of the trajectory, when the aerial robot is at the furthest distance from the scaled asteroid surface. At such distance (around 3m), possible UAV vibrations and 3D sensor noise greatly affect the initial point cloud comparisons, thus making the matching process much more challenging than when the UAV is closer to objects in general.

Figure 3.5 shows the results of the hovering trajectory also in position and orientation. Errors in position are similar than those in the case of the descent trajectory. In this case, the UAV flight height was approximately 1m above the scaled asteroid, which allowed the 3D sensor to provide depth measurements without much noise. However, at certain moments, the surface region acquired by the sensor appears mainly flat; hence the point cloud matching can be ambiguous since there are not enough depth details that could help the matching process.

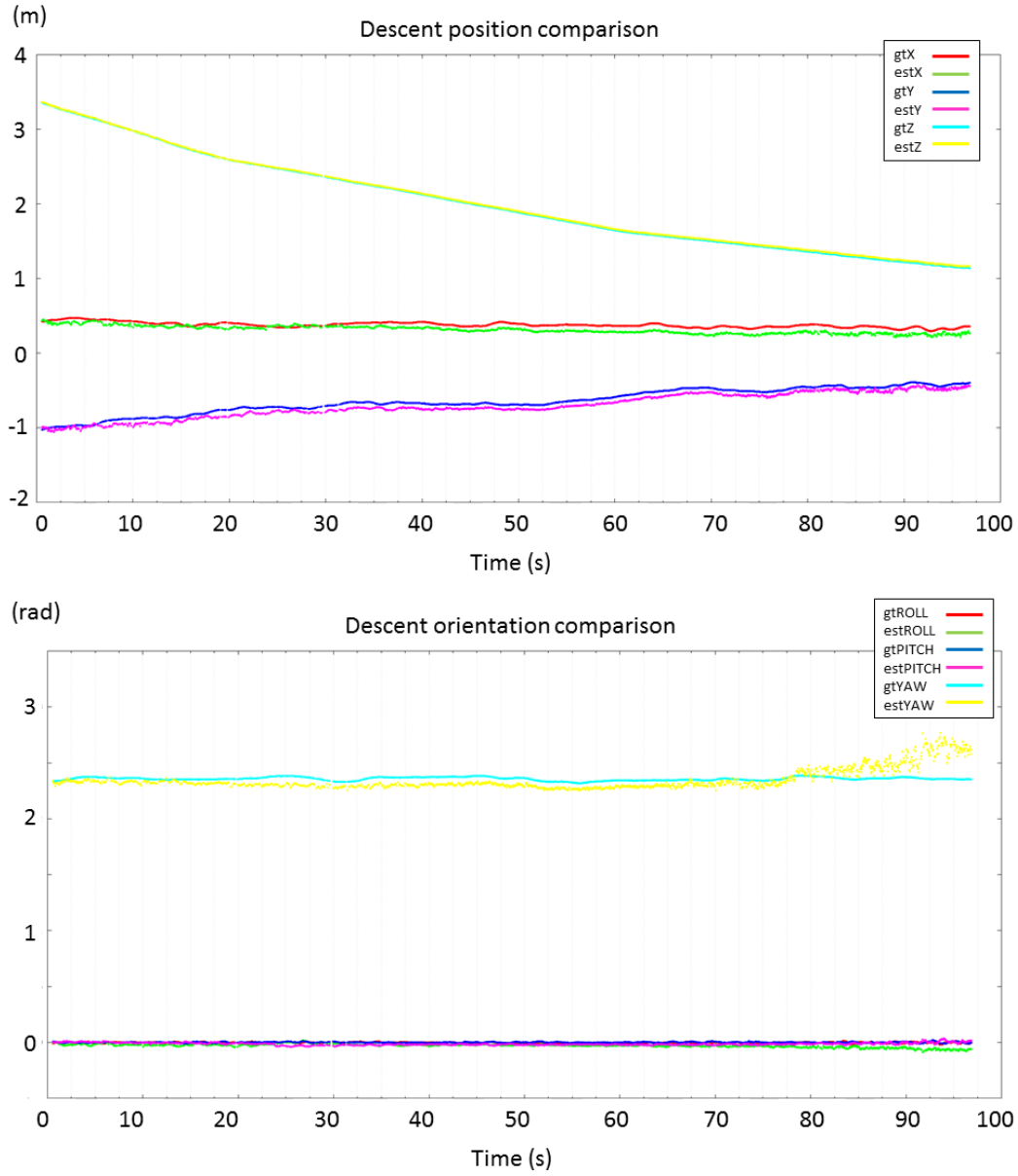

Figure 3.4: Pose estimation for the descent trajectory compared to ground-truth data.

Table 3.1 summarizes the root-mean-square (RMS) errors of pose estimations for both trajectories, in position and orientation.

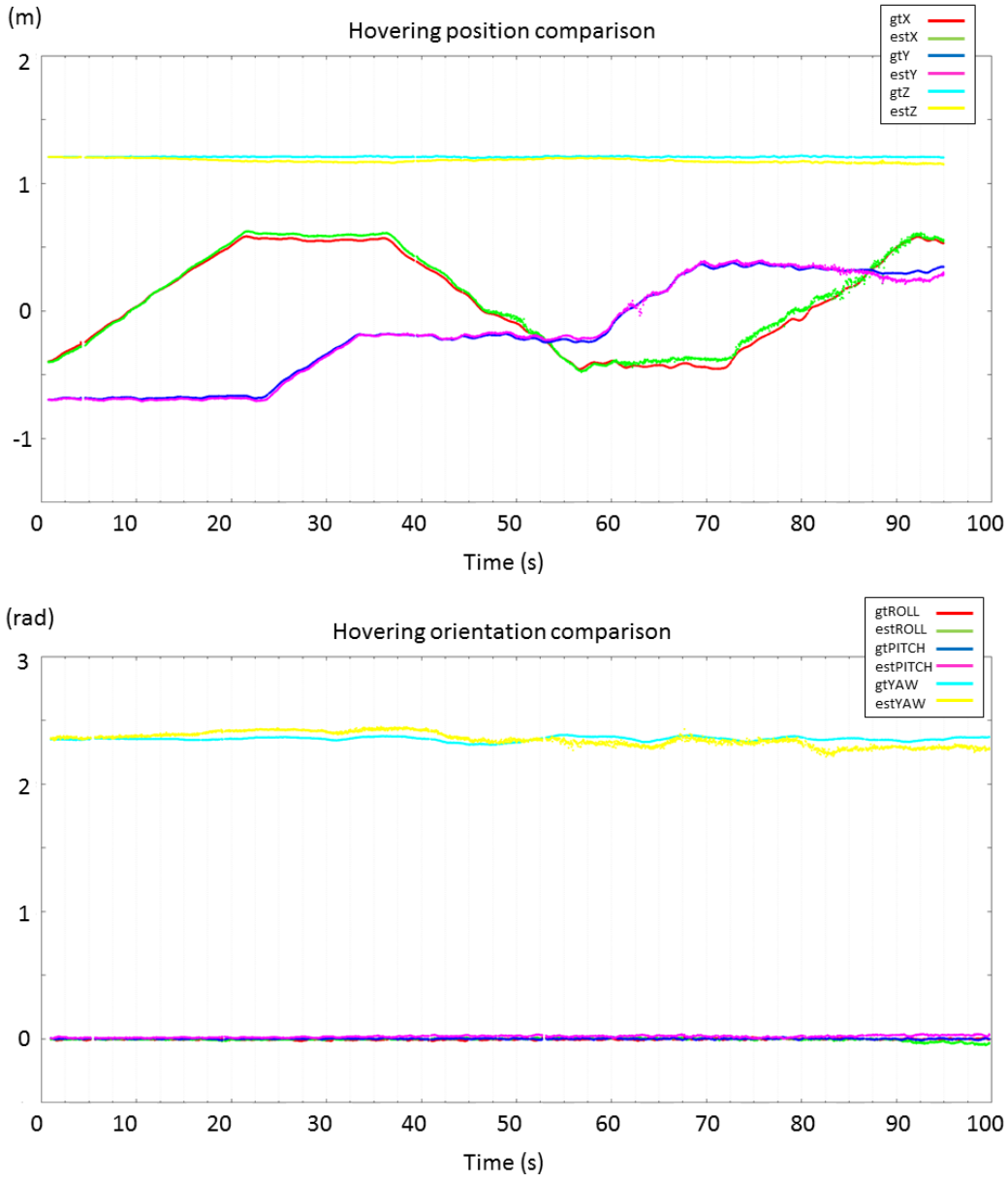

Figure 3.5: Pose estimation for the hovering trajectory compared to ground-truth data. (Left) Position plots. (Right) Orientation plots.

The results show the good performance of the proposed algorithm, but a number of shortcomings prevent this setup from constituting a generic solution for long-term UAV localization:

Table 3.1: RMS localization errors

	x (m)	y (m)	z (m)	*roll* (°)	*pitch* (°)	*yaw* (°)
Descent trajectory	0.024	0.042	0.063	0.007	0.002	0.111
Hovering trajectory	0.027	0.023	0.009	0.004	0.02	0.003

- The distance that the UAV traveled in the experiments was relatively short (<5m).
- The velocity at which the UAV performed the experiments was low (0.05m/s) in order to recreate scaled-down space exploration trajectories.
- Sensor data was underused. Apart from 3D point clouds, the RGB-D sensor provided color images that were not used in order to faithfully emulate the Flash LIDAR sensor under test. Besides, orientation data from the on-board IMU were not used either.

Nevertheless, these experiments helped to start the development of the software framework for the rest of technologies towards this dissertation, as well as gaining field experience in the setup of indoor flight testing.

3.1.2 Color-depth Registration

3D imaging sensors usually provide not only a point cloud of the scene, but also their associated color images. It seems reasonable to make use of such images, since adding RGB information to the software processing pipeline can improve the reliability of the registration process. In this way, we can take full advantage of all the information that the sensor provides (either a stereo or RGB-D camera), both visual and depth data.

Over the last decades, different ICP variants have been introduced Rusinkiewicz and Levoy (2001) proposing improvements in any or more of the stages of the algorithm, namely:

- Selecting some set of points in one or both point clouds.

- Matching the points from one cloud to samples in the other point clouds.
- Weighing the corresponding pairs appropriately.
- Rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs.
- Assigning an error metric based on the point pairs.
- Solving the optimization problem.

In order to reduce the computational cost of our processing pipeline, we have focused on optimizing the first stage of the ICP algorithm, i.e. selecting interest points on both the reading and the reference point clouds. At each frame, a set of visual features is extracted from the color image and mapped to their 3D locations using the associated point cloud. The set of 3D points is then matched across consecutive frames to estimate sensor pose increments since the last processed frame. Figure 3.6 shows an overview of the method.

The interest points are extracted using the Features from the Accelerated Segment Test (FAST) algorithm Rosten and Drummond (2006), which is a computationally efficient method for corner detection. According to this algorithm, a pixel is defined as a key-point if in a circle surrounding the pixel, N or more contiguous pixels are all significantly brighter than, or all significantly darker than the center pixel, as depicted in Figure 3.7. The extracted key-points are pixels which contain local information that ideally makes them repeatable across consecutive frames.

Depending on the environment, the scene in front of the sensor could be feature-less, or there might be strong-detailed regions where all the key-points are detected. Both cases pose problems in the subsequent stages of the registration method and may lead to poor transformation estimations. In order to overcome such issues, a bucketing technique has been adopted. The image is divided into several regions, and a fixed number of key-points is required to be extracted from each one. Our approach is based on subdividing the image into six buckets (two columns and three rows), hence providing a relatively homogeneous distribution of key-points in the image.

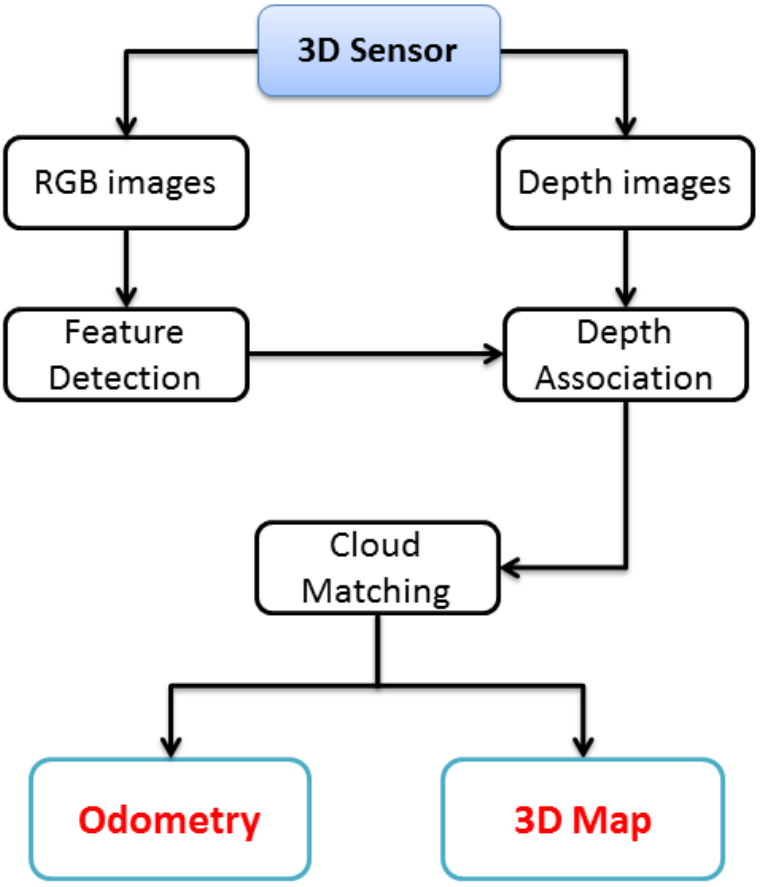

Figure 3.6: Schematic overview of the RGB-D registration pipeline.

The proposed approach is not purely based on visual feature matching though, in the sense that it does not try to find the same key-points across consecutive RGB images. Instead, the set of 2D key-points based on FAST features is enhanced with depth information to turn it into a set of 3D key-points. This is performed by directly using their corresponding depth values from the associated point cloud (in the case of stereo cameras, from the disparity map computed from the pair of images). Then, the strategy is to align consecutive sets of 3D key-points, assuming that the selected FAST features are repetitive enough to be found in consecutive frames.

3D imaging sensors provide depth information for most of the image pixels, but not all of them. This is a common situation in both stereo and RGB-D cameras.

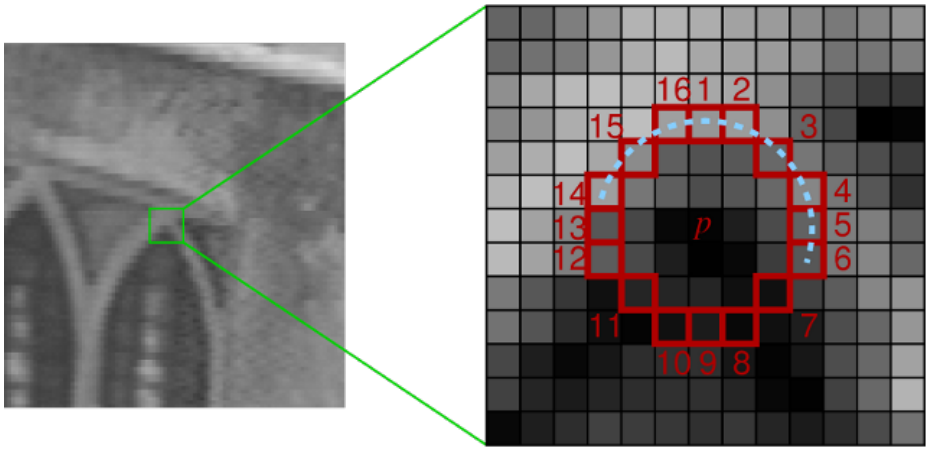

Figure 3.7: Pixel comparisons to determine the existence of a FAST key-point.

Depending on the scene and the physical properties of the materials in the environment, a 2D key-point might not have a corresponding depth value. Moreover, a 2D key-point might correspond to a very far location, thus the noise of the depth measurement can be significant. In these cases, our approach rejects the 2D key-points from the feature set. The fact that we might be throwing away possibly "good" key-points does not significantly impact on the performance of the algorithm, since the size of the key-points sets is relatively large (several hundreds of points).

Subsequent frames are analyzed in order to obtain sets of 3D key-points. Once a set of 3D key-points has been filtered, an alignment process is carried out to find the best fit between two sets. As in depth-only registration, the compared sets do not correspond with consecutive frames in order to mitigate the drift effect commonly present in odometry approaches. The sets to align are the current frame and the overall map that is continuously being built using the aligned 3D points. The overall goal is still to apply a transformation to one set to bring it as close as possible to the other, and ICP is used again to find such transformation. In this case, due to the sparse distribution of the 3D key-points, the point-to-point error metric is used in the iterative process:

$$E(T) = \sum_{i=1}^{N} \|Tu_i - z_i\| \tag{3.1}$$

where N is the size of the current set of 3D key-points u_i, z_i is the corresponding point in the map point cloud, and T is the transformation matrix composed of a rotation and translation.

ICP's computational load is greatly decreased with respect to our previous approach by reducing the size of the point clouds when only using the 3D key-points from both data sets. Other main drawbacks of ICP include its inability to deal with noise and outliers in the point clouds, or the absence of enough overlap between the clouds. Both cases are covered in the proposed approach, since the 3D key-point extraction process focuses on removing noise from distant points and possible outliers. Additionally, the high frame rate of 3D imaging sensors facilitates high overlap between the current point cloud and the overall map that is being built.

The sensor pose update is finally transformed to the robot body frame in order to obtain a localization pose update.

Experimental Results

Color-depth registration results were obtained in the context of a research stay performed at the Australian Centre for Field Robotics (ACFR) at the University of Sydney. RGB images and 3D point clouds are acquired from an RGB-D sensor facing slightly down on-board the Mars Analog Multi-Mode Traverse Hybrid (MAMMOTH) rover Reid et al. (2014), shown in Figure 3.8. Even though the robotic platform is not a UAV, the accuracy of the localization algorithm was successfully validated through an active articulation strategy for the rover, which relied on the correct on-board on-line robot localization and a sparse map building in order to estimate each wheel contact point in the terrain. The rover can then articulate its limb joints in order to actively conform to the terrain while traversing rough areas.

Ground-truth localization data for direct comparison were available only for orientation thanks to an on-board IMU; nevertheless, the following results describe the successful overall performance of the approach in both position and orientation. An example terrain point cloud generated from consecutive frames is shown in Figure 3.9, for which accurate localization is first needed in order to perform proper registration of the 3D point clouds.

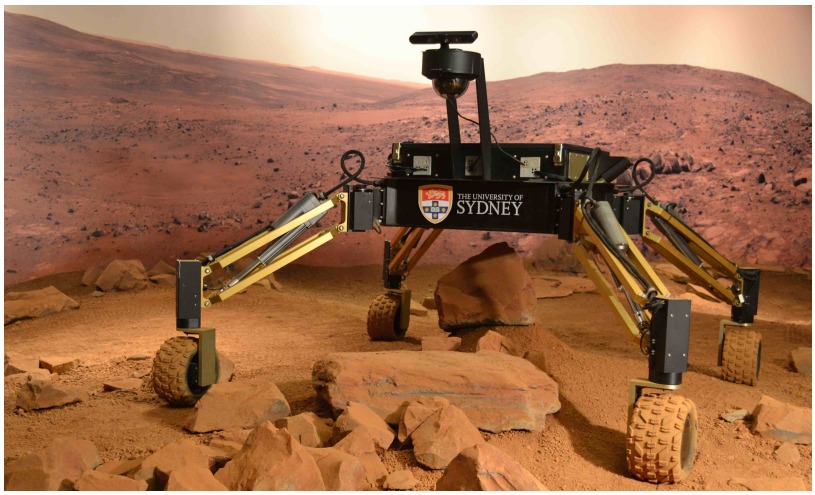

Figure 3.8: MAMMOTH rover with an RGB-D sensor on top.

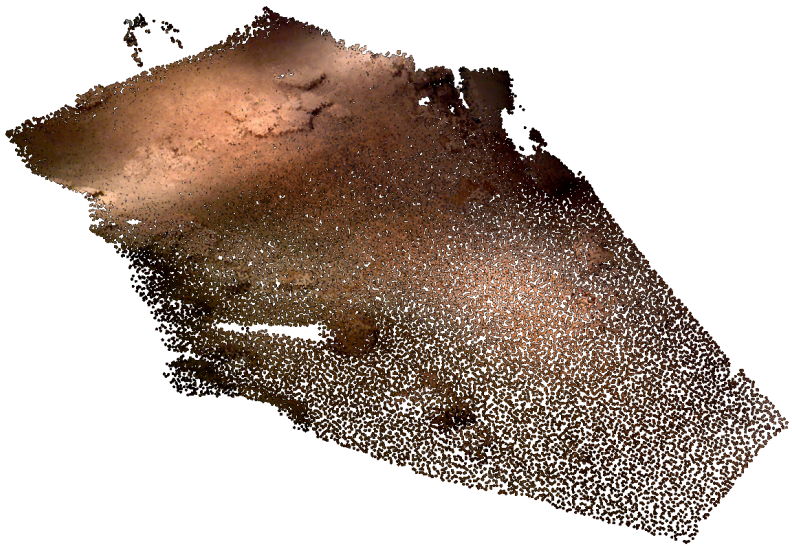

Figure 3.9: Map generated after driving the MAMMOTH rover over a section of the Mars Yard.

To validate the actively articulated suspension technique based on the RGB-D sensor based localization described before, the MAMMOTH rover is driven across a step obstacle and instructed to maintain a constant orientation and linear velocity while driving forward (along the x-axis relative to the world frame). The terrain that the MAMMOTH rover traverses is at the Sydney Powerhouse Museum's Mars Yard,

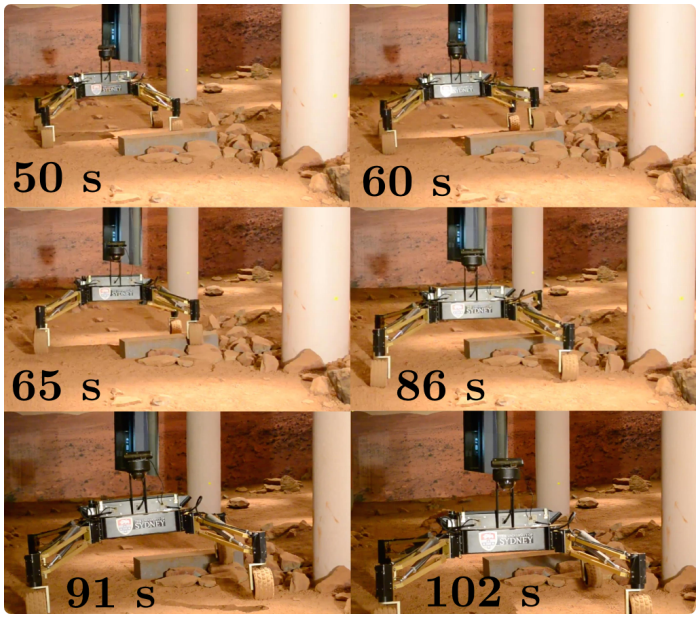

Figure 3.10: Sequence of images from the trial.

a 7x17 m space designed to resemble a section of the Columbia Hills region visited by the "Spirit" Mars Exploration Rover from the National Aeronautics and Space Administration (NASA).

The MAMMOTH rover moves over a bed of rocks that surrounds a 0.15m tall cinder block, shown in Figure 3.10. In all trials, the rover initially drives over a 4m section of relatively flat terrain so as to allow each of its wheels to be within the mapped terrain region. Once all wheels are within this region, the active articulation controller is activated by an operator. The desired rates to be kept for each trial are zero translational velocity along the y and z axes, and zero rotational velocity for roll, pitch and yaw angles. In the trial shown in the plots, the commanded speed of the rover is $v_x = 0.05m/s$. The world frame is defined at the base of the rover at the beginning of its traverse. Initially, the x and y positions of the world frame are coincident with body frame's x and y positions. The z position of the world frame is at the average initial height of each of the wheel contact frames relative to the body frame.

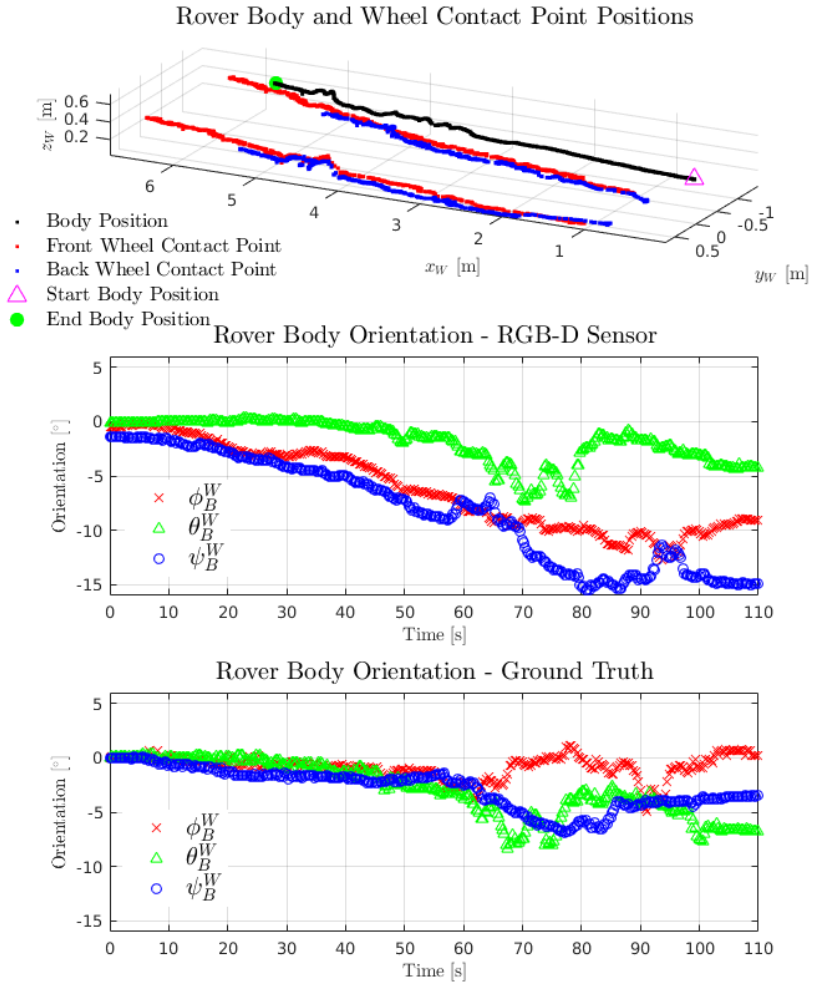

Figure 3.11: Position and orientation plots of the MAMMOTH rover from the trial.

During the trial, the rover traverses the block obstacle and articulates legs 1 and 2 as the left side of the rover passes over the obstacle. Figure 3.11 shows the RGB-D based position and orientation of the vehicle and the estimated contact points during the traverse. Additionally, orientation data from an on-board IMU is provided. An initial deviation in roll (ϕ) and yaw (ψ) is observed. Assuming the IMU orientation data to be a ground-truth, discrepancies in the estimated orientation are apparent. For the majority of the traverse, pitch (θ) data from the two localization systems are similar and indicate that there was a deviation no greater than 7° over the entire traverse. During the initial approach to the obstacle between 0 s and 50 s, it is

observed that the yaw deviates by 2.5° due to operator defined yaw re-alignment maneuvers. This yaw is approximately four times larger by 50 s according to the RGB-D sensor localization data, and may be caused by drift in the solution. A similar drift is seen along the roll axis between 10 and 70 s.

Experimental results demonstrate active terrain adaptation with the MAMMOTH rover. This method does not rely on any assumption about the kinematics of the movement, however it benefits from the fact that the rover motion is slow (on the order of centimeters per second). The basic functionality of this system is demonstrated, however improvements to the localization solution are still required, including the fusion of RGB-D sensor based estimations with IMU data in order to obtain more robust orientation estimations.

Nevertheless, these experiments contributed to the improvement of the processing pipeline and involved great and complex integration efforts in order to successfully run the algorithms in a different platform. Besides, the on-line control of the rover articulations entirely relied on the visual-based estimations that our approach was delivering, increasing the complexity of the experiments.

3.2 Visual-Inertial Odometry

Despite its popular use and recent developments regarding increased efficiency in 3D data processing, registration algorithms such as ICP are still computationally expensive, and highly sensitive to significant differences in the clouds to register, as the correct points correspondence is more difficult to find.

Using image features or key-points as the basic source of information can alleviate the computational load of the algorithm, but additional improvements are needed in order to achieve reliable localization. Fusing pose estimations with information provided by inertial sensors such as gyroscopes and accelerometers (through an on-board IMU) helps reducing uncertainties in the robot localization when fast motions are present, as it is usually the case when working with UAVs. Inertial sensors are prone to substantial drift in position estimates when compared with purely vision

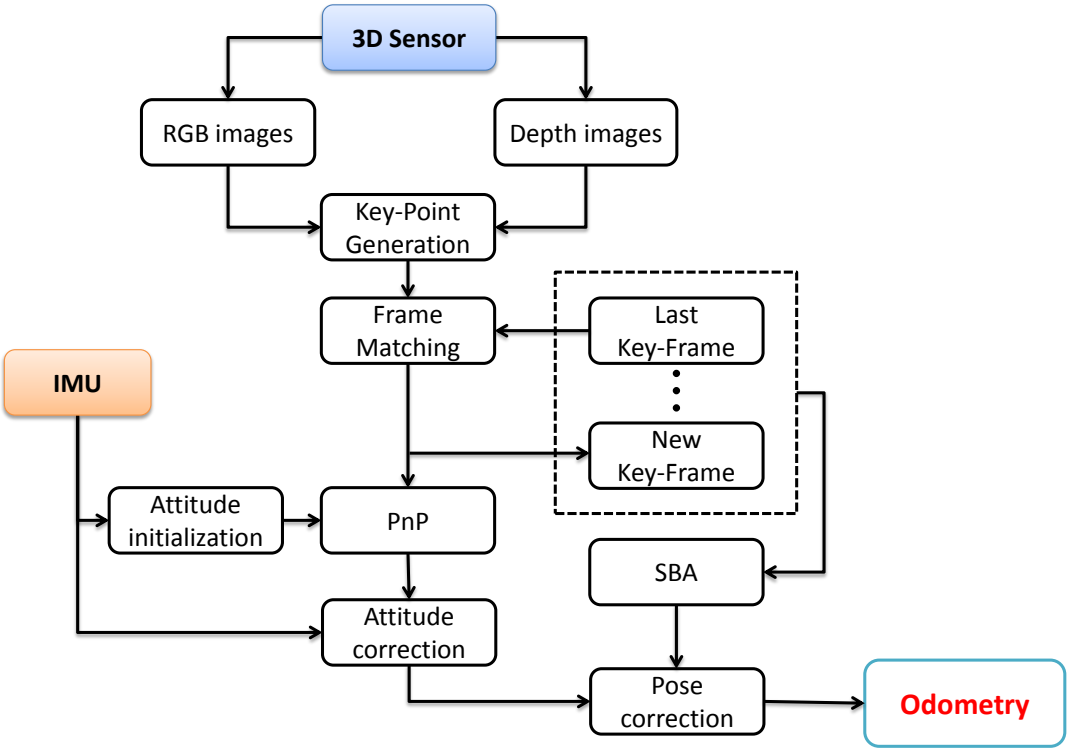

Figure 3.12: Schematic overview of our visual-inertial odometry pipeline.

approaches, but our approach only takes limited information from such sensors in order to obtain robust estimations.

This section focuses on the visual-inertial motion estimation algorithm presented in Figure 3.12. The proposed odometry system combines images, 3D point clouds, inertial sensors, pose estimation, key-framing and sparse bundle adjustment in a loose-coupling filter in order to estimate a reliable and accurate localization at short term. The algorithm is publicly available[1].

The approach is able to work using either a stereo camera or an RGB-D sensor. If a stereo camera is used, disparity images would need to be generated first using common approaches such as Semi-Global Matching Hirschmuller (2005). RGB-D cameras directly provide depth images.

[1]http://wiki.ros.org/viodom

3.2.1 Feature Detection

In the case of stereo cameras, at each frame, after removing the image distortion from the stereo pair, a set of key-points is first selected from both the left and right images. These interest points are visual features extracted using the FAST algorithm as in the previously explained approach. When working with RGB-D cameras, instead of focusing on left and right images, the algorithm input is directly the color image. The FAST algorithm is also used in this case for finding the key-points set.

In both sensing approaches, sometimes the scene is very homogeneous, or there are many features grouped in a specific region in the image, and this could later lead to inaccurate transformation estimations. In order to overcome such situations, a bucketing technique similar to the previous approach has been adopted. On one hand, a minimum number of key-points is required to be extracted from equally divided regions of the images, i.e. buckets. On the other hand, a maximum number of key-points is defined in order to select only the most powerful ones from each bucket, according to the FAST detector definition. The combination of both parameters allows a homogeneous distribution of the strongest N key-points in the images. Our approach is based on subdividing the image into six buckets (two columns and three rows). The difference between the originally detected features and the ones detected using the bucketing technique can be seen in Figure 3.13.

3.2.2 Feature Description

In the processing pipeline, the next step involves finding correspondences between the sets of features from different images. To achieve this, some kind of feature point description is needed in order to compare and find similarities between the feature sets, and for that, the general purpose Binary Robust Independent Elementary Features (BRIEF) descriptor Calonder et al. (2012) is used. This descriptor is very efficient since it is based on simple intensity difference tests, and the similarity evaluation is done using the Hamming distance. Together with the computational efficiency, BRIEF descriptors are especially interesting for feature tracking under small visual perturbations, a very usual situation in visual odometry. Other descriptors that

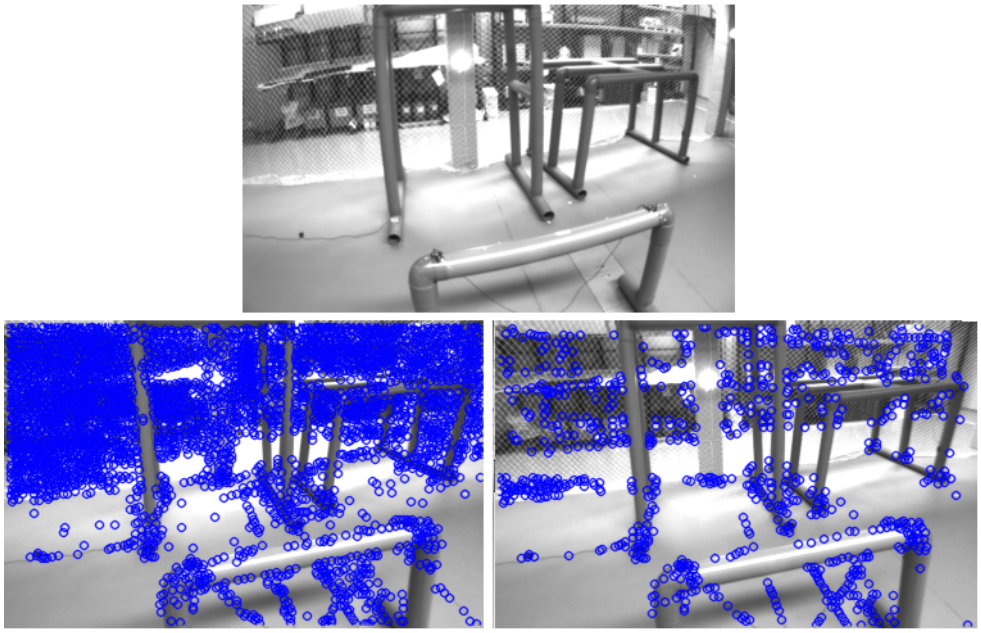

Figure 3.13: Features detected from the original image (top) on a sample image before (bottom left) and after (bottom right) applying the bucketing technique.

include rotation into their formulation have been tested, such as the Oriented FAST and Rotated BRIEF (ORB) Rublee et al. (2011), the Binary Robust Invariant Scalable Keypoints (BRISK) Leutenegger et al. (2011) or the Fast Retina Keypoint (FREAK) Alahi et al. (2012), but no significant improvements have been detected for local matching, hence the one with the smallest computational impact (BRIEF) is finally used in this approach.

3.2.3 Frame Matching

In the case of using stereo cameras, correspondences between the sets of feature descriptors from the left and the right images have to be found. In order to do that, a robust matcher has been implemented. This matcher rejects correspondences that are not symmetrical (i.e. are not found from the left to the right image and vice-versa), and also makes use of epipolar geometry to significantly reduce the matching candidates, speeding up the process. An example of stereo matching is shown in Figure 3.14.

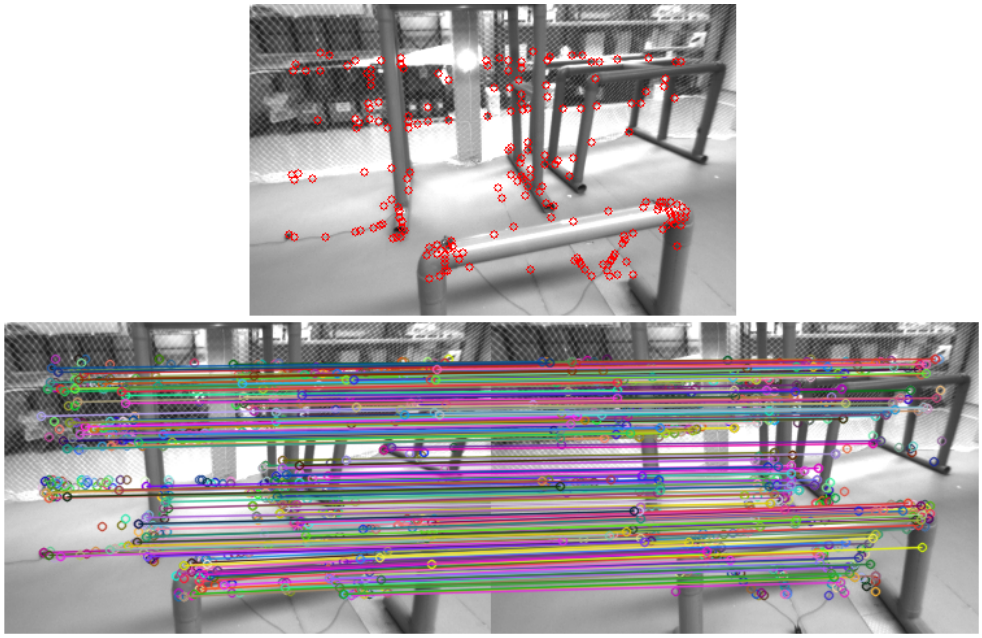

Figure 3.14: Feature matching between left and right images from the stereo pair.

The depth associated with each matched key-point is stored to enhance the set of 2D key-points for subsequent processing stages.

In the case of RGB-D cameras, the set of key-points obtained in the previous sub-section is directly enhanced with their associated depth information. As in the previous section, if the depth value is not available or is too far from the camera, the key-point is rejected.

Once a set of robust key-points enhanced with depth information has been found from either the stereo pair of images or the RGB-D frame, the next step of the algorithm focuses on finding correspondences between the sets of key-points from consecutive frames. In the case of stereo, only left images are used hereafter. This allows to find a subset of even more robust key-points which survive across frames. The same matcher used for finding correspondences between the stereo pair images is also used between consecutive frames.

If enough matches are found between the key-points from the previous and the current frames, this set of matches is used to solve a Perspective-n-Point problem (PnP). This consists in the pose estimation from n 3D-to-2D point correspondences,

i.e. the estimation of a camera pose (6 DoF: rotation and translation) with respect to a coordinate frame in which 3D points and their 2D projections in the camera are provided, given the camera calibration parameters. The 3D points are the coordinates of the key-points from the previous frame, while the 2D projections are their matched correspondences in the current frame. Hence the resulting pose provides an estimation of the camera motion between both frames.

Given a set of n 3D points in a world reference frame and their corresponding 2D image projections, as well as the calibrated intrinsic camera parameters, the pose of the camera with respect to the world frame is calculated as follows. The perspective model for the camera is:

$$s\ p_c \ = \ K\ [R|t]\ p_w \tag{3.2}$$

where $p_w = [x\ y\ z\ 1]^T$ is the homogeneous world point, $p_c = [u\ v\ 1]^T$ is the corresponding homogeneous image point, K is the matrix of intrinsic camera parameters (which are f_x and f_y for the scaled focal lengths, γ for the skew and (u_0, v_0) is the principal point), s is a scale factor for the image point, and R and t are the desired 3D rotation and 3D translation of the camera (extrinsic parameters) that are to be estimated. This leads to the following equation for the model:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.3}$$

The assumption made in most solutions is that the camera is already calibrated. For each solution to PnP, the chosen point correspondences cannot be coplanar. In addition, PnP can have multiple solutions, and choosing a particular solution would require post-processing of the solution set. Furthermore, using more point correspondences can reduce the impact of noisy data when solving the problem.

A commonly used solution to the PnP problem exists for $n = 3$, which is called Perspective-Three-Point (P3P) Gao et al. (2003). However, with only 3 correspondences, P3P yields many solutions, so a fourth correspondence is used in practice

to remove ambiguity. Let P be the center of projection for the camera, A, B and C the 3D world points with corresponding image points u, v and w. Let $X = |PA|$, $Y = |PB|$, $Z = |PC|$, $\alpha = \angle BPC$, $\beta = \angle APC$, $\gamma = \angle APB$, $p = 2cos\alpha$, $q = 2cos\beta$, $r = 2cos\gamma$, $a' = |AB|$, $b' = |BC|$ and $c' = |AC|$. This forms triangles PBC, PAC and PAB from which we obtain the equation system for P3P:

$$Y^2 + Z^2 - YZp - b'^2 = 0 \tag{3.4}$$

$$Z^2 + X^2 - XZq - c'^2 = 0 \tag{3.5}$$

$$X^2 + Y^2 - XYp - a'^2 = 0 \tag{3.6}$$

It is common to normalize the image points before solving P3P. Solving the P3P system results in four possible solutions for R and T. The aforementioned fourth world point D and its corresponding image point z are then used to find the best solution among the four.

As previously mentioned, using more point correspondences helps to reduce the impact of noisy data, which will generally be our case when using 3D imaging sensors on-board UAVs. In particular, the Efficient PnP (EPnP) algorithm Lepetit et al. (2008) has been used in our approach, which provides an efficient implementation for solving the PnP problem for $n \geq 3$. This method is based on the notion that each of the n points (which are called reference points) can be expressed as a weighted sum of four virtual control points. Thus, the coordinates of these control points become the unknowns of the problem. It is from these control points that the final pose of the camera is solved.

As an overview of the process, first note that each of the n reference points in the world frame, p_i^w, and their corresponding image points, p_i^c, are weighted sums of the four controls points, c_j^w and c_j^c ($j = 1...4$) respectively, and the weights are normalized

per reference point as shown below. All points are expressed in homogeneous form.

$$p_i^w = \sum_{j=1}^{4} \alpha_{ij} c_j^w \tag{3.7}$$

$$p_i^c = \sum_{j=1}^{4} \alpha_{ij} c_j^c \tag{3.8}$$

$$\sum_{j=1}^{4} \alpha_{ij} = 1 \tag{3.9}$$

From this, the derivation of the image reference points becomes

$$s_i \, p_i^c = K \sum_{j=1}^{4} \alpha_{ij} c_i^c \tag{3.10}$$

The homogeneous image control point has the form $c_j^c = \begin{bmatrix} x_j^c & y_j^c & z_j^c \end{bmatrix}^T$. Rearranging the image reference point equation yields the following two linear equations for each reference point:

$$\sum_{j=1}^{4} \alpha_{ij} f_x x_j^c + \alpha_{ij}(u_0 - u_i) z_j^c = 0 \tag{3.11}$$

$$\sum_{j=1}^{4} \alpha_{ij} f_y y_j^c + \alpha_{ij}(v_0 - v_i) z_j^c = 0 \tag{3.12}$$

Using these two equations for each of the n reference points, the system $Mx = 0$ can be formed where $x = \begin{bmatrix} c_1^{c^T} & c_2^{c^T} & c_3^{c^T} & c_4^{c^T} \end{bmatrix}^T$. The solution for the control points exists in the null space of M and is expressed as

$$x = \sum_{i=1}^{N} \beta_i v_i \tag{3.13}$$

where N is the number of null singular values in M and each v_i is the corresponding right singular vector of M. N can range from 1 to 4. After calculating the initial

coefficients β_i, the Gauss-Newton algorithm is used to refine them. The R and T matrices that minimize the reprojection error of the world reference points, p_i^w, and their corresponding actual image points p_i^c, are then calculated.

This solution has $O(n)$ complexity and works in the general case of PnP for both planar and non-planar control points.

3.2.4 Attitude Correction

The PnP problem estimates a 6 DoF motion between frames, including translation and rotation. Since the UAV counts with an on-board IMU, our odometry system is able to read its data and integrate the roll and pitch angles with the purely visual estimation before and after the PnP algorithm. They are first introduced in the algorithm to provide an initial estimation for the rotation that took place between the frames under comparison. Afterwards, they are used again to compensate for the final angles that the algorithm outputs.

3.2.5 Key-Framing

In order to partially mitigate the effect of cumulative errors usually found in odometry approaches, a key-framing approach has been adopted. Thus, new key-frames are produced when the feature tracking flow exceeds a given threshold. This is a measure of how much the image shifted since the last creation of a key-frame, by comparing the pixel positions of the key-points.

At that moment, the current image key-points descriptors, their 3D estimated position and the robot position/orientation are stored. Subsequent frames will compute the transformation with respect to the last key-frame (and its pose) instead of the immediately preceding image, so that errors are only accumulated with the introduction of new key-frames, instead of with each frame.

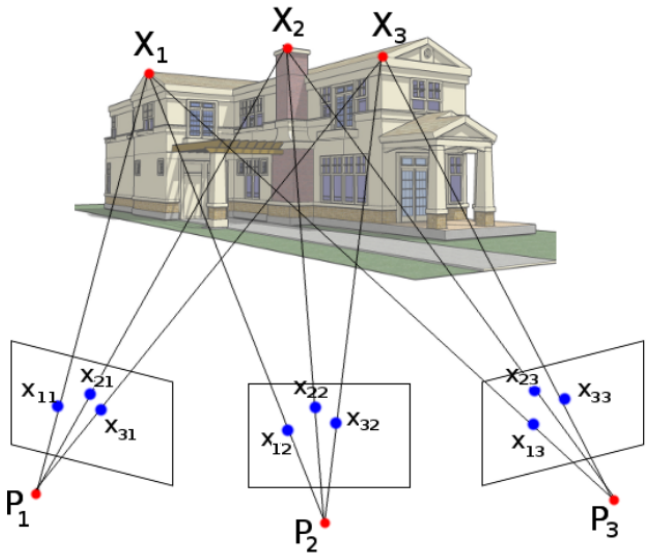

Figure 3.15: Bundle adjustment projection example.

3.2.6 Sparse Bundle Adjustment

In general, Bundle Adjustment (BA) is the problem of refining a visual reconstruction, and is almost invariably used as the last step of every feature-based multiple-view reconstruction vision algorithm to obtain jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates. It is optimal because the parameter estimates are found by minimizing a cost function that quantifies the model fitting error, and it is jointly optimal because the solution is simultaneously optimal with respect to both structure and camera variations Triggs et al. (1999). Its name refers to the bundles of light rays originating from each 3D feature and converging on each camera's optical center, which are adjusted optimally with respect to both the structure and viewing parameters. Figure 3.15 shows an example using three views.

BA boils down to minimizing the reprojection error between the image locations of observed and predicted image points, which is expressed as the sum of squares of a large number of nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least-squares algorithms. Of these, Levenberg–Marquardt (LM) has proven to be one of the most successful due to its ease of implementation and its use

of an effective damping strategy that lends it the ability to converge quickly from a wide range of initial guesses.

BA amounts to jointly refining a set of initial camera and structure parameter estimates for finding the set of parameters that most accurately predict the locations of the observed points in the set of available images. More formally, assume that N 3D points $\mathbf{X}_i$ are seen in M views $\mathbf{P}_j$, and let $\mathbf{x}_{ij}$ be the projection of the i-th point on image j. Let v_{ij} denote the binary variables that equal 1 if point i is visible in image j, and 0 otherwise. Assume also that each camera $\mathbf{P}_j$ is parameterized by a vector $\mathbf{a}_j$, and each 3D point $\mathbf{X}_i$ by a vector $\mathbf{b}_i$. Bundle adjustment minimizes the total reprojection error with respect to all 3D point and camera parameters, specifically

$$\min_{\mathbf{a}_j,\,\mathbf{b}_i} \sum_{i=1}^{N} \sum_{j=1}^{M} v_{ij}\, d(\mathbf{Q}(\mathbf{a}_j,\, \mathbf{b}_i),\ \mathbf{x}_{ij})^2 \tag{3.14}$$

where $\mathbf{Q}(\mathbf{a}_j,\, \mathbf{b}_i)$ is the predicted projection of point i on image j, and $d(\mathbf{x},\, \mathbf{y})$ denotes the Euclidean distance between the image points represented by vectors $\mathbf{x}$ and $\mathbf{y}$. Clearly, BA is by definition tolerant to missing image projections and minimizes a physically meaningful criterion.

However, due to the large number of unknowns contributing to the minimized reprojection error, a general purpose implementation of the LM algorithm (such as MINPACK's *lmder* Moré et al. (1980)) incurs high computational costs when applied to the minimization problem defined in the context of BA.

Fortunately, the lack of interaction among parameters for different 3D points and cameras results in the underlying normal equations exhibiting a sparse block structure. Our approach makes use of an efficient solution called Sparse Bundle Adjustment (SBA) Lourakis and Argyros (2009), which exploits this sparseness by employing a tailored sparse variant of the LM algorithm that leads to considerable computational gains. SBA is generic in the sense that it grants the user full control over the definition of the parameters describing cameras and 3D structure. Therefore, it can support virtually any manifestation or parameterization of the multiple view reconstruction problem, such as arbitrary projective cameras, partially or fully intrinsically calibrated

cameras, exterior orientation (i.e. pose) estimation from fixed 3D points, refinement of intrinsic parameters, etc.

In our case, let us consider a series of key-frames, each of them with their associated camera poses and key-points with 3D coordinates and 2D image projections. The idea is to improve the computation of the camera motion between key-frames according to the sets of observations. SBA is able to optimize not only the motion estimations, but also reduce the errors that were introduced into the 3D coordinates of the key-points. Our approach only considers the last K produced key-frames (K=10 in the experiments), and applies two subsequent methods in order to allow on-line computation of the pose refinement:

- It only considers the key-points that are seen and matched across at least $K/2$ consecutive key-frames.

- It applies a bucketing technique to the surviving key-points, keeping the best 3 key-points from 28 image buckets (four rows and seven columns).

3.2.7 Ground Plane Estimation

Another feature that was considered in this work is the estimation of the ground plane, given the particularities of the sensor arrangement and the testing environment. The 3D imaging sensor is mounted slightly pointing downwards, hence the point cloud can easily be used to estimate the flight height of the UAV, as an altimeter does, assuming that enough ground scene is within the field of view of the camera. The point cloud is first downsampled in order to keep the computational efficiency of the whole pipeline. A simple plane segmentation has been performed, i.e. find all the points within the point cloud that support a plane model. The detected planes are filtered by their normal vector, since we are looking for horizontal planes up to a maximum normal angle (10° in our approach). The estimated height of the UAV is an input to the pose correction regarding Z, which is averaged with the purely visual height using a configurable weight factor (0.5 in our experiments).

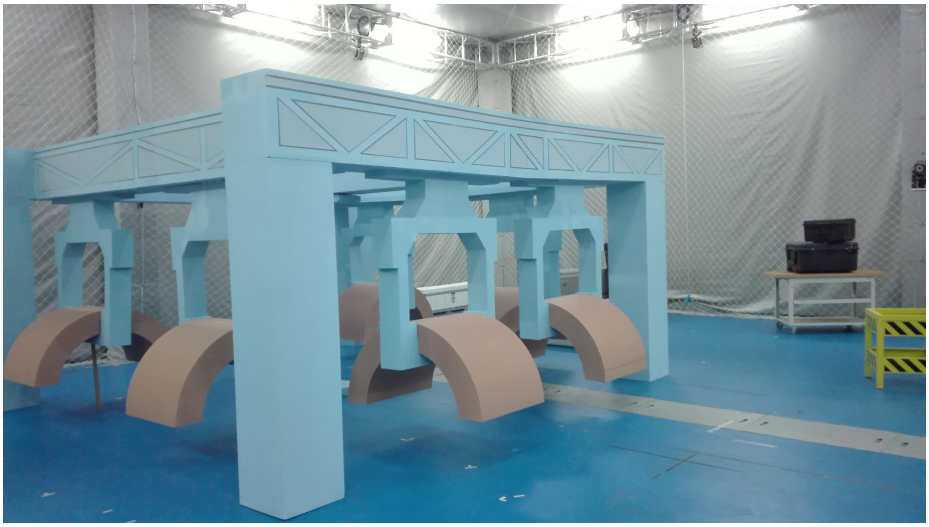

Figure 3.16: CATEC testbed with a mockup scenario.

The final camera pose estimation is transformed from the sensor frame to the aerial robot body frame, and used to calculate the new relative position and orientation estimations with respect to the initial pose.

3.2.8 Experimental Results

In order to validate our approach using the odometry approach described in this section, a UAV has performed a flight in CATEC's indoor testbed within a scenario recreating an industrial facility, as shown in Figure 3.16. The visual odometry has been used as the localization estimation input in order to close the control loop, and the UAV was teleoperated through a joystick to command small increments in position (x, y or z) and orientation (only yaw). The main objective of the experiment was to demonstrate the suitability and accuracy of the approach in real-time.

This section provides experimental results from the estimated localization compared with ground-truth data obtained from the testbed tracking system. The UAV used to demonstrate our approach is shown in Figure 3.17. The main sensor on-board the platform is the RGB-D camera Orbbec Astra, which is facing forward and slightly tilted down (25°). The trajectory followed by the UAV according to ground-truth data is shown in Figure 3.18.

Figure 3.17: The UAV with the RGB-D sensor at the front.

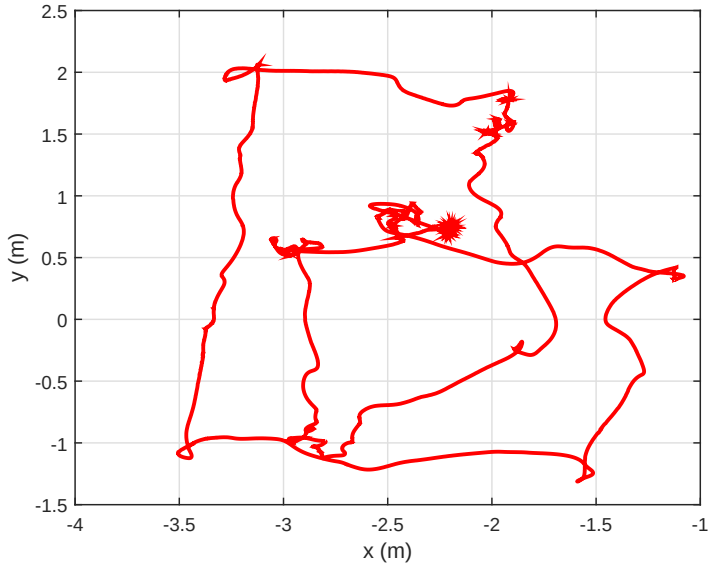

Figure 3.18: Ground-truth UAV trajectory in XY during the experiment.

The estimated UAV localization provided by the visual odometry algorithm during the experiment can be seen in Figure 3.19. The estimated position and yaw angle

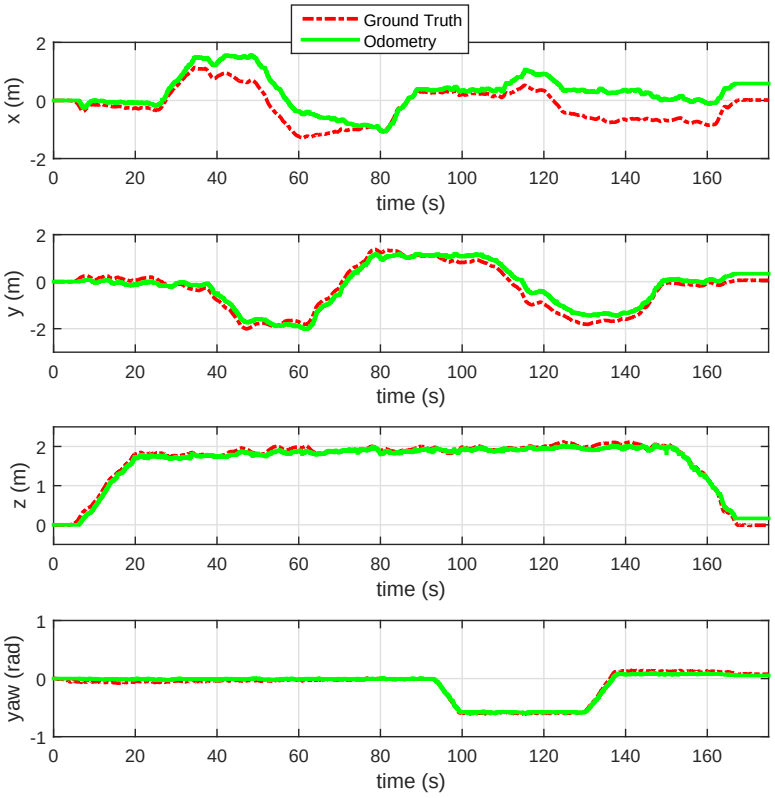

Figure 3.19: Estimated UAV position and orientation.

during the experiment are shown: the red dashed line corresponds to the UAV ground-truth captured by the motion capture system, while the green dotted line is the estimated UAV position and orientation using the proposed visual odometry. Roll and pitch angles were acquired directly from the on-board IMU and hence are not shown in the plots, since they are directly integrated into the algorithm.

As expected, the odometry exhibits some drift, though it provides a smooth estimation which tremendously helps the control algorithms achieve a stable performance. Figure 3.19 also shows how the odometry is consistent throughout the whole flight trajectory, taking into account the relatively adverse conditions of the flight within the indoor testbed for the algorithm, since there are almost no visual features in the floor or the mockup scenario, and sometimes the nearest objects are more than five

meters far ahead. The absolute values of the errors in the estimations are shown in Figure 3.20, along with the computed RMS errors for each axis, which are also shown in Table 3.2.

Errors in x are significantly higher than those in y, probably due to the fact that the commanded trajectory towards the mockup scenario consisted of displacements in the x axis. A reactive local planner was also running during the experiment, which prevented the UAV from colliding with nearby obstacles. This local planner trims the trajectory, but closing the control loop using the proposed visual odometry caused small perturbations that led to small incremental errors. Estimations in x recovered around the middle of the trajectory, but then experienced a similar behavior due to a new collision situation, and then could not recover until the end of the experiment. Estimations in y, however, exhibit a robust and stable performance. Note how RMS error in z is very small (less than 10cm) thanks to the ground plane estimation based on the point clouds, used as an altimeter input.

In order to quantitatively assess the performance of the odometry algorithm, we have tested the data from the experiment against three popular approaches that were cited in Section 1, which also make use of RGB-D data: CCNY-RGBD[2] Dryanovski et al. (2013), RTAB-Map[3] Labbe and Michaud (2014) and RGBD-SLAM[4] Endres et al. (2012). Even though our approach does not constitute a SLAM solution, we chose to compare our localization estimations with these approaches because it can be considered that some "mapping" is performed given how we integrate key-frames and the SBA, which takes into account the 3D points and their projections from the past several key-frames.

The plots in Figure 3.21 show the performance of the same flight data processed off-line using the three tested approaches, and Table 3.2 includes general RMS errors compared to those from our approach. Notice that we did not fine tune the parameters of the algorithms, so a better accuracy could be achieved. Besides, these approaches do not make use of any additional estimation for localization as we do with Z using the ground plane estimation.

[2]http://wiki.ros.org/ccny_rgbd
[3]http://wiki.ros.org/rtabmap
[4]http://wiki.ros.org/rgbdslam

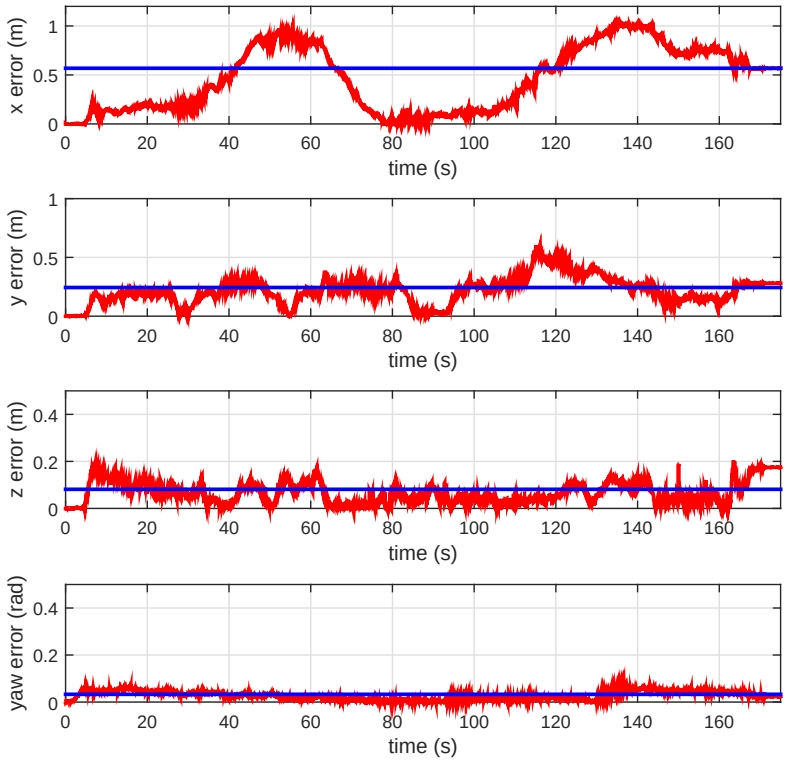

Figure 3.20: Localization errors in UAV estimation.

Table 3.2: RMS localization errors

	x (m)	y (m)	z (m)	*yaw* (rad)
CCNY-RGBD	**0.19**	0.34	0.43	0.04
RTAB-Map	0.45	0.29	0.22	0.09
RGBD-SLAM	1.19	2.75	1.85	0.85
Our approach	0.44	**0.24**	**0.08**	**0.03**

The plots show overall good performances of CCNY-RGBD and RTAB-Map. Particularly critical are the moments in which the UAV is resting on the ground, since the RGB-D camera is pointing slightly down and hence there is little sensor information, and such information is usually at the top of the image and at a far

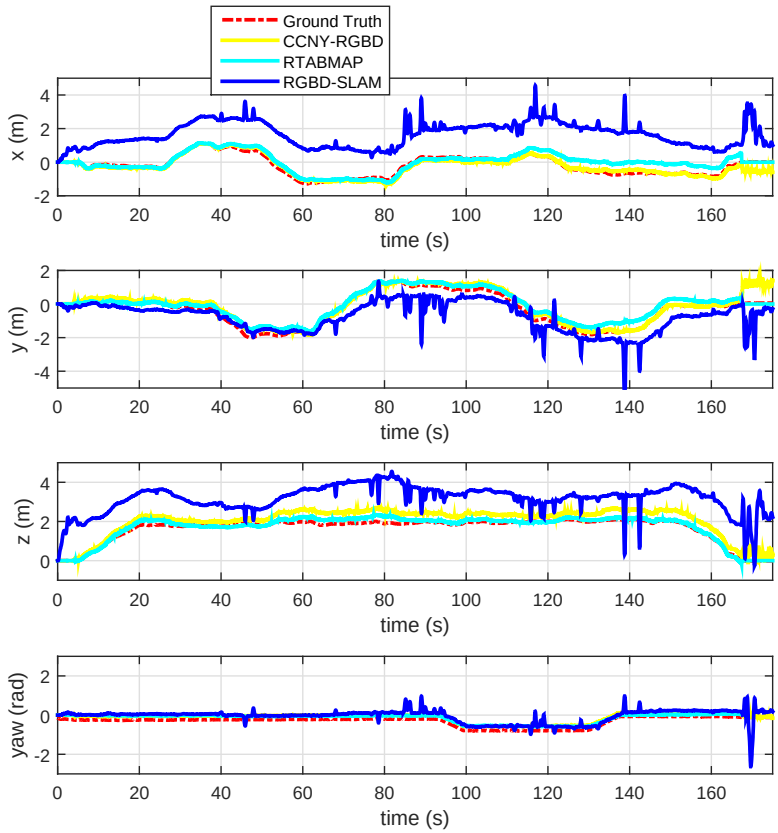

Figure 3.21: Estimated UAV position and orientation using other approaches.

distance. This often results in wrong pose estimations that could lead to unexpected behaviors. CCNY-RGBD experiences this issue at the very end of the experiment, when the UAV landed and the estimation suddenly jumps. RTAB-Map seems to be the most robust in this aspect; however we had to manually reset its odometry estimation as soon as the UAV took off, otherwise no estimation at all would have been published since it could not find a connection between consecutive frames while the UAV was still on the ground during the off-line test. RGBD-SLAM heavily suffers from this situation, as it accumulates error at the beginning (especially in z) and was not able to recover during the rest of the flight. Our approach is able to successfully cope with this situation thanks to the configuration of a minimum number of matches between the current frame and the last key-frame, and a maximum distance at which

depth information can be considered valid in the algorithm. Apart from that, Fig. 3.21 reveals sudden pose estimation changes in RGBD-SLAM due to wrong loop closures within the map. Those abrupt changes would surely represent a problem in the control loop, most likely resulting in an undesired flight termination. Our algorithm provides a smooth estimation over the whole duration of the experiment.

Table 3.2 shows how most of the RMS errors in x, y, z and yaw of the compared algorithms are similar to those of the proposed odometry approach (the smallest values are in bold). Hence our odometry is able to provide an accuracy at the level of the state of the art, while keeping robustness and computational efficiency as key features. Table 3.3 includes values for the mean frequency and corresponding mean processing times at which new pose estimations are published, using the UAV on-board i7 Linux computer. Our approach is able to deliver pose estimations approximately every 60ms, which makes our approach suitable for working at around half the frame rate of the RGB-D sensor. The other approaches may publish pose estimations to the ROS ecosystem at a higher rate than the one shown in the table, but internally they just copy the same values to the output message until a new estimation is computed.

Table 3.3: Pose estimation speed

	Average Freq (Hz)	Frame processing time (ms)
CCNY-RGBD	8.1	124
RTAB-Map	6.2	161
RGBD-SLAM	5.8	173
Our approach	**15.8**	**63**

3.3 Conclusions

This chapter focuses on the evolution of approaches that have been developed, implemented and validated regarding vision sensors in order to obtain reliable UAV pose

estimations at short term. In particular, stereo or RGB-D cameras have been used as the main sensor of the vision-based system.

Registration was first tested, using only 3D point clouds and later adding information from the RGB images. Experience gained through these initial approaches led towards a more optimized approach that combines images, point clouds, inertial measurements from the on-board IMU, pose estimation algorithms, key-framing and sparse bundle adjustment in a loose-coupling filter. This allows to obtain a localization accuracy at the level of the state of the art, while providing much more robust estimations and using less computational resources from the on-board processor.

In general, experimental results show the efficiency and reliability of the proposed odometry algorithm, through its on-line use to localize an aerial robot during a real flight. However, the proposed approach accumulates drift over time and this can make the system unreliable for autonomous long-term operation. Additional improvements are still needed in order to achieve safe and robust autonomous operations using aerial robots.

Motion estimation algorithms based on matching visual features do not usually perform as well in regions with few visual features. In large open areas, the visible structure is often far beyond the reliable range of the 3D sensor, and as a result, the system actually performs better in cluttered environments. Handling these challenges will likely require the integration of other sensors in order to complement each other's capabilities. Additional sensing modalities can reduce, but not eliminate, state estimation failures.

Chapter 4

Multi-Modal Sensor Fusion for Long-Term Localization

Different sensors on a robotic system can provide different information about its environment. Then, the information that an aerial robot obtains using its sensors needs to be combined in order to determine where it is. Sensor fusion can be thought as the problem of combining data from different sensors into one unified view of the environment. However, sensors are noisy, and there are usually many things that cannot be sensed directly.

As stated in Section 1, visual odometry systems are able to provide reliable and accurate localization estimations at short term. However, odometry will eventually diverge after a period of time. Instead of using a loop closing approach, the fusion of visual information with other types of 3D data acquired from the aerial robot environment is proposed.

UWB-based sensing provides distances to fixed locations that can be used to absolutely localize the aerial robot, in a similar fashion as GPS works. Several UWB beacons can be installed in the scenario and another UWB sensor on-board the UAV to gather distance measurements between the different beacons. However, the measurement noise typically present in radio-based distance computation and, mostly, the outliers produced by radio multi-path effect normally prevent its use as a stand-alone localization system when high accuracy is required. Moreover, this system

does not provide enough information to constitute a full localization system, since the data provided does not include bearing information.

By fusing both sensing modalities, the reliable short-term position estimation based on odometry is combined with distances to fixed UWB sensors to correct its cumulative drift. Both benefit from each other when integrated in order to obtain fast and error-bounded localization estimations. Furthermore, the high efficiency of the implemented algorithms makes them suitable for real-time localization in the usually constrained equipment that can be mounted on-board an aerial vehicle. This particular combination is the main strength of the approach, in which two different sensor fusion strategies have been developed and tested using UAVs; they are explained in detail later in this chapter. First, an overview of how the aerial robot state estimation is handled from a probabilistic point of view is presented.

4.1 State Estimation

Generally speaking, the interaction of a robot and its environment can be modeled as a dynamic system, in which the robot can manipulate its environment by choosing controls, and in which it can perceive its environment through sensor measurements. In probabilistic robotics, the dynamics of the robot and its environment are characterized in the form of two probabilistic laws:

- the state transition distribution, which characterizes how state changes over time, normally as the effect of a robot control (e.g. a movement), and
- the measurement distribution, which characterizes how measurements are governed by states.

Both laws are probabilistic, accounting for the inherent uncertainty in state evolution and sensing.

The estimation of the state of a mobile robot is usually achieved by the Bayes filter Thrun et al. (2005). The localization problem consists in finding the robot state at time t, $\mathbf{x}_t$, given the last measurement z^t and a prior state $\mathbf{x}_{t-1}$. The Markov

assumption holds, i.e. a state is a complete summary of the past. Hence the Bayes filter is recursive, since $\mathbf{x}_t$ is calculated from $\mathbf{x}_{t-1}$.

The Bayes filter algorithm possesses two essential steps. It first calculates a belief over the state $\mathbf{x}_t$ based on the prior belief over state $\mathbf{x}_{t-1}$ and a control input u_t. This is usually called the *prediction* step. The second step is called the *update*, in which the algorithm multiplies the belief $\mathbf{x}_t$ by the probability that the measurement z^t may have been observed. The result is normalized since the resulting product is generally not a probability.

In order to compute $\mathbf{x}_t$ recursively, the algorithm requires an initial belief at time $t = 0$. If one knows the value of $\mathbf{x}_0$ with certainty, it should be initialized with a point mass distribution that centers all probability mass on the correct value, assigning zero probability anywhere else. If one is entirely ignorant about the initial value $\mathbf{x}_0$, it may be initialized using a uniform distribution over the domain of $\mathbf{x}_0$.

Bayesian filters are implemented in several different ways. Each of them relies on different assumptions regarding the state transition, measurement probabilities and the initial belief. In general, exact techniques for state calculation are not available and hence the state has to be approximated. Finding a suitable approximation is usually a challenging problem. When choosing an implementation, several properties have to be considered.

- Computational efficiency: some approximations allow computing state beliefs in time polynomial in the dimension of the state space, for example linear Gaussian approximations. Particle-based techniques have an *any-time* characteristic, enabling them to trade-off accuracy with computational efficiency.

- Accuracy of the approximation: some techniques can approximate a wider range of distributions more tightly than others. For example, linear Gaussian approximations are limited to unimodal distributions. Particle representations can approximate a wider range of distributions, but the number of particles needed to attain the desired accuracy can be large.

- Ease of implementation: it depends on a variety of factors, such as the form of the measurement probability and the state transition probability. Particle

representations often yield surprisingly simple implementations for complex non-linear systems, which is one of the reasons for their recent popularity.

The next two sections describe the two techniques that have been used in the context of this dissertation, which represent examples of probably the two most popular families of recursive state estimation techniques, both derived from the Bayes filter: Gaussian filters and particle filters.

4.2 Gaussian Filters for State Estimation

Gaussian techniques share the basic idea that beliefs of the robot state are represented by multi-variate normal distributions. The density over the variable $\mathbf{x}$ is characterized by two sets of parameters:

- The mean μ, which is a vector that possesses the same dimensionality as the state $\mathbf{x}$.
- The covariance Σ, which is a quadratic matrix that is symmetric and positive-semidefinite. Its dimension is the dimensionality of the state $\mathbf{x}$ squared (i.e. the number of elements depends quadratically on the number of elements in the state vector).

Representing the belief by a Gaussian distribution has important implications. Most importantly, Gaussians are unimodal, that is, they possess a single maximum. This is characteristic of many tracking problems in robotics, in which the belief of the state is focused around the true state with a small margin of uncertainty. However, they are a poor match for many global estimation problems in which many distinct hypotheses exist.

Probably the best studied Gaussian technique for implementing Bayes filters is the Kalman filter (KF) Kalman et al. (1960), which has been widely used for filtering and prediction in linear systems. At time t, the belief is represented by the mean μ_t and the covariance Σ_t. Apart from the Markov assumption, the following three properties must hold:

- The state transition probability $p(\mathbf{x}_t|u_t, \mathbf{x}_{t-1})$ must be a linear function in its arguments with added Gaussian noise:

$$\mathbf{x}_t = A_t\mathbf{x}_{t-1} + B_t u_t + \epsilon_t \tag{4.1}$$

- The measurement probability $p(z_t|\mathbf{x}_t)$ must also be linear in its arguments with added Gaussian noise:

$$z_t = C_t\mathbf{x}_t + \delta_t \tag{4.2}$$

- The initial belief in $t = 0$ must be normal distributed.

The assumptions of linear state transitions and linear measurements with added Gaussian noise are rarely fulfilled in practice. For example, a robot that moves with constant translational and rotational velocity typically moves on a circular trajectory, which cannot be described by linear state transitions. This observation, along with the assumption of unimodal beliefs, renders plain Kalman filters inapplicable to all but the most trivial robotic problems.

The extended Kalman filter (EKF) overcomes one of these assumptions: linearity. Here the assumption is that the state transition and the measurement probabilities are governed by non-linear functions:

$$\mathbf{x}_t = g\left(\mathbf{x}_{t-1}, u_t\right) + \epsilon_t \tag{4.3}$$

$$z_t = h\left(\mathbf{x}_t\right) + \delta_t \tag{4.4}$$

A Gaussian projected through the non-linear state transition function g is typically non-Gaussian. Linearization approximates g by a linear function that is tangent to g at the mean of the Gaussian. By projecting the Gaussian through this linear approximation, the result is also a Gaussian. The same applies to the multiplication of Gaussians when the measurement function h is involved. Tangents are linear, making the filter applicable. There exist many techniques for linearizing non-linear functions. EKFs utilize Taylor expansion, which involves calculating the first derivative of the

target function, and evaluating it at a specific point. The result of this operation is a matrix known as the Jacobian.

The EKF is one of the most popular tools for state estimation in robotics Mao et al. (2007); Purvis et al. (2008); Rullan-Lara et al. (2011). Its strength lies in its simplicity and computational efficiency. The EKF owes its computational efficiency to the fact that it represents the belief by a multi-variate Gaussian distribution. A Gaussian is a unimodal distribution, which can be thought of as a single guess, annotated with an uncertainty ellipse. In many practical problems, Gaussians are robust estimators.

An important limitation of the EKF arises from the fact that it approximates the state transition and measurement functions using linear Taylor expansions. In virtually all robotic problems, these functions are non-linear. The accuracy of this approximation depends on two main factors: the degree of non-linearity of the functions that are being approximated, and the degree of uncertainty which is directly related to the width of the probability distribution (the larger the uncertainty, the higher the error introduced by the linearization).

In our incremental approach for achieving robust long-term localization of UAVs, an EKF has been first used to fuse the information provided by the visual odometry algorithm and UWB-based distance measurements. Although localization based on visual odometry will eventually diverge due to cumulative errors, it can be used as a short-time predictor of the UAV motion (*prediction*). The UWB sensor readings can then constrain the odometry drift (*update*). Thus, using the odometry as motion prior helps filtering UWB noise and detecting/removing measurement outliers. At the same time, the UWB-based update helps to remove the cumulative errors in the odometry, building an accurate and stable localization system both together. Figure 4.1 shows an overview of the method.

The UWB sensor on-board the UAV computes the distance to all the sensors at a lower operating frequency than that of the visual odometry system, and integrates the information into an EKF described below. As the robot moves, the UWB sensor periodically sends out a query, and any other sensor within range responds by sending a reply. Since each sensor transmits a unique ID number, distance readings are automatically associated with the appropriate tags, so the data association problem is

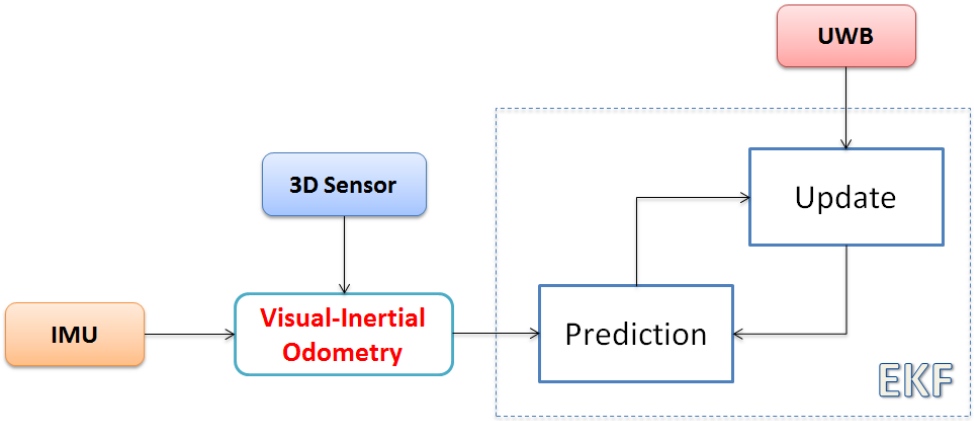

Figure 4.1: Schematic overview of the EKF approach.

trivially solved. Another key advantage is that the robot can estimate the distance to each responding sensor even when they are not within line of sight. This is very useful in situations where visual-based methods usually fail, such as poorly illuminated scenes or highly dynamic environments.

The range measurements and odometry priors are all integrated into a simple EKF that allows estimating the robot's position and velocity with the following state vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{p} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ v_x \\ v_y \\ v_z \end{bmatrix} \tag{4.5}$$

where $\mathbf{p}$ and $\mathbf{v}$ refer to the robot position and velocity respectively. Orientation data is not considered in the EKF since, as stated before, the UWB-based measurements used in the update step do not provide bearing information. Hence, visual odometry is the only source of information for estimating the aerial robot orientation. As a reminder, roll and pitch angles are integrated directly from the on-board IMU, while yaw is estimated purely from visual information.

4.2.1 Prediction

The visual odometry system provides the estimated velocity of the robot periodically, $\mathbf{vo} = [vo_x, vo_y, vo_z]^t$. This information is used in the prediction stage of the EKF to constrain the robot localization according to the following expression:

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \Delta t \cdot \mathbf{v}_{t-1} \tag{4.6}$$

$$\mathbf{v}_t = \mathbf{vo} \tag{4.7}$$

Eventually, the odometry system might not be able to provide an estimation, for instance, due to the lack of texture in the scene. In such a case, the state is predicted based on random walk.

4.2.2 Update

Given the distance d_i measured from the robot to a beacon with known position $\mathbf{b}_i$, the following constraint can be applied to the robot position:

$$d_i = ||\mathbf{p}_t - \mathbf{b}_i|| \tag{4.8}$$

This constraint can be easily integrated into the update stage of the EKF, following the equation:

$$z_t = h(\mathbf{x}(t)) \tag{4.9}$$

where $z_t = d_i$ and $h(\mathbf{x}(t)) = ||\mathbf{p}_t - \mathbf{b}_i||$.

This filter update is applied to all the UWB measurements received within the last period of time according to the filter operating frequency (50ms in our experiments). If more than one measurement to a specific beacon are available, the distance is averaged. Note that the filter does not need distances to several beacons in order to perform a filter update; the distance to a single sensor is enough. This is possible thanks to the state prior provided by the prediction stage, which is expected to be accurate and stable at short-term. This is a great advantage with respect to state-of-the-art

UWB-based localization systems, which usually wait until the distances to three or more sensors are measured.

4.2.3 Outlier Rejection

Distance measurements based on radio sensors are subject to frequent outliers when performed indoors. These outliers are caused by multi-path and signal reflection in walls and structures, and introduce significant errors in the localization.

The odometry prior (EKF prediction) provides a short-term accurate and reliable estimation of the robot's position, so the EKF measurement residual $\mathbf{y}_t = \mathbf{z}_t - h(\mathbf{x}_t)$ can be used to decide whether a range measurement is an outlier or not. Thus, if the residual of a range measurement is above a threshold r_{th}, the measurement is determined as an outlier with high probability and the filter will reject the data. This threshold has been set to $r_{th} = 2m$ in our experiments.

4.2.4 Experimental Results

In order to validate the aforementioned approach, which makes use of visual odometry and UWB-based measurements, an aerial robot has performed a relatively long flight (around 7 minutes). The flight has taken place in an indoor controlled scenario where several UWB devices (six sensors) have been installed. The main objective of the experiment is to demonstrate the suitability of the approach during regular operations.

The experiments took place at the indoor testbed of CATEC, whose motion capture system provided ground-truth localization data.

Prior to the discussion of the experimental results, the setup describing all the elements involved in the experiment is presented:

- Six UWB sensors used as beacons, installed across the indoor testbed at different locations, homogeneously distributed also at different heights. These locations are previously known by the system.

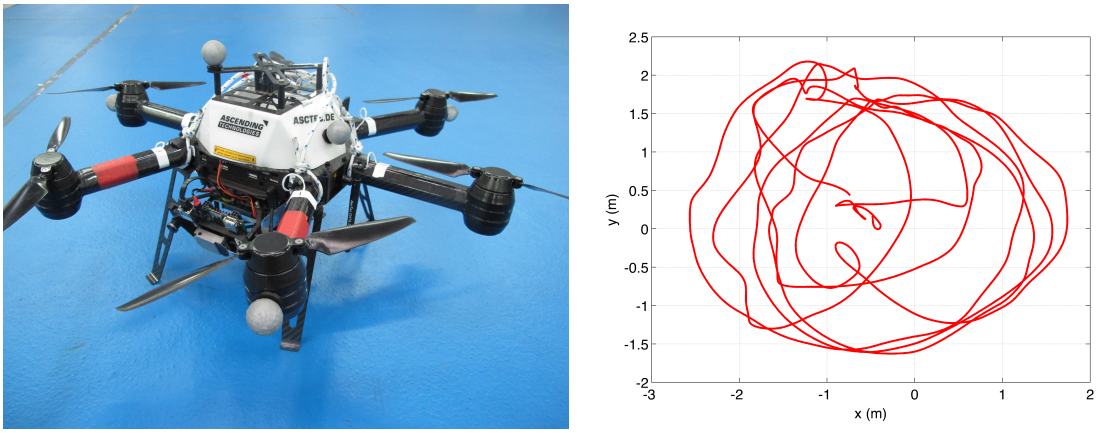

Figure 4.2: UAV with the VI-sensor at the front (left) and ground-truth trajectory in the XY plane (right).

- An aerial robot with the following on-board sensors: a VI-sensor (stereo camera), an IMU and a UWB sensor in order to compute point-to-point distances (see Figure 4.2).

The aerial robot was manually piloted through the testbed area at different heights. The ground-truth trajectory followed by the vehicle is also presented in Figure 4.2. This is a 75 meters long trajectory in which the robot was piloted within the indoor testbed, flying at different altitudes.

The estimated UAV position during the experiment can be seen in Figure 4.3, which includes the estimated fused localization (blue solid line) together with the raw visual odometry output (green dotted line), both compared to the ground-truth position (red dashed line). As expected, the odometry system slowly diverges through time, while the proposed method integrating radio-based measurements follows the ground-truth with small deviations. Only position results are analyzed since radio-based sensing does not provide orientation measurements.

It can be also seen in Figure 4.3 how the odometry is consistent most of the time, taking into account the relatively adverse conditions of the flight within the indoor testbed: low texture (there are almost no visual features in the floor) and high distance to 3D scenes (sometimes up to ten meters far ahead).

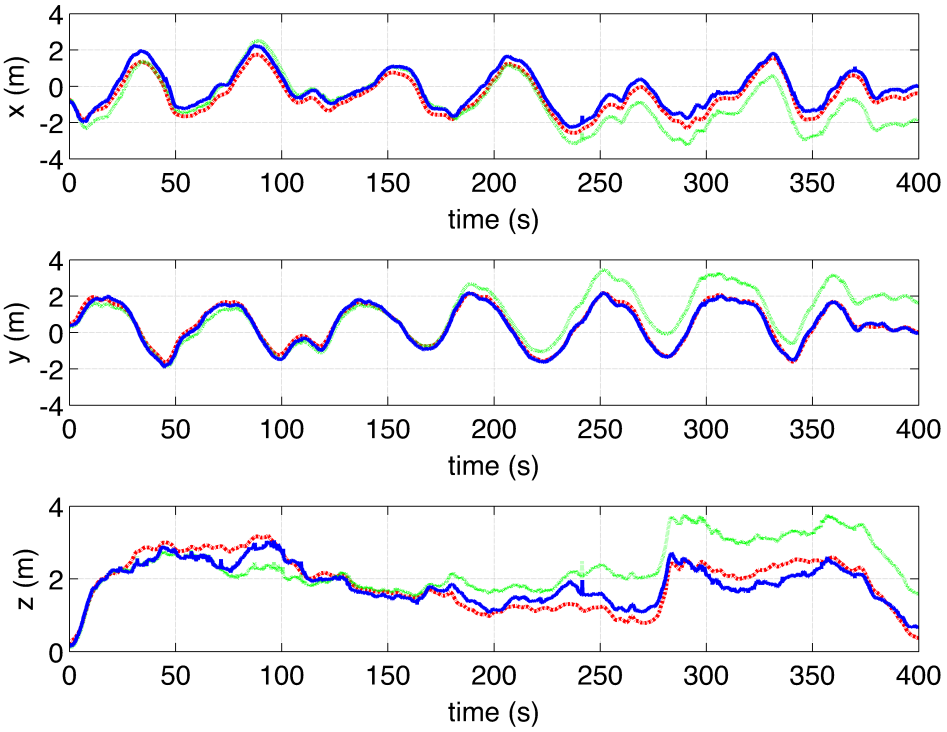

Figure 4.3: Estimated UAV localization (ground-truth in red, visual odometry in green, proposed approach in blue).

Figure 4.4 shows the estimation error of the proposed approach in X, Y and Z separately. The RMS error for each localization component is also presented. It can be seen how the RMS error is around 0.25m in Y and Z, and 0.4m for X.

As it was previously observed, the flight conditions are far from ideal regarding the visual odometry approach. The scenario floor exhibits a very homogeneous scene, and the introduced objects to recreate a 3D environment are sometimes very far from the aerial vehicle (up to ten meters). Hence, disparity computation results in higher errors that lead to possible wrong matches across frames. Nevertheless, the results presented in this section show that these issues, when combined with radio-based measurements, can be overcome resulting in a more robust solution, suitable for long-term operation of autonomous vehicles.

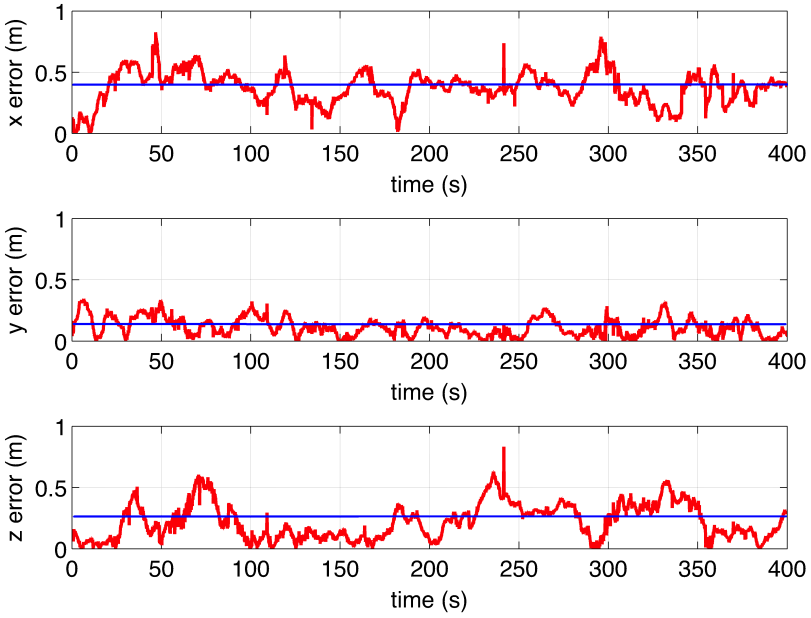

Figure 4.4: Localization errors in position and associated RMS error.

In general, experimental results show how the introduction of low-cost UWB sensing into the environment allows resetting odometry errors, hence estimating a reliable localization. Having UWB sensors in the environment guarantees a localization error in the order of the sensing deviation, while at the same time visual odometry allows properly filtering UWB noise and outliers.

Nevertheless, total position errors of around 0.5m may not constitute an appropriate solution for closing the control loop using such localization estimations. Besides, this approach is not valid for robust long-term full 6 DoF localization, since the yaw angle estimation only relies on the visual odometry algorithm; hence it will drift with time and eventually get the UAV lost by mixing yaw errors with position errors. Moreover, sensor fusion based on EKF is confined to cases where the Gaussian linear assumption is a suitable approximation. The following section focuses on a different method to address the more general case of unconstrained probability distributions, and constitutes a full solution for robust long-term 6 DoF state estimation.

4.3 Particle Filters for State Estimation

Instead of representing the distribution by a parametric form (the exponential function that defines the density of a normal distribution), particle filters represent a distribution by a set of samples that are randomly drawn. Such a representation is approximate, but is non-parametric, and therefore can represent a much broader space of distributions than, for example, Gaussians. The quality of this approximation depends on the number of particles used. Particle filter is also known as Monte Carlo Localization (MCL) estimation.

Particle filters do not make strong parametric assumptions on the probability density. In particular, they are well-suited to represent complex multi-modal beliefs. For this reason, they are often the method of choice when a robot has to cope with phases of global uncertainty, and when it faces hard data association problems that yield separate, distinct hypotheses. However, the representational power of these techniques comes at the price of added computational complexity. Fortunately, it is possible to adapt the number of particles to the (suspected) complexity of the distribution. When it is of low complexity (e.g. focused on a single state with a small margin of uncertainty), a small number of particles can be used; for complex distributions (e.g. with many modes scattered across the state space), the number of particles could grow large.

The particles are denoted as follows:

$$\mathbf{X}_t = \mathbf{x}_t^{[1]}, \mathbf{x}_t^{[2]}, ..., \mathbf{x}_t^{[N]} \tag{4.10}$$

Each particle $\mathbf{x}_t^{[i]}$, with $1 \leq i \leq N$, is a specific instantiation of the state at time t, that is, a hypothesis as to what the true state may be at time t. N denotes the number of particles in the particle set $\mathbf{X}_t$, which is usually a large number that may or may not be modified in order to adapt the complexity of the problem to the computational load, as explained before.

Since the set of particles approximates the belief over the current state of the robot, the denser a sub-region of the state space is populated by samples, the more likely it is that the true state falls into this region.

Just like other Bayes filter algorithms, the particle filter algorithm constructs the particle set $\mathbf{X}_t$ recursively from the set $\mathbf{X}_{t-1}$ using the most recent control u_t and the most recent measurement z_t (Markov assumption). Three basic steps are carried out in each iteration of the algorithm:

- A new hypothetical state $\mathbf{x}_t^{[i]}$ is generated based on the particle $\mathbf{x}_{t-1}^{[i]}$ and the control u_t.

- For each particle $\mathbf{x}_t^{[i]}$, an *importance factor* $w_t^{[i]}$ is calculated as the probability of the measurement z_t under the particle $\mathbf{x}_t^{[i]}$. If we interpret $w_t^{[i]}$ as the *weight* of the i-th particle, the set of weighted particles represents (in approximation) the robot state.

- Finally, a re-sampling stage takes place: the algorithm draws with replacement N particles. The probability of drawing each particle is given by its importance weight. Re-sampling transforms a particle set into another particle set of the same size, but with different distribution of particles, refocusing the particle set to regions in state space with high probability of being the true state.

Our work extends the particle filter proposed in Hornung et al. (2010). Unlike the previously introduced EKF which only takes as measurements the UWB-based distances, our approach also uses a 3D map of the environment in order to estimate how well the point cloud generated by the on-board 3D imaging sensor fits the current robot pose. Hence, this approach accounts for yaw estimation through map matching. How to build such map is explained in Chapter 5.

The motion model for the *prediction* step used in this work is based on a fast and reliable visual odometry. The odometry estimation is applied to the particle set as an estimation of the *a priori* distribution of the state belief. Then the measurements are integrated: point clouds provided by the 3D imaging sensor and distances to several UWB beacons installed in the environment. The weight of each particle is then calculated according to the two types of sensor readings. The point clouds are transformed to each particle's pose in order to find correspondences between the cloud and what the map should look like from such particle's pose. The UWB distance

measurements are used to check how well each particle position matches such distances. In order to fuse both types of sensor readings, each particle has a weight for each type of sensor, and they are later fused into a single weight.

The particle filter consists of N particles, each of them with the following state vector:

$$\mathbf{x}^{[i]} = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix} \tag{4.11}$$

where ψ refers to the yaw angle of the aerial robot. Even though a 4D state vector is used, we are able to provide 6D localization estimations since roll and pitch angles are observable through the UAV on-board IMU. This greatly reduces the computational complexity of the algorithm, allowing for real-time computation. The weights $w^{[i]}$ associated to the particle set satisfy:

$$\sum_{i=1}^{N} w^{[i]} = 1 \tag{4.12}$$

4.3.1 Initialization

Particles can be initialized automatically or manually by setting the initial position together with a covariance matrix to distribute the particles in the space.

An example of automatic initialization is shown in Figure 4.5. Particles are drawn uniformly over the 4D state (x, y, z and ψ) into the whole map. As soon as the UAV starts moving, the filter updates and good hypotheses start gaining weight while low-weight particles are shifted towards more interesting areas in the re-sampling stage. The figure also reveals how the robot does not need to move much to converge to a solution. This mainly occurs because the radio ranging measurements quickly induce the right location. However, some of these particles do not represent the correct yaw angle. As soon as the UAV starts moving in the horizontal plane towards some obstacles, the point cloud matching with the map refines the position and properly approximates the yaw angle.

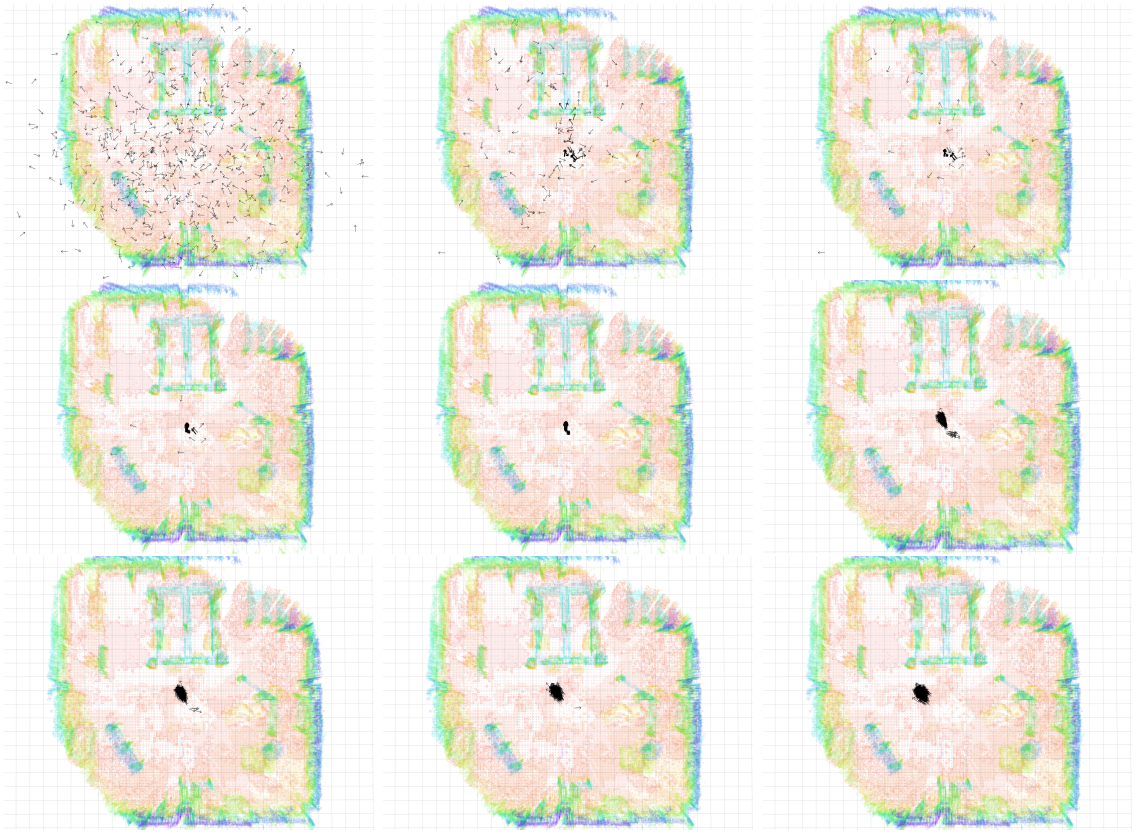

Figure 4.5: Automatic initialization of particles. From left to right and top to bottom, time evolution of particles after automatic initialization. Black arrows represent the particles.

Even though particle filters are able to deal with the "kidnapped robot" problem, we assume that the initial state of the robot is known and thus make use of manual initialization. This is usually the case of indoor UAV applications, since the aerial robots often take off from a designated location. Moreover, the robot navigation requires a smooth and stable motion estimation in order to perform a safe flight, which would not be the case if we start from an unknown position and let the filter converge around the true pose of the robot, most likely producing sudden and relatively high changes in the robot current state. The starting point of the method is an initial belief of the pose probability distribution, which determines the distribution of the particles around such initial state. In the case of manual initialization, a 4D normal

distribution is sampled using the provided initial position and its covariance matrix. The associated weights w_i are initialized to $1/N$, uniformly, being N the number of particles used.

4.3.2 Prediction

The state transition model is used to propagate the current state of all the particles, according to the visual odometry estimation computed. The prediction step involves applying a motion model to the current localization estimate represented by the current value of particles.

$$\Delta\mathbf{x} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta\psi \end{bmatrix} \tag{4.13}$$

This information is used to compute the *a priori* distribution of the particles. Thus, the state of the particles will evolve according to the following expressions:

$$x_t^{[i]} = x_{t-1}^{[i]} + \Delta x * cos(\psi_{t-1}^{[i]}) - \Delta y * sin(\psi_{t-1}^{[i]}) \tag{4.14}$$

$$y_t^{[i]} = y_{t-1}^{[i]} + \Delta x * sin(\psi_{t-1}^{[i]}) + \Delta y * cos(\psi_{t-1}^{[i]}) \tag{4.15}$$

$$z_t^{[i]} = z_{t-1}^{[i]} + \Delta z \tag{4.16}$$

$$\psi_t^{[i]} = \psi_{t-1}^{[i]} + \Delta\psi \tag{4.17}$$

The values of Δx, Δy, Δz and $\Delta\psi$ are drawn randomly following a normal distribution centered on the values provided by the visual odometry, and with standard deviations proportional to each increment itself, e.g. $\sigma_x = k_x * \Delta x$ with $k_x > 0$. The value of k_x is always positive greater than zero and it is a design parameter that depends on the accuracy of the odometry system. This noise term is necessary to avoid premature convergence and to maintain the diversity of localization hypotheses. Besides, this term can also account for the imprecision of the motion model.

4.3.3 Update

In this phase, we use a measurement model to incorporate information from the sensors. We have defined thresholds in position and orientation such that if the visual odometry estimation exceeds any of those, a filter update is performed using the last 3D point cloud received from the stereo/RGB-D camera, and all the distance measurements to the UWB beacons received since the last filter update. Then, each particle $\mathbf{x}_t^{[i]}$ evaluates its relative importance by checking how likely it would receive such sensor readings at its current pose, hence computing a new weight value $w_t^{[i]}$. Then, particles can later be re-sampled considering these weights, thus obtaining a new estimate of the current state given the last measurement z_t.

In this work, z_t corresponds to the point clouds from the 3D imaging sensor and the distance measurements to the UWB sensors. The update is performed through the use of a 3D map of the environment in the form of an OctoMap Hornung et al. (2013), augmented with the position of the fixed locations of UWB beacons. Given the distinct nature of the two technologies involved, we calculate separate weights for each sensing modality, i.e. $w_t^{[i],map}$ and $w_t^{[i],uwb}$. A simple weighted average is later used to obtain the final weight of each particle:

$$w_t^{[i]} = \alpha * w_t^{[i],map} + (1 - \alpha) * w_t^{[i],uwb} \tag{4.18}$$

where α is chosen depending on the particularities of the indoor environment where the UAV is going to operate. If the map used in the filter does not contain the full environment, or its accuracy is not enough to trust the map matching, α should be lower than 0.5. Whereas if there are few UWB sensors deployed in the scenario, or their location is not accurate, α should be higher.

Computation of $w_t^{[i],map}$

The acquired 3D point cloud is transformed to each particle's pose in order to find correspondences between such cloud and what the map should look like from that particle's pose. Since this is very expensive computationally, we first compute a 3D probability grid as in Hornung et al. (2010), in which each position stores a value of

how likely it is that such position falls within an occupied point of the map, instead of storing binary information about occupancy as in the provided map. Each 3D position p_i of the grid is then filled with probability values according to a specific Gaussian distribution centered in the closest occupied point in the map from p_i, map_i, and whose variance σ^2 depends on the sensor noise used in the approach.

$$grid(p_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-||p_i - map_i||^2 / 2\sigma^2} \tag{4.19}$$

Such probability grid only needs to be computed once, is not required to be updated for a given environment, and relieves from performing numerous distance computations between each cloud point for each particle and its closest occupied point in the map. Besides, each point cloud is first transformed according to the current roll and pitch provided by the on-board IMU. This transformation is done just once per update, reducing the computational requirements as well.

Then, for every point of the transformed cloud, we access its corresponding value in the 3D probability grid. Such value would be an indicator of how likely is that point to be part of the map. By doing this with every point of the cloud and adding all the probability values, we obtain a figure of how well that particle fits the true location of the aerial robot according to the map.

Finally, the weight w_i of each particle $\mathbf{p}_i$ is computed. Assuming that the point cloud is composed by M 3D points $\mathbf{c}_j$, the weight is computed by adding all the associated probability grid values as follows:

$$w_i^{map} = \frac{1}{M} \sum_{j=1}^{M} grid(\mathbf{p}_i(\mathbf{c}_j)) \tag{4.20}$$

where $\mathbf{p}_i(\mathbf{c}_j)$ stands for the transformation of the point to the particle's state, and $grid(\mathbf{p}_i(\mathbf{c}_j))$ is the evaluation of the probability grid in such transformed position.

Computation of $w_t^{[i],uwb}$

On the other hand, distance measurements between UWB sensors are used to compute another weight value for each particle according to how well their state fits to the

distribution of fixed radio beacons. Since the radio beacons do not provide bearing information, we first define new states without ψ (yaw angle), i.e. $\mathbf{x'}_t^{[i]} = [x, y, z]^T$, and UWB beacon states using their estimated positions, $\mathbf{b}_j = [x_j, y_j, z_j]^T$. Given a measured distance d_j from the UWB sensor on-board the UAV to the j-th beacon, the following constraint can be applied to each particle state:

$$d_j = ||\mathbf{x'}_t^{[i]} - \mathbf{b}_j|| \tag{4.21}$$

This can be easily applied to the particle weight calculation according to the actual Euclidean distance r_{ij} between the state $\mathbf{x'}_t^{[i]}$ of the i-th particle and $\mathbf{b}_j$, and how close this is to d_j. The product is used to aggregate the values from the measurements of different beacons, since they are independent probabilistic processes. The weight of the i-th particle associated to UWB sensing is calculated as follows:

$$w_t^{[i],uwb} = \prod_{j=1}^{B} \frac{1}{\sigma\sqrt{2\pi}} e^{-(d_j - r_{ij})^2 / 2\sigma^2} \tag{4.22}$$

where B is the number of beacons in the scenario. The distance measurements of the sensors used in our approach have a standard deviation σ of roughly 0.1m, after filtering potential outliers.

A great advantage of this approach is that the distance to a single sensor is enough to update the weights of the particles, it does not need to wait until distances to three or more sensors are obtained by the on-board radio-tag, which usually happens in other state-of-the-art range-based localization systems.

Weight combination

Before combining the weights of both sensing approaches, all the weights must first be normalized within their categories in order to verify Equation 4.12, hence representing a valid probability distribution. This also prevents combining weights of distinct nature which usually are in different orders of magnitude. Equation 4.18 is used to obtain the final weight of each particle, and the new weights are normalized again in order to represent a valid probability distribution.

We have also evaluated the option of drawing new particles into the filter with each new radio-based measurement, taking into account the lack of bearing information. Thus, with each distance measurement, a set of particles would be drawn in a sphere surrounding the on-board sensor position into the map. Most of these particles will be later on destroyed in the next re-sampling after update thanks to the weight contribution of the point cloud matching with the 3D map. However, this approach was discarded because it requires a large number of particles for a proper representation of the range hypotheses, which hinders the computational efficiency.

4.3.4 Re-sampling

The next step of the method in order to complete one iteration of the algorithm involves the generation of a new set of particles from the current one, by choosing each particle $\mathbf{x}_t^{[i]}$ according to its weight $w_t^{[i]}$. Hence, the new poses of the particles will be more likely to be accurate. The algorithm employed for re-sampling is the low variance sampler Thrun et al. (2005). In this case, a single random number is used as a starting point to obtain N particles according to their relative weights.

4.3.5 Pose Computation

The particle filter method is an iterative process that involves moving, sensing and re-sampling multiple localization hypotheses. While the filter produces a set of localization hypotheses, the navigation needs to decide which hypothesis is the correct one. The updated state vector for the aerial robot is then calculated as the weighted sum of all the particles, since when the filter converges to a single hypothesis, that is a good estimation of the true position of the robot. In this case, all particles are found to be clustered around a specific region.

Due to the mixed sensing approach, the outliers commonly present in indoor UWB measurements are rejected thanks to the particle weighting. If an outlier in the distance measurement from a radio-based sensor is received, following Equation 4.22, this would result in very low values for $w_t^{[i],uwb}$; in this case the re-sampling would only depend on the associated weights calculated from the point cloud matching, and

the relative scoring among the set of particles would remain unaffected. On the other hand, if the map matching between using the 3D point clouds is not accurate, its associated weight would not influence the overall particle's weight, which will only depend on the contribution from the UWB side.

4.3.6 Experimental Results

An experimental setup has been conceived to validate the presented approach. The field test took place at an indoor testbed of CATEC; part of it can be seen in Figure 4.6.

The aerial robot in Figure 4.7 performed a flight based on the testbed's motion capture system for localization, and navigated using joystick commands that sent nearby position and orientation waypoints in the robot coordinate frame. The results presented in this section were obtained off-line. In order to validate the long-term character of our approach, the flight took roughly 9 minutes so the visual odometry could drift enough to verify that the two sensing measurements used in the particle filter help preventing localization error growth.

The complete experimental setup is composed of the following elements:

- An aerial robot with the following on-board sensors: an RGB-D sensor (Orbbec's Astra), an IMU and a UWB sensor (see Figure 4.7).
- A 3D map of the working area (see Figure 4.6) obtained using an RGB-D sensor (ASUS's Xtion PRO LIVE) and motion capture data.
- Three UWB beacons installed across the indoor testbed at known positions.

The 3D map of the area was acquired using a different RGB-D sensor from the one on-board the UAV, whose housing was augmented with several passive markers (see Figure 4.8) in order to get its pose with high accuracy from the testbed's motion capture system. Such sensor was manually carried around the testbed while connected to a laptop, and the acquired point clouds were projected using the corresponding sensor pose and merged into an accurate OcTree. This map can be seen in Figure 4.6.

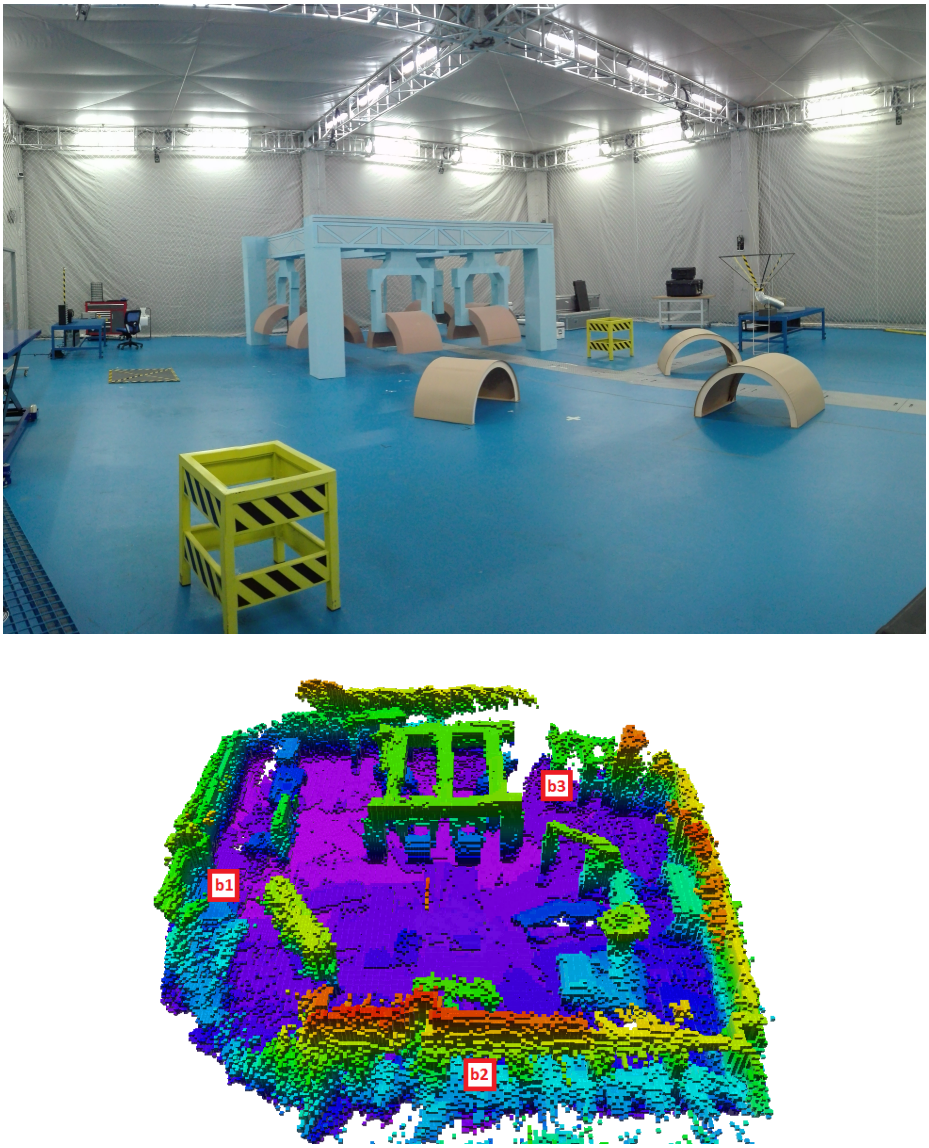

Figure 4.6: CATEC indoor testbed (top) and 3D map with approximate radio beacons locations(bottom) used for field experiments.

A small infrastructure of UWB beacons (three sensors) has been installed in the indoor scenario where the flight has taken place. The radio-based sensors have been installed at three different locations homogeneously distributed within the indoor testbed. Their true locations were acquired using the motion capture system and

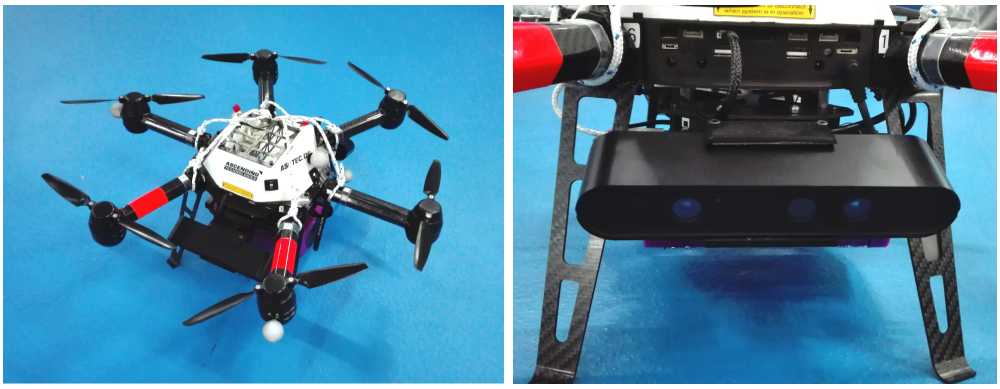

Figure 4.7: UAV with an RGB-D sensor at the front.

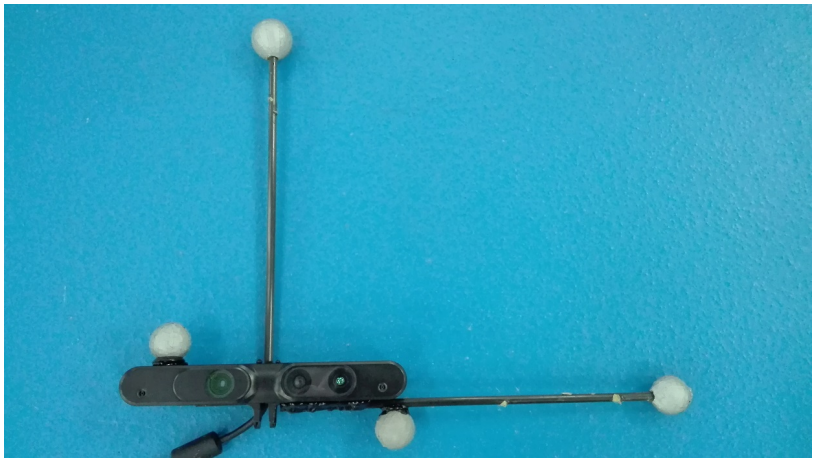

Figure 4.8: Another RGB-D sensor with passive markers for precise 3D map building.

provided to our filter. There is an additional UWB sensor installed on the aerial robot in order to compute point-to-point distances.

The ground-truth trajectory followed by the vehicle is presented in Figure 4.9. This is a roughly 200 meters long trajectory in which the robot performed a trajectory for a whole battery duration.

The main objective of the experiment is to demonstrate the suitability of the approach in real-time during regular operations. The estimated UAV position and yaw during the experiment can be seen in Figure 4.10, both from the raw visual odometry and the particle filter. Roll and pitch angles are not included in the plots because, as previously stated, they are observable from the on-board IMU and are directly

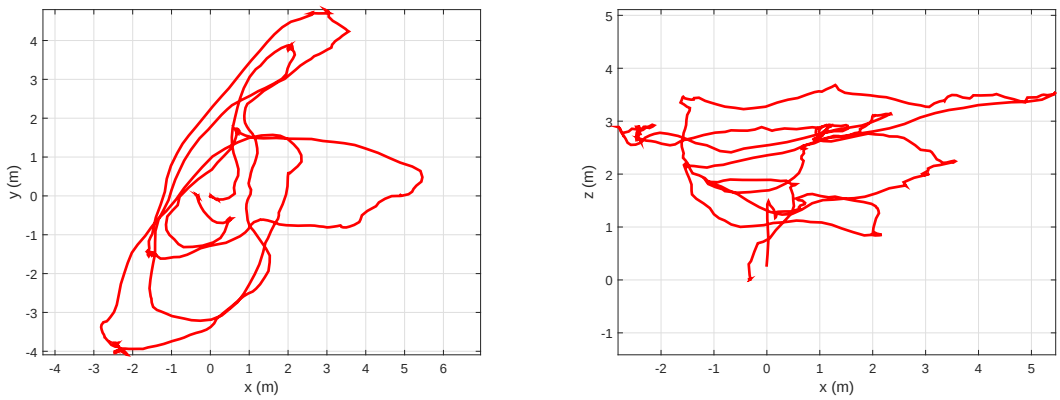

Figure 4.9: Ground-truth trajectory followed by the aerial robot during the experiment.

integrated into our localization approach. The IMU used in our approach provides accurate and filtered estimations of such angles.

These results (Figures 4.10 and 4.11) were obtained after configuring the particle filter with the parameters shown in Table 4.1.

Table 4.1: MCL parameters

Parameter	**Value**
Number of particles	500
α (Eq. 4.18)	0.5
OcTree resolution	0.1m
RGB-D sensor σ	0.05m
UWB sensor σ	0.1m
Update threshold (pos)	0.1m
Update threshold (rot)	0.1rad

The plot also shows the ground-truth data provided by the testbed's motion capture system. It can be seen how the visual odometry slowly accumulates errors over time, while the estimations from our complete approach closely follow the ground-truth during all the experiment. It is also worth to mention that this estimation does not drift with time and the errors are approximately bounded.

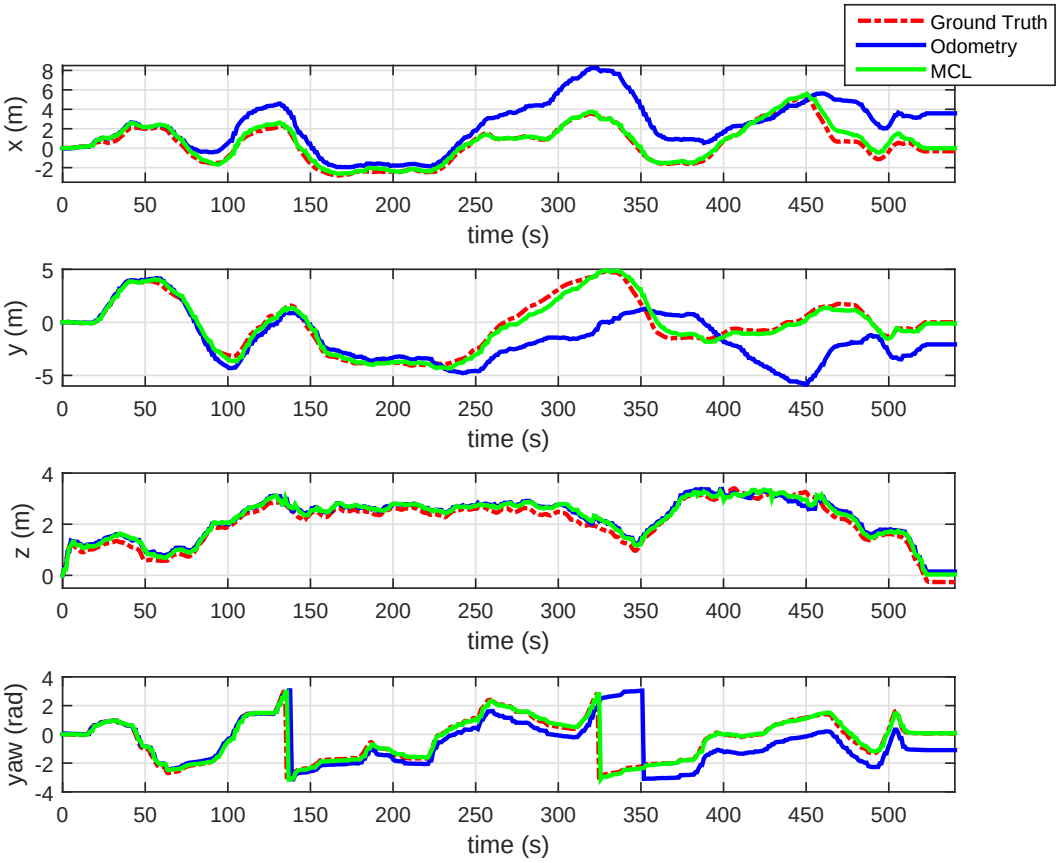

Figure 4.10: Localization results (position and yaw) showing the ground-truth, visual odometry and the proposed approach.

Figure 4.11 shows the error of the estimation during the trajectory execution. The error is computed as the Euclidean distance between the estimation and the ground-truth at every time step. In the figure, left plots correspond to the odometry approach, while right plots show results from the complete proposed approach including the filter. Besides, the plot depicts the computed RMS errors for each axis, which are also shown in Table 4.2.

It can be seen how the RMS errors are approximately 0.4m in x and y, while in z is less than 0.2m thanks to the ground estimation module. RMS error in yaw angle is around 10 degrees. Errors in x and y are higher due to the larger distance traveled

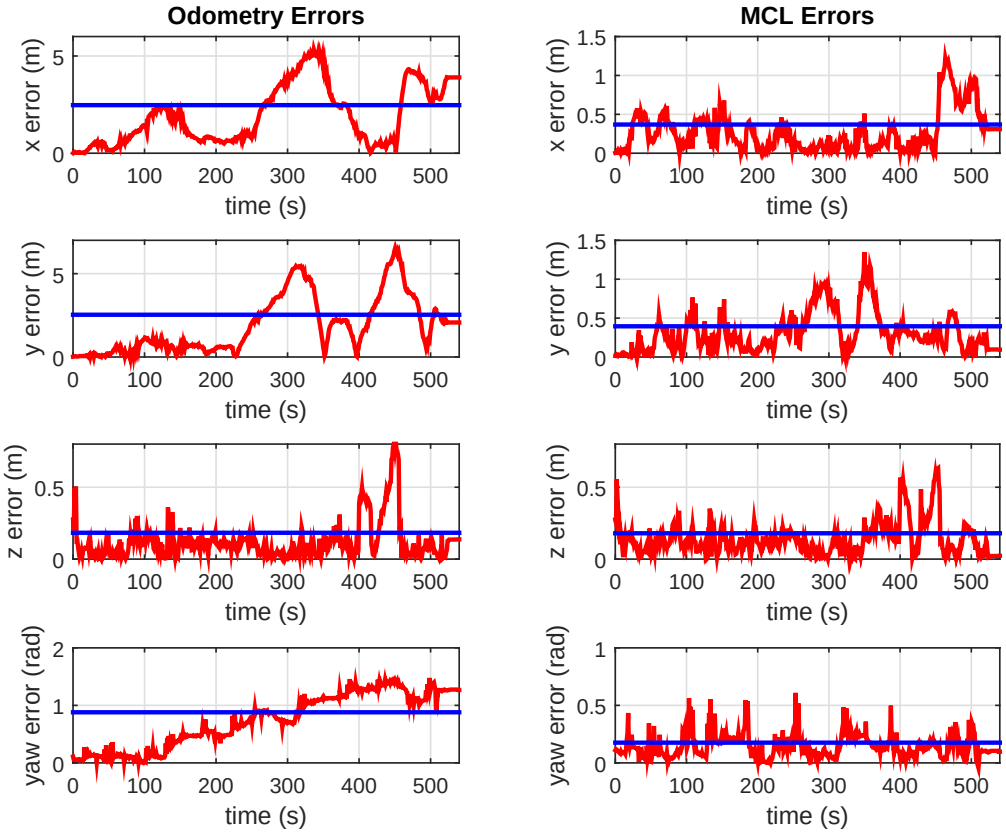

Figure 4.11: Position and orientation errors with respect to ground-truth through time.

through these axes. Nevertheless, the global RMS error in position is 0.32m which can be considered acceptable for robot navigation.

Table 4.2: RMS localization errors

	x (m)	y (m)	z (m)	yaw (rad)
Odometry	2.47	2.53	0.19	0.88
MCL	0.36	0.39	0.17	0.18

In order to quantitatively assess the performance of our localization approach, we have tested the data from the validation experiment against other popular approaches

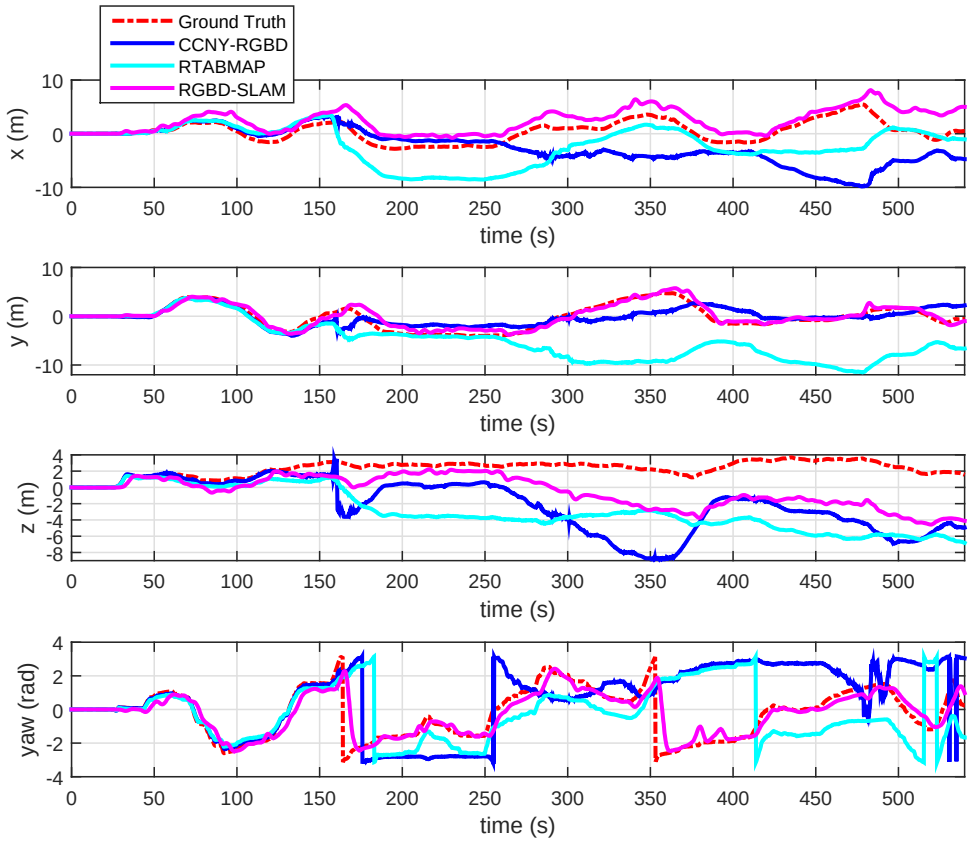

Figure 4.12: Estimated UAV position and orientation using other approaches based on RGB-D sensors.

that make use of RGB-D data for robot localization, either visual odometry and mapping such as CCNY_RGBD[1], or full SLAM approaches such as RTAB-Map[2] or RGBD-SLAM[3]. The plots in Figure 4.12 show the performance of the same flight data processed off-line using these approaches, and Table 4.3 summarizes general RMS errors compared to those from our approach.

The RMS errors in x, y and z from the compared approaches are slightly higher. Omitting z values, which could be observable through the use of an altimeter, only RGBD-SLAM seems to be a viable alternative to the proposed approach. We are able

[1]http://wiki.ros.org/ccny_rgbd
[2]http://wiki.ros.org/rtabmap
[3]http://wiki.ros.org/rgbdslam

Table 4.3: RMS error comparison

	x (m)	y (m)	z (m)	yaw (rad)
CCNY RGBD	5.07	1.94	5.55	2.24
RTAB-Map	3.68	6.59	6.09	2.06
RGBD-SLAM	2.46	0.79	3.63	0.71
Our approach	**0.36**	**0.39**	**0.17**	**0.18**

to provide a better accuracy while keeping robustness and computational efficiency as key features. Notice that we did not fine tune the parameters of the algorithms, so a better accuracy could be achieved.

Table 4.4 includes values for the mean frequency and corresponding mean processing times at which new pose estimations are published, using an i7 Linux laptop with 2 Cores (the UAV on-board processor is also an i7 Linux computer). Our approach, including the visual odometry, is able to deliver pose estimations approximately every 60ms when using 500 particles, which makes it suitable for working at around half the frame rate of the RGB-D sensor. The other approaches may publish pose estimations to the ROS ecosystem at a higher rate than that of the Table, but internally they just copy the same values to the output message until a new estimation is computed.

The visual odometer took an average 90% of a single thread and the proposed particle filter algorithm the 55% of a single thread. This configuration leaves plenty of computation for robot planning and navigation.

Table 4.4: Pose estimation speed

	Average Freq (Hz)	Frame processing time (ms)
CCNY-RGBD	8.1	124
RTAB-Map	6.2	161
RGBD-SLAM	5.8	173
Our approach	**15.8**	**63**

It can be seen how the localization accuracy can be improved in long-term operation by using both RGB-D and radio-range sensing. In order to evaluate how each sensing

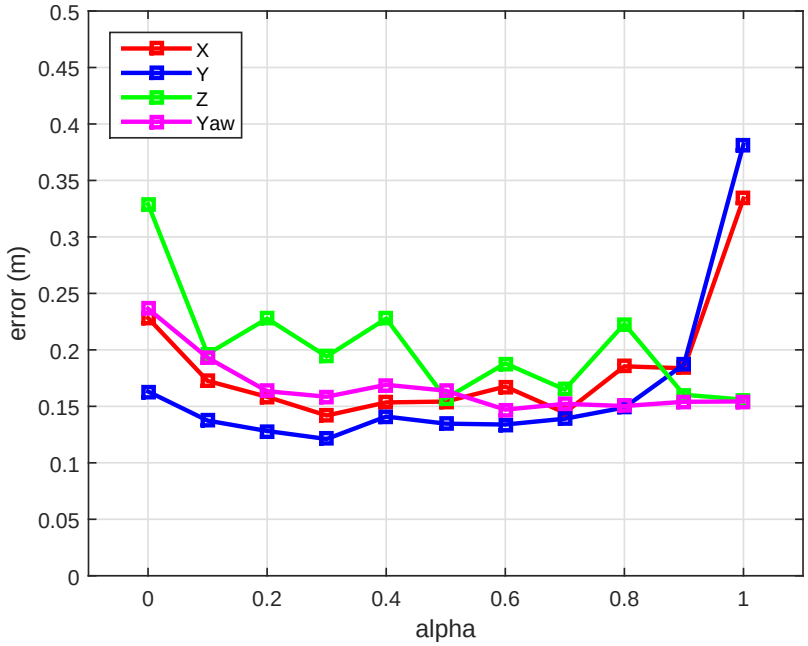

Figure 4.13: Localization errors for different values of α.

modality contributes to such improvement, the parameter α from Equation 4.18 is modified in order to assess localization errors. The rest of parameters from Table 4.1 remained the same for this experiment. Figure 4.13 shows errors in x, y, z and yaw for different values of α, from $\alpha = 0$ (we only rely on radio-based sensing) to $\alpha = 1$ (we only use map matching). While, in general, RMS errors are low, best performances in terms of overall stability and particle cloud convergence were achieved between $\alpha = 0.5$ and $\alpha = 0.7$. As expected, yaw error increases when the contribution from map matching decreases (lower α), since radio sensing does not provide such angle and the approach relies on the odometry integration. Errors in z are also expected to be higher when α is closer to 0, since the spatial distribution of the installed radio beacons in height was limited, leading to poor estimations in the UAV altitude when not using the point cloud matching with the 3D map. On the other hand, when α is close to 1, point cloud matching leads to smoother estimations, but sometimes is not enough to correct odometry drift, leading to higher errors in x and y.

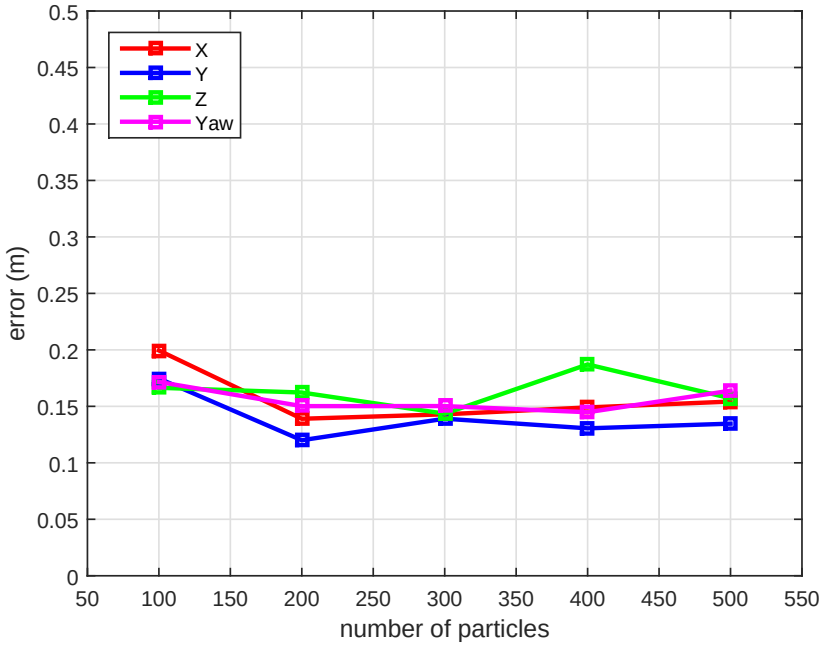

Figure 4.14: Localization errors for different number of particles used.

Figure 4.14 presents the estimation accuracy by using different numbers of particles, along with the required computational times included in Table 4.5 in order to demonstrate the capabilities of our approach. The rest of parameters are still the default ones, as shown in Table 4.1. It can be seen how the impact of the number of particles on RMS errors is fairly minor, while computation times are drastically reduced.

Table 4.5: Computation times for one iteration when modifying the number of particles

Number of particles	Avg. processing time (ms)
100	26
200	42
300	50
400	58
500	65

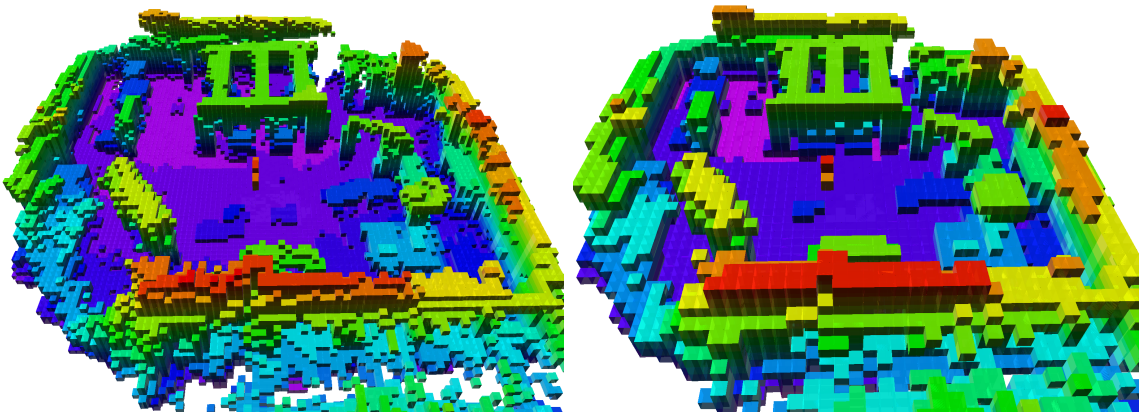

Figure 4.15: 3D map of the area with different resolutions: 0.2m (left) and 0.4m (right).

The spatial occupancy of the working area is approximated by an OcTree, as shown in Figure 4.6. Another set of experiments have been performed by modifying the spatial resolution of such map from our default value (0.1m) to the following two depths of the tree, which correspond to leaf sizes of 0.2m and 0.4m. Figure 4.15 shows these maps. Running the experiment with higher sizes was discarded because the map is no longer representative enough for map matching. The number of particles was 500, and $\alpha = 0.5$. Figure 4.16 shows the localization RMS errors using different map resolutions. It can be noted how when using a map resolution of 0.2m, and downsampling the sensor point clouds accordingly, it is possible to achieve similar localization errors while decreasing the computational complexity of the approach, as shown in Table 4.6.

Table 4.6: Computation times for one iteration when modifying the 3D map resolution

Map resolution (m)	Avg. processing time (ms)
0.1	65
0.2	52
0.4	28

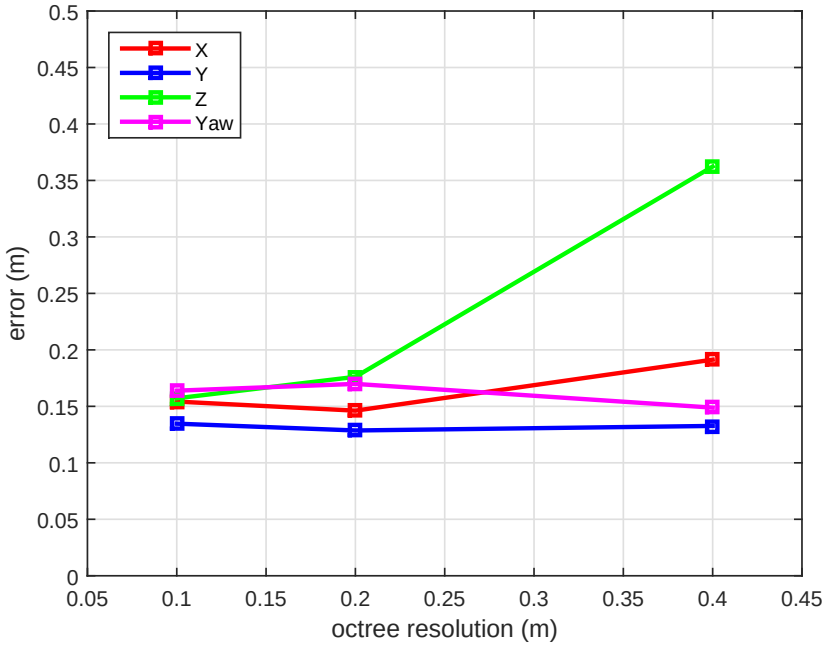

Figure 4.16: Localization errors for different OcTree resolutions of the 3D map.

4.4 Conclusions

This chapter focuses on the fusion of measurements from different sensors in order to improve our localization solution. The main contribution of the proposed approach is how the localization accuracy can be improved in long-term operation by using both visual-inertial and radio-range sensing.

First, an EKF-based strategy was adopted to combine the visual-inertial odometry from Chapter 3 with UWB-based distance measurements to fixed radio beacons. However, this solution lacked observability of the yaw angle for long-term operation, and a novel approach was proposed based on a particle filter using a 3D map of the environment. The level of accuracy of our solution improved the performance of available SLAM-based approaches, while exhibiting greater robustness under flight motions and larger computational efficiency when compared to such methods.

Experimental results demonstrate the feasibility of the approach, both in accuracy and computational efficiency, using an aerial robot during a long flight. The estimations

from our complete approach closely follow the ground-truth during all the experiment. It is also worth to mention that this estimation does not drift with time and the errors are approximately bounded.

Even though good accuracy through the whole flight duration were achieved, the localization approach relied on a 3D map that was built using a motion capture system. In order to be able to extend the proposed system to a custom indoor scenario for autonomous UAV operation, we must complement this methodology with mapping capabilities relying also only on 3D imaging sensors and UWB sensing. This is explained in detail in Chapter 5.

Chapter 5

Multi-Modal Mapping

Prior to robot localization, a 3D map of the environment needs to be built in order to make use of the particle filter proposed in Chapter 4. Available sensor data to achieve this are the 3D point clouds from stereo or RGB-D cameras and the distance measurements between UWB sensors (one of them is on-board the UAV and the rest are fixed in the environment). Neither the UAV trajectory nor the position of the UWB sensors in the environment are known *a priori*.

The objectives of this approach are to map the environment based on the 3D point clouds and to accurately localize the set of fixed UWB sensors, or beacons. Approaching these two tasks separately may seem logical. Firstly, the scene could be mapped based exclusively on the point clouds while at the same time the aerial robot trajectory is calculated. This is usually performed through the use of a SLAM approach based on 3D point clouds such as Endres et al. (2012); Labbe and Michaud (2014). Later, the trajectory information could be used to accurately locate the UWB beacons into the map, according to the corresponding distance measurements from each UAV pose to each beacon. However, this method does not take advantage of UWB sensing for mapping, neither localization.

The method proposed in this chapter follows an integrated approach in which we take advantage of both sensing modalities. The position of the UWB beacons are firstly approximated, and later on refined together with the 3D map of the environment. There are two main steps in our approach. The first step considers data from the

UWB sensors to compute a globally consistent 3D trajectory of the UAV, and then to automatically perform loop closing, which is the detection of a previously visited location by some means. Both the initially computed trajectory and the loop closures will be used in a second step to optimize the 3D position of the UAV, and also the positions of the UWB beacons. With this information, a 3D map can be built according to the estimated UAV trajectory and projecting the 3D point cloud at each location. Further details about the implementation of both steps are explained in the following sections.

5.1 Range-only localization and mapping (step 1)

The objective of the first step of the algorithm is to compute an initial guess of the positions of the UWB beacons based on the measured distances to the sensor on-board the UAV. Then we can use this information to obtain a globally consistent trajectory of the aerial robot. This is generally referred to as a Range-Only Simultaneous Localization and Mapping (RO-SLAM) Kantor and Singh (2002); Newman and Leonard (2003). In general, RO-SLAM aims to localize a mobile system and at the same time map the position of a set of range sensors. In contrast with other SLAM approaches based on cameras or LIDAR, RO-SLAM has the advantage of not requiring direct line of sight between each pair of sensors when radio-based range sensing is used. Besides, the problem of data association is trivially solved; this could pose a significant obstacle for the algorithm, since range sensors typically provide distance measurements to some object without identifying such object. In the case of radio-based sensors, they usually transmit their unique ID along with the range information, as it is the case of the UWB sensors used in the proposed approach.

Most of the approaches for RO-SLAM in the literature are based on time filtering and probabilistic frameworks such as EKF-SLAM, Unscented Kalman Filter (UKF)-SLAM, FastSLAM, and others. Comparisons of these frameworks can be found in Kurt-Yavuz and Yavuz (2012) and Li et al. (2012). Kurt-Yavuz and Yavuz (2012) present how the unscented FastSLAM exhibits better performance over other classical methods based on EKF or UKF. However, FastSLAM solutions do not preserve the

correlation between different beacons of the map in those applications in which it might exist. Thus, for example, in Blanco et al. (2008); Wang et al. (2009); Yang (2012), a FastSLAM solution is proposed using a particle filter for robot localization and another one for each beacon (i.e. for each radio emitter). In Yang (2012), an optimization on the beacons' particle filter is proposed by reducing the number of particles, hence decreasing the computational burden of the filter. On the other hand, Wang et al. (2009) optimize the problem using an adaptive re-sampling method which dynamically reduces the number of particles required for each beacon. A different beacon-based SLAM algorithm is considered in Hai et al. (2010), where the authors use a particle filter to initialize the EKF filter for each new beacon of the FastSLAM. The main drawback of the previous approaches lies in the delayed initialization of the beacons into the filters, which significantly reduces the optimization of the robot localization until the position solution of the range sensors has converged.

This problem is solved in Boots and Gordon (2013), where a batch solution is proposed to estimate the position of the mobile robot and the beacons by using a singular value decomposition (SVD) of the observation matrix. A batch processing solution is also presented in Kehagias et al. (2006) based on optimization. However, these methods assume measurements from all the radio beacons at every robot position; otherwise, the measurement must be interpolated. This is a hard constraint in realistic implementations where the visibility of all the range sensors cannot be guaranteed at all times.

This section proposes a new optimization approach for the RO-SLAM problem that follows the batch processing ideas but generalizes to a more common situation in which the range measurements from the range sensors might arrive at the robot independently, or even a specific robot pose does not have associated range measurements.

5.1.1 Problem Definition

The UAV trajectory is represented as a series of N robot poses $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, where each pose is defined as $\mathbf{x}_i = [x_i, y_i, z_i, \psi_i]^T$, being ψ the yaw angle of the aerial robot. There is also a set of M UWB beacons $B = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_M\}$, each one located at

a position defined by $\mathbf{b}_j = [x_{b_j}, y_{b_j}, z_{b_j}]^T$. The objective of the RO-SLAM optimization problem is to compute the best UAV trajectory X and locations of UWB beacons B that minimizes the errors with respect to the observations of the UWB measurements from each UAV pose.

Note how the robot roll and pitch angles are not included into the robot pose definition. We assume these angles are available and accurate enough in an UAV through the use of its on-board IMU. They are fully observable and their values are usually accurate in aerial robots because they are the most basic control variables (together with the rotation rates) for the system stability. While it is true that roll and pitch estimation based on accelerometer and gyroscope integrations might be biased under constant accelerations (e.g. loitering in fixed-wing UAVs), these scenarios are very difficult to achieve indoors and hence are not considered in this approach.

The UWB-based range observations are the distances between the UAV position and each beacon position. Thus, the observations at pose $\mathbf{x}_i$ are modeled as the set of measurements $D_i = \{d_{i1}, d_{i2}, ..., d_{iM}\}$ with arbitrary length from 0 (no measurements) to M (measurements to all UWB sensors), where d_{ij} is the Euclidean distance between pose $\mathbf{x}_i$ and beacon position $\mathbf{b}_j$:

$$d_{ij} = \sqrt{(x_i - x_{b_j})^2 + (y_i - y_{b_j})^2 + (z_i - z_{b_j})^2} \tag{5.1}$$

Thus, the resulting aerial robot trajectory and UWB beacon positions will be that which minimizes the following expression:

$$\underset{\{X,B\}}{\arg\min} \left[\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} \left(\|\mathbf{x}_i - \mathbf{b}_j\| - d_{ij} \right) \right] \tag{5.2}$$

where c_{ij} is a variable that takes value 1 if there is a measurement from pose $\mathbf{x}_i$ to UWB beacon $\mathbf{b}_j$, and 0 otherwise.

However, due to the nature of the problem addressed, there exists the possibility that a number of poses do not have associated measurements (the UAV is out of range of all UWB beacons) and, more frequently, that the number of range measurements in a pose is below four (minimum number to compute the robot position in 3D).

These limitations are overcome by including odometric constraints into Equation 5.2, obtaining the final expression to minimize:

$$\underset{\{X,B\}}{\arg\min} \left[\sum_{i=1}^{N} \left(E_i + \sum_{j=1}^{M} c_{ij}(\|\mathbf{x}_i - \mathbf{b}_j\| - d_{ij}) \right) \right] \tag{5.3}$$

where E_i stands for the error between pose $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ according to odometry information. This function transforms the pose $\mathbf{x}_{i-1}$ according to the estimated robot odometry and computes the error with respect to $\mathbf{x}_i$ in each pose dimension. We make use of the visual-inertial odometry algorithm presented in Chapter 3.

5.1.2 Optimization

Solving Equation 5.3 is straight-forward if good guesses about the aerial robot poses and the positions of UWB beacons are available. However, RO-SLAM does not necessarily have prior information about the position of the beacons. An option would be to initialize B to random positions and let the optimization process estimate them, but the algorithm will most likely get stuck in a local minimum.

In order to overcome this problem, this approach proposes two actions. First, the z_b parameter of every UWB beacon in B could be estimated by just measuring their distances to the floor. This is an easy process that can be implemented accurately. This assumption allows reducing the number of unknown parameters of each UWB sensor position to two (x_b and y_b), but still, more information is needed in order to initialize each UWB beacon position for the optimization method.

The second action consists in re-parameterizing the beacon position so that we can feed several estimation hypotheses into the optimizer, as proposed in Fabresse et al. (2013). Thus, when the robot is at pose $\mathbf{x}_i$ and receives the first range measurement d_{ij} to sensor j, the beacon $\mathbf{b}_j$ would be in a circumference around the current robot pose and at altitude z_{b_j}. The proposed approach samples this circumference with multiple position hypotheses at different angles, allowing the optimizer to choose the best one. Figure 5.1 shows an example including a UAV 3D trajectory seen from an orthogonal view for easy visualization (triangles are robot poses and circles are UWB

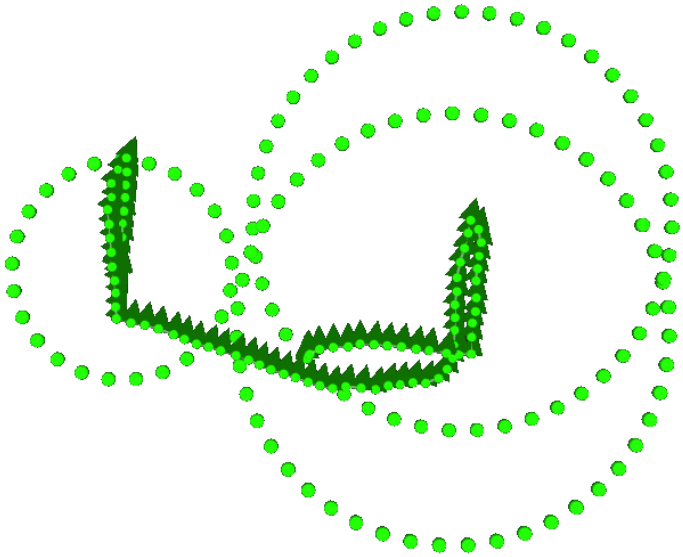

Figure 5.1: Multiple hypotheses for the localization of three UWB beacons.

beacon position hypotheses). Thus, assuming H different position hypotheses along the circumference, the UWB beacon position will be parameterized as follows:

$$\mathbf{b}_j = [\mathbf{b}_{j1}, \mathbf{b}_{j2}, ..., \mathbf{b}_{jH_j}] \tag{5.4}$$

where each position hypothesis is represented as $\mathbf{b}_{jk} = [x_{b_{jk}}, y_{b_{jk}}, z_{b_j}]^T$. Note how z_{b_j} is the same for all hypotheses because it is already known.

With this parameterization in mind, Equation 5.3 can be reformulated as follows:

$$\underset{\{X,B\}}{\arg\min} \left[\sum_{i=1}^{N} \left(E_i + \sum_{j=1}^{M} \frac{1}{H_j} \sum_{k=1}^{H_j} c_{ij} \left(\|\mathbf{x}_i - \mathbf{b}_{jk}\| - d_{ij} \right) \right) \right] \tag{5.5}$$

Note how the contribution of a single UWB sensor j is scaled by $1/H_j$ for every pose $\mathbf{x}_i$, so that we do not double-count the information provided by a single range measurement.

5.1.3 Initialization

In summary, the parameters of the optimization process are the position of the robot in each pose $\mathbf{x}_i$, which is initialized according to the odometry values, and the different position hypotheses for every UWB beacon.

When a range measurement d_{ij} is received from the j-th beacon for the first time, the current robot position estimation $\mathbf{x}_i$ is used to initialize the H_j position hypotheses according to the following equations:

$$x_{b_{jk}} = x_i + d_{ij} \cdot cos(2\pi(k-1)/H_j) \tag{5.6}$$

$$y_{b_{jk}} = y_i + d_{ij} \cdot sin(2\pi(k-1)/H_j) \tag{5.7}$$

$$z_{b_{jk}} = bz_j \tag{5.8}$$

The value of H_j is initialized to $H_j = 10 \cdot d_{ij}$, so that the number of hypotheses adapts to the sensor distance.

Thus, the outcome of this first step is a globally consistent trajectory and an initial position estimation for the UWB beacons. If the robot motion was enough to let the optimizer disambiguate the horizontal flip ambiguity Fabresse et al. (2016), then the algorithm converges to a single solution and all the hypotheses will be localized in a specific position.

5.1.4 Weighting

It is important to take into account that the different sources of information in Equation 5.5 have different levels of accuracy. It is very usual to make use of the constraint's associated information matrix to fine tune the minimization process. In this way, very different sources of information such as visual odometry and UWB sensors can be easily considered.

However, practical experience shows that the UWB measurements tend to incorporate outliers frequently due to multi-path effects or signal attenuation. This is the reason behind weighting the different components into Equation 5.5 differently. We establish a weighting factor assuming that the odometry is reliable in the short term,

while UWB might provide outliers depending on the environment and no matter the time. Thus, in the proposed implementation, UWB constraints are multiplied by a factor ranging from 0.5 to 0.1 depending on the quality of the information, while odometry will be always weighted by a factor of 1.0. That is, we trust odometry constraints more than UWB at short-term.

5.2 3D Mapping and Pose Refinement (step 2)

The results of the first step are a coherent UAV trajectory and initial position estimations of the UWB beacons. The second step in our methodology involves the automatic detection of loop closures, which is usually based on different approaches such as visual place recognition or scan matching. In this case, a massive scan matching process is carried out among all the poses that fall within a given search radius, based on the 3D point clouds acquired by the on-board 3D imaging sensor (a stereo or RGB-D camera).

This loop-closure detection will add new constraints to the robot poses Grisetti et al. (2010). However, this approach goes one step forward and takes into account a new type of constraint apart from the usual transform between poses, so that we can also optimize the alignment between the 3D point clouds directly into the optimizer. Given the sensor point cloud $\mathbf{pc}_i$ at pose $\mathbf{x}_i$, and the point cloud $\mathbf{pc}_j$ at pose $\mathbf{x}_j$, the scan matching process establishes the transform that best aligns both point clouds (using ICP as in Chapter 3). In the literature, this transform is commonly used as a constraint between both poses, and its associated information matrix allows tuning the importance of such constraint into the non-linear optimization process. Instead, we propose to include the alignment error into the optimization process.

To this end, each point cloud is transformed to the global reference frame according to its associated pose ($\mathbf{pc}_i^g$) and then the alignment error between point clouds is computed as the averaged Euclidean distance between their individual 3D points. Equation 5.3 is used instead of Equation 5.5, since now a single hypotheses is available for each UWB position. This equation can be enlarged with this new constraint as

follows:

$$\underset{\{X,B\}}{\arg\min} \left[\sum_{i=1}^{N} \left(E_i + \sum_{j=1}^{M} c_{ij}(\|\mathbf{x}_i - \mathbf{b}_j\| - d_{ij}) + \sum_{l=1}^{P_i} D(\mathbf{pc}_i^g, \mathbf{pc}_l^g) \right) \right] \tag{5.9}$$

where P_i is the number of loop closures in which pose i is involved, and the function $D(\mathbf{pc}_i^g, \mathbf{pc}_l^g)$ computes the average Euclidean distance between the given point clouds in the global frame.

Obtaining $D(\mathbf{pc}_i^g, \mathbf{pc}_l^g)$ could have a significant computational cost if the point clouds are large, because we need to calculate the closest 3D point of each cloud into the other, hence slowing down the optimization process. However, assuming that the poses to be optimized are not far from the final estimates thanks to the RO-SLAM step, these associations between the point clouds can be pre-computed. In this way, the function $D(\mathbf{pc}_i^g, \mathbf{pc}_l^g)$ only needs to compute the average distance between 3D points because the associations are already known.

5.3 Experimental Results

An experimental setup has been conceived in order to test the proposed approach. A UAV has been equipped with two RGB-D sensors (Orbbec's Astra), one in the front and another one in the rear side of the robot. Both are tilted down slightly (25°). A UWB sensor has been also installed on top of the aerial robot, and a set of three UWB beacons have been placed in the scenario. The UWB distance measurements have a standard deviation of approximately 20cm, but they are subject to further distortions due to reflexions or sigma attenuation. The UAV used for testing is depicted in Figure 5.2.

The experiments have been carried out once again at CATEC's indoor testbed, in which a scenario recreating an aircraft manufacturing plant has been installed as shown in Figure 5.3. As in previous experiments, accurate ground-truth localization data for the UAV were acquired, as well as the positions of the three UWB beacons installed in the environment, using the testbed's motion capture system.

Figure 5.2: The UAV with RGB-D sensors at the front (left) and the rear (right) side, and a UWB sensor on top.

Figure 5.3: The indoor testbed at CATEC with a mock-up scenario for 3D map building.

Two different flights were recorded. The UAV first performed a short flight in the area in order to acquire sensor data with the purpose of building a 3D map of the environment following the approach described in this chapter. Later on, a much longer flight was carried out in order to gather ground-truth localization data and sensor data so that we could compare the motion capture localization with the output

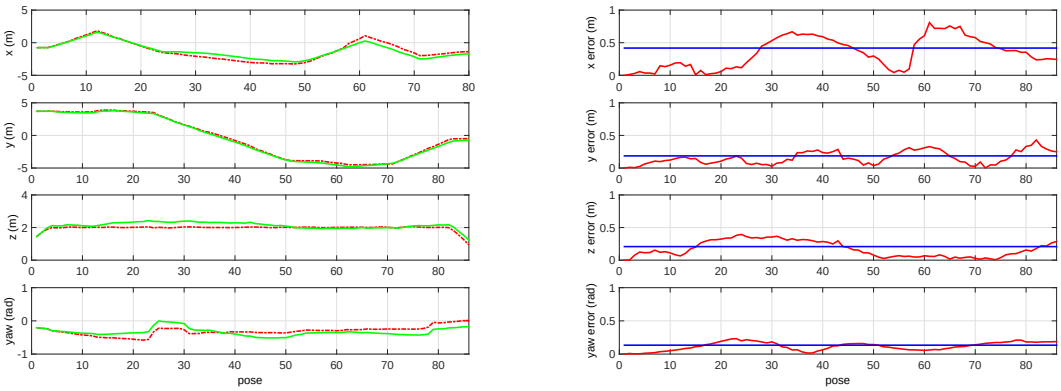

Figure 5.4: Results of the first step: Range-only localization and mapping.

of our particle filter approach described in Chapter 4 using the previously computed map. The following subsections detail the results obtained in each experiment.

5.3.1 Mapping Experiment

The 3D point clouds and UWB-based distance measurements gathered during the "mapping" flight were used to build a pose graph and the multiple hypotheses for the unknown positions of the three UWB beacons. As previously introduced, we measured the z coordinates of the UWB beacons (height with respect the ground), so that only the x and y coordinates of the sensors need to be estimated.

The solution of the first step of the optimization process is shown in Figure 5.4, where the estimation for each of the robot poses is shown together with the associated errors with respect to the ground-truth localization data. Left plots show both the ground-truth (red) and the estimated localization (green), while right plots present the errors per axis (red) and their corresponding RMS errors (blue). We can see how the error in the UAV trajectory is less than 0.29 m in position, and 0.13 rad in yaw angle. The positions of the three UWB beacons converged to a single solution with an RMS error in xy of $0.46m$. Although this is a significant error, it helped the optimizer to compute a globally consistent UAV trajectory, hence improving the loop-closure detection for the second step of the optimization.

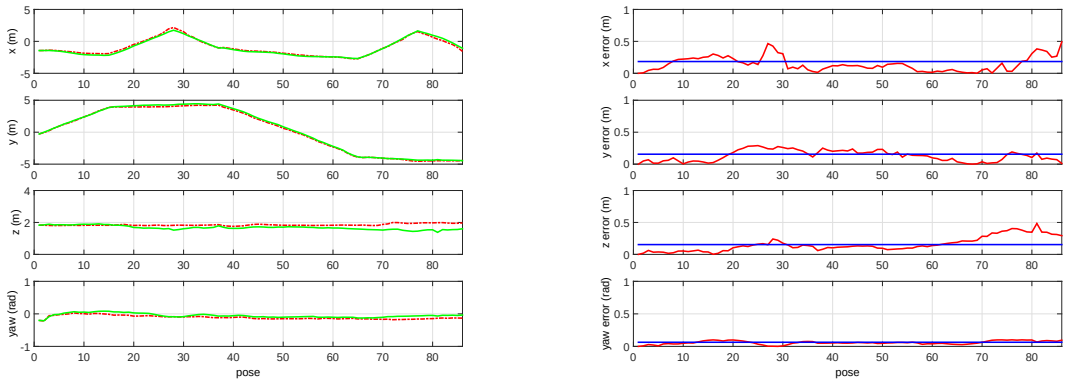

Figure 5.5: Results of the second step: 3D Mapping and Pose Refinement.

Finally, Figure 5.5 shows the results after the second step of the optimization. Again, left plots show the estimated position and yaw of the aerial robot, both the ground-truth (red) and the estimated values (green); right plots present the errors per axis (red) and their corresponding RMS errors (blue). After performing scan matching in the detected loop-closures, we are able to reduce the localization error down to 0.17m RMS in total, while yaw error is reduced to 0.06rad. In addition, the localization error in the position of the UWB beacons is decreased to 0.15m RMS. We can see a comparative analysis of the evolution in the localization estimations of the three UWB beacons along the different optimization steps in Table 5.1.

Table 5.1: UWB Localization Estimations and Errors in XY

	x_1 (m)	y_1 (m)	x_2 (m)	y_2 (m)	x_3 (m)	y_3 (m)	err. (m)
First step	1.87	-6.38	-0.38	6.25	5.41	-1.36	0.46
Second step	1.44	-6.85	-0.01	5.94	5.12	-2.10	0.15
True position	1.20	-6.60	0.02	5.92	5.03	-1.99	-

Once the full UAV trajectory was optimized, the 3D point clouds associated to each pose can then be projected into a single map to reconstruct the robot environment. Figure 5.6 shows the map computed using the ground-truth poses from the motion capture system and the map obtained using the proposed optimization approach. It

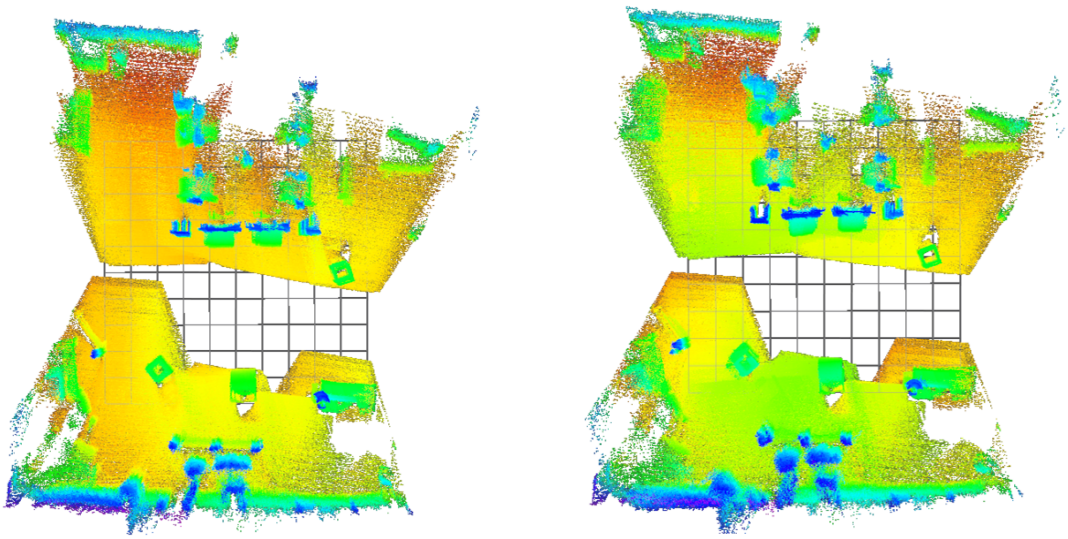

Figure 5.6: Top view of the ground-truth map (left) and reconstructed map (right).

can be observed how there are some errors due to inaccuracies in the trajectory, but the general structure and details are faithfully captured in the reconstruction.

5.3.2 Localization Experiment

The second experiment took place in the same environment in order to use the previously built map for localization. However, the flight time was longer (around 7 minutes) in order to test the long-term capabilities of the proposed localization approach. Figure 5.7 shows the estimated position and yaw angle provided by the particle filter during this flight (green), along with the ground-truth values provided by the motion capture system (red). Roll and pitch angles were acquired directly from the on-board IMU and hence are not shown in the plots, since they are directly integrated into the algorithm.

As it can be seen, the estimations closely follow the ground-truth during all the experiment and do not exhibit drift with time. Except for one of the turns at 360 seconds, where the estimation in X moved away for a moment, being later recovered, the errors were kept approximately bounded. RMS errors of less than 0.3m were

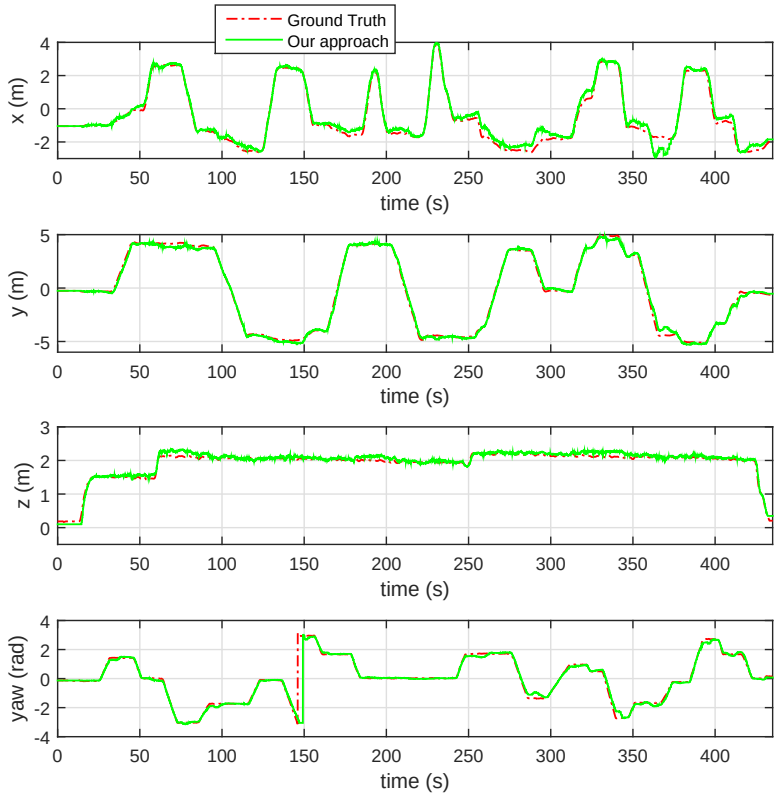

Figure 5.7: Estimated UAV position and yaw angle for the long-term flight.

obtained for x and y, whereas in z we reached 0.14m, largely thanks to the ground estimation performed within the visual odometry algorithm. Yaw RMS error of 0.12 rad (around 7°) is considered acceptable. It is also worth to mention that the UAV traversed areas that were not traversed during the previous "mapping" flight, hence are not included in the map that was used for localization. Figure 5.8 shows a moment of the flight in which the RGB-D point cloud barely matched the existing 3D locations of the map in which the MCL relies on localization estimations. The use of UWB beacons greatly helped when the UAV visited previously unexplored regions, or with different yaw angles that led to unknown viewpoints. In contrast, the map matching allowed a better estimation in z since the position of the UWB beacons did not exhibit

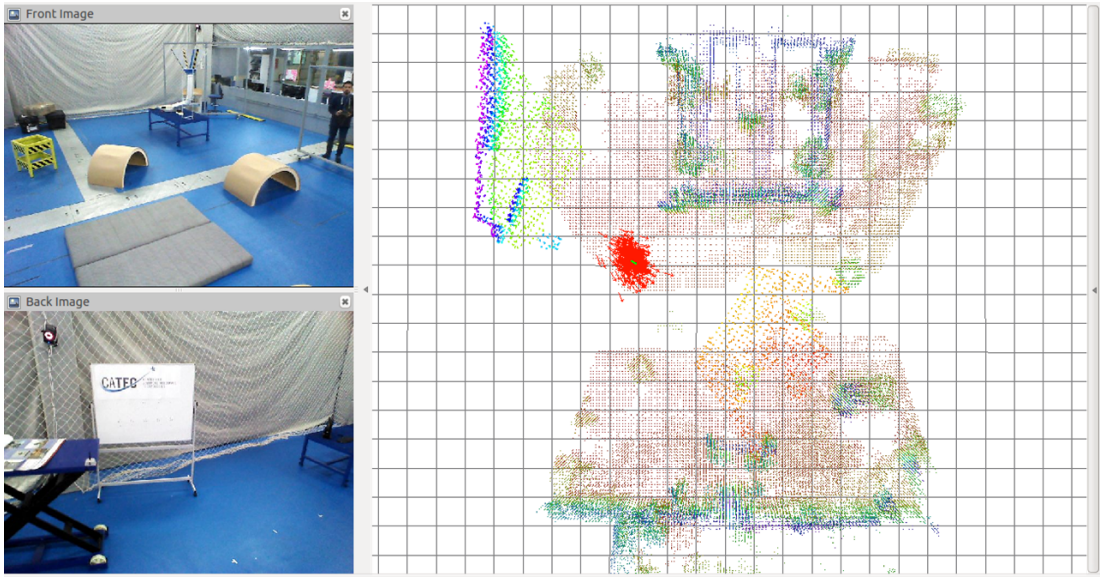

Figure 5.8: 3D visualization of sensor data, along with the particles (red cloud).

much difference in height (due to practical reasons in their installation), and especially in *yaw* since this angle cannot be estimated from UWB sensing.

In order to quantify the contribution of our mapping approach, we have analyzed the same flight data comparing the UAV localization results using the ground-truth map generated with the motion capture UAV localization shown in Figure 5.6. Figure 5.9 presents the errors (red) and their RMS values (blue) during the whole trajectory when using the ground-truth map and when using the proposed approach. Table 5.2 summarizes the RMS errors in position and yaw angle for both runs. As expected, errors are slightly lower when using the ground-truth map. Nevertheless, our results do not show great discrepancies.

Table 5.2: RMS localization errors using different maps

	x (m)	y (m)	z (m)	yaw (rad)
Ground-truth map	0.25	0.25	0.06	0.13
Our map	0.27	0.21	0.14	0.12

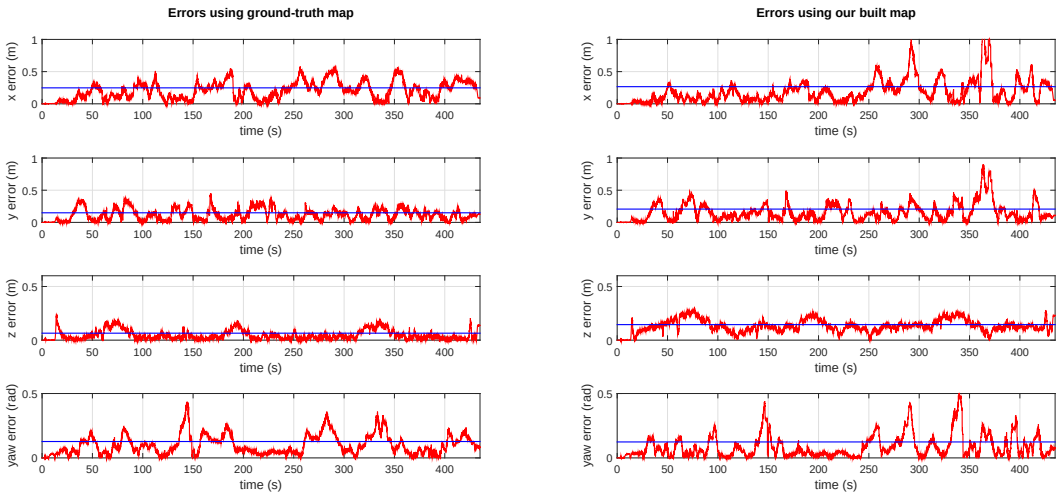

Figure 5.9: Errors in the estimated UAV position and yaw angle using the ground-truth map and the reconstructed map.

5.4 Conclusions

The experimental results show how a multi-sensor suite can be used to perform 3D mapping and long-term localization, integrating sensor readings from UWB beacons and RGB-D cameras to build a robust approach. The mapping algorithm exploits the synergies between UWB and RGB-D to build an accurate 3D map of the environment and to localize the UWB sensors into such map. The localization approach successfully integrates both sensor types to overcome the limitations of each sensor modality by its own.

This mapping approach is oriented to be used with the particle filter presented in Chapter 4, hence providing a full methodology for applying this safe, robust and long-term localization approach to any custom scenario in which small UAVs need to be deployed.

Chapter 6

System Architecture and Framework

In order to simplify the accomplishment of all the tasks related to the autonomous navigation of UAVs, a framework for combining and executing heterogeneous software modules has been developed. There are several open-source projects that target complete software architectures Lim et al. (2012), such as ArduPilot[1] or PX4[2]. However, these approaches usually lack flexibility to perform and support high-level functionalities, which are often demanded by users that wish to develop innovative approaches towards UAV autonomy.

In recent years, several initiatives have arisen from some research groups. The Autonomous Systems Lab from Eidgenössische Technische Hochschule (ETH) Zurich developed *asctec_mav_framework*[3], however it is dependent on the UAV manufacturer. The Paparazzi project[4] from École Nationale de l'Aviation Civile (ENAC) is a complete solution that also includes specific hardware, the Paparazzi autopilot. Another example is *hector_quadrotor*[5] from Technische Universität Darmstadt, which is heavily focused on simulation environments and has limited applicability in real systems performing

[1] http://ardupilot.org
[2] http://px4.io
[3] http://wiki.ros.org/asctec_mav_framework
[4] http://wiki.paparazziuav.org/wiki/Main_Page
[5] http://wiki.ros.org/hector_quadrotor

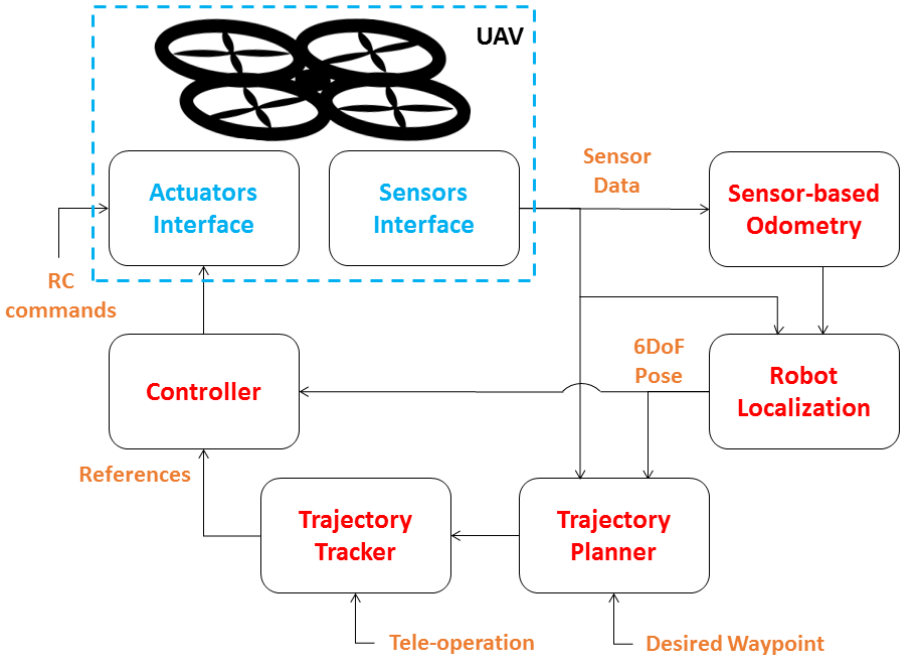

Figure 6.1: Schematic overview of the proposed architecture.

experimental flights. Even though these approaches have produced major advances in this line of research, there are still important challenges related to the achieved level of autonomy and flexibility of the system to be able to adapt it for different aerial robots or applications.

Figure 6.1 shows the software architecture that has been built and refined throughout the different works towards this dissertation. The system is decomposed into several modules which are in charge of different tasks. One of the first things to consider for an autonomous robot is the estimation of the current state of the platform in a specific coordinate system. For this, sensor data is first processed and used to obtain a localization estimation with respect to its environment. This is commonly carried out by an odometry calculation in *Sensor-based Odometry* and a localization estimation in *Robot Localization* to produce a reliable 6 DoF pose (position and orientation). Sensor data is commonly used in both modules in order to reduce uncertainties and improve the robustness of the estimations, especially for long-term operation of the

aerial robot. These two modules have been the specific target of the work described in this dissertation.

Once reliable robot localization is available, subsequent modules can consider tackling more complex tasks, such as where to go or how to reach a specific location. The generation of safe trajectories is another essential capability for autonomous navigation, especially in cluttered areas where several obstacles or even other robots can be found. Realistic operating environments for aerial robots may exhibit fixed or mobile elements and simple or complex obstacles, all of them sharing the same area. Therefore, the aerial robot must continuously conduct a local recalculation of the trajectory in real-time in order to avoid collisions with different types of elements as the on-board sensors gain information about its surroundings. Additionally, the UAV might also need to adopt different collision avoidance strategies depending on the nature of the obstacles; the robot should avoid persons differently than static objects for instance. Sensor data is also acquired by the *Trajectory Planner* for reactive collision avoidance, along with the localization output in order to compute safe trajectories in the aerial robot environment. The *Trajectory Tracker* ensures the UAV reaches the desired waypoints at the correct times, and thus sends the *Controller* the references that it needs in order to calculate the necessary actions for the UAV. Special focus has been given to safety in the operation of the aerial robot, hence the key components in our approach are the perception of the environment for both localization and motion planning. This will allow the UAV to select an appropriate behavior according to the awareness of the current situation.

The core idea behind the architecture is the combination of simple concepts and components to build a reliable system. Each module runs as an independent process on-board the UAV. The different modules can implement a wide variety of algorithms and techniques for performing their task, according to the equipment of the aerial robot. It is important to point out that working with UAVs involves additional constraints in terms of computation and payload capacity which must be taken into account for the development of applicable systems. That said, the different algorithms can be developed independently and switch or replace them within any of the modules to solve the task that is intended to perform. This does not affect the rest of the

architecture provided that the new algorithm complies with the applicable input and output interfaces of the specific module.

Besides, the architecture is focused on the software components, remaining independent from the hardware used underneath. The core algorithms of the system do not depend on the specific manufacturer of the UAV platform or the specific sensors that are used, and hence it is possible to replace them with a different model or even with an upgraded version of the device, implying only minor changes to the system. Apart from that, it is possible to complement the system with additional sensors or enhanced processing capabilities without altering the overall scheme. Moreover, this architecture is able to accommodate applications involving a single or multiple UAVs, as well as highly autonomous operations or teleoperated missions.

Our approach uses as middleware the Robot Operating System (ROS) Quigley et al. (2009), which was developed by Willow Garage and Stanford University as part of the STanford Artificial Intelligence Robot (STAIR) project as a free and open-source robotic middleware for the large-scale development of complex robotic systems. ROS acts as a meta-operating system for robots as it provides hardware abstraction, low-level device control, inter-processes message-passing and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

Currently, ROS is widely used in research activities and is becoming a *de facto* standard for robotic applications. One of the main advantages of ROS is that it allows manipulating sensor data as a labeled abstract data stream, called topic, without having to deal with hardware drivers. The interfaces between our modules are based on standard ROS data types for transformations, pose estimations or trajectory commands. This greatly improves the flexibility of the architecture to allow testing custom high-level algorithms for autonomous operation, while at the same time enables the use of UAVs from different vendors and a wide variety of sensors for which supported drivers are already available. We have made extensive use of the tools available in ROS to help to debug and detect issues in both hardware and software. This includes simulations run in Gazebo, logging and playing back data from different experiments or visualization and inspection tools. This is especially useful when an

innovative approach or algorithm is under development, and there is a potential risk of crashing the platform. Thanks to the use of these tools, it is possible to test the algorithms and modules off-line using real flight data and debug internal states of the UAV or present relevant data in a graphical interface.

As stated in Chapter 1, the main context in which the proposed architecture has been developed and tested is EuRoC, in which high-level semi-autonomous operation of a UAV for inspection tasks has been demonstrated. The aforementioned framework has been designed and implemented in order to perform UAV localization without external positioning systems, among other tasks. Our team GRVC-CATEC successfully completed Stage I of the project (*Qualifying Simulation Contest*), being among the top 15 challenger teams. In order to gain access to Stage II of EuRoC project, an end-user had to be selected in order to actually demonstrate the system capabilities, and in our case, the end-user is Airbus Defence&Space (D&S), a world leader in aircraft manufacturing, interested in the automation of logistic processes in their manufacturing plants. The evaluation of the proposals was carried out by the Challenge Advisory Board (ETH Zurich among other institutions) with the help of external reviewers and renowned independent experts, and it was based on novelty of the application, level of difficulty of the use case, potential market for it, strength of the team, further to the rank in the simulation contest. Our team was granted access to Stage II with the second highest score, and we are currently competing with other four European challenger teams to gain access to the final Stage III. The proposed architecture has been developed in the context of this project, and the UAV has been extensively tested in order to succeed in the different rounds of experiments.

6.1 UAV Platform

The UAV used in EuRoC's Stage II is a research prototype from Ascending Technologies called AscTec Neo.It is a hexacopter with 9" propellers which can lift up to 2Kg, but the maximum nominal flight time of 20 minutes (without payload) will be reduced accordingly. This is a platform developed and optimized within the framework of different European research projects.

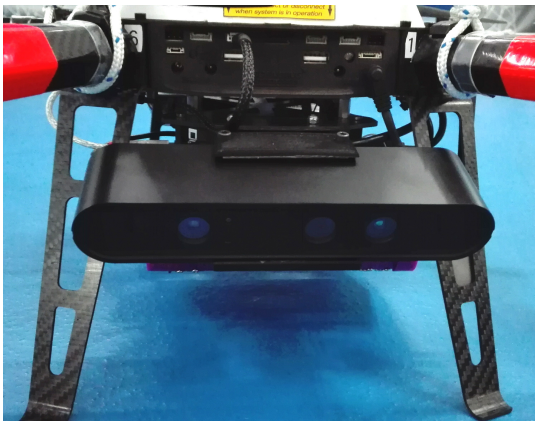

Figure 6.2: The two main sensors tested on-board the UAV: VI-Sensor (left) and Astra (right).

The default main sensor on-board the platform is Skybotix's VI-Sensor, a stereo camera with a calibrated and synchronized IMU. Nevertheless, the original mount has been modified in order to accommodate different sensors. Our architecture has been validated not only with the VI-Sensor, but also using an RGB-D camera as the main sensor, in particular Orbbec's Astra, as it can be seen in Figure 6.2. The sensor is mounted facing forward in the direction of the flight in order to maximize the information gathered regarding possible nearby obstacles. The orientation of the sensor is customizable as we have manufactured different mounts using a 3D printer. Figure 6.3 shows one of the tested mounts. The selected mount will depend on the experiments scenario, the flight height or the type of obstacles that the UAV might encounter in its environment. Depending on the main visual sensor mounted on-board the UAV, the software modules involved with sensor data were adapted accordingly in order to work with the type of data that each sensor delivers.

The on-board embedded computer (Intel Next Unit of Computing (NUC) with Core i7) is in charge of acquiring sensor data, estimating the aerial robot current state, generating the appropriate trajectories according to a given navigation goal and running the autopilot for controlling the UAV. Given all these requirements, special care has been put over all the running modules in order to cut down the computational needs when possible.

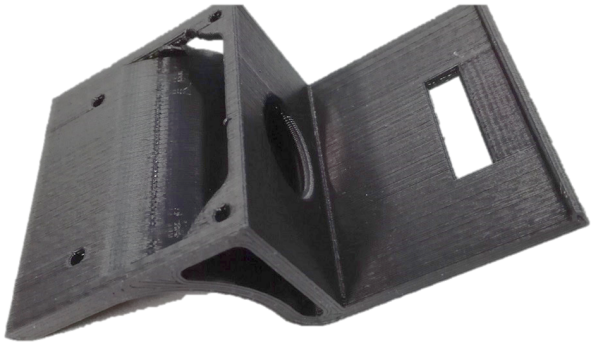

Figure 6.3: A sample of 3D printed mount for the main vision-based sensor on-board the UAV.

It is important to point out that all the results shown in this chapter correspond to different flights in which the control loop is closed using the localization estimations based on on-board sensors, which are computed on-line, unlike previous chapters where data were processed off-line.

6.2 Controlled tests

To succeed in achieving robust autonomous navigation, the aerial robot must demonstrate both reliability and good performance. Extensive field testing has taken place at the indoor testbed of CATEC, shown in Figure 6.4, whose motion capture system tremendously helped us to benchmark the developed algorithms against a very precise ground-truth for robot localization, motion planning or obstacle avoidance with safety distance assurance.

The goal of the controlled tests performed at this indoor testbed was to monitor our development progress and measure flying skills of the aerial robot. We also used the tests to evaluate the effects of changes in hardware and software on the UAV. The flying skills which we have systematically tested to assess our progress were the following:

- Ability of the UAV to autonomously take-off and perform stable hovering.
- Ability of the UAV to accurately follow a pre-planned obstacle-free trajectory.

Figure 6.4: One of the testing scenarios in CATEC's indoor testbed.

- Ability of the UAV to generate an obstacle-free trajectory in order to safely reach a specific waypoint.

- Ability of the UAV to dynamically modify a generated trajectory towards a waypoint to avoid perceived obstacles along the way.

As it is apparent in Figure 6.4, the testbed structure allowed us to secure the UAV from the top via a safety rope to prevent an undesired flight termination. This has been very useful during the development process, as testing new features in the algorithms may sometimes result in unexpected behaviors.

6.3 EuRoC Benchmarking

The first part of Stage II consisted of performing a series of tasks defined by the Challenge Host (ETH Zurich), which were the same for all the teams in the same flying arena with the same UAV platform. This section briefly describes our experience

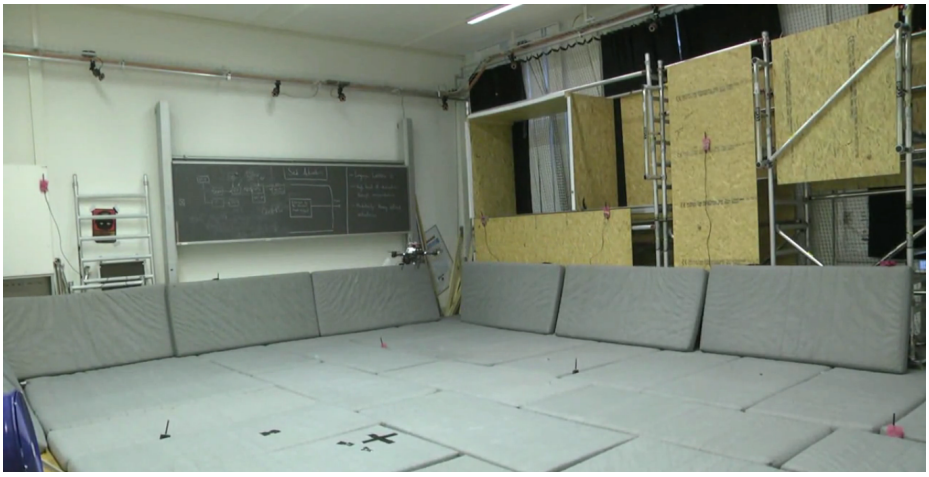

Figure 6.5: ETH's flying arena used in the *Benchmarking* and *Free-Style* rounds.

and results obtained from the experiments carried out during the summer of 2016 regarding the *Benchmarking* round, that took place in ETH Zurich (see Figure 6.5).

The UAV running the proposed architecture has been extensively tested in order to fulfill the proposed tasks, which were meant to guide our development towards high-level autonomous tasks. Our system relied on a *Sensor-based Odometry* based on the visual-inertial odometry algorithm described in Chapter 3, gathering sensor data from the on-board VI-Sensor stereo camera. The *Robot Localization* module used in this experiment round was based on Stereo Parallel Tracking and Mapping (S-PTAM) Pire et al. (2015), a stereo SLAM system which we adapted in order to make it work within our architecture and using our odometry estimations along with the acquired stereo images.

A series of *Benchmarking* experiments were proposed, which consisted of several tasks that the aerial robot should perform, and metrics to evaluate them.

- **Task 1 - System Setup**: this task simply benchmarks the time that teams need to set up the platform out of the transportation box until stable hovering at 1 meter of altitude above the take-off location.

- **Task 2 - State Estimation**: the purpose of this task is to assess the drift of the visual localization system and state estimator. Collision-free random

trajectories are sent to the UAV and have to be followed as accurately as possible. Besides, disturbances and turbulences from wind may be added in the scenario.

- **Task 3 - Waypoint Navigation**: this task benchmarks local map creation as well as planning. First, the arena is explored by following a collision-free trajectory (as in Task 2) that covers the arena quite well, but may leave some unknown spots. After this trajectory is finished, collision-free paths have to be planned to several waypoints with increasing difficulty, and followed by the UAV.

- **Task 4 - Reactive Obstacle Avoidance**: this task benchmarks reactive collision avoidance. Velocity commands are sent to the aerial robot, steering it towards multiple obstacles of increasing difficulty, e.g. walls, small sticking objects or hanging cables. The task is evaluated according to the minimum distance between the UAV and the obstacles during the task execution.

- **Task 5 - Structure Following**: the goal of this task is to cover a specified area at a fixed distance in minimal time. The area can be bent or non-vertical, and the metric is a combination of the error maintaining the desired distance, the accumulated inspected area and the time needed for covering it.

Overall our team performed well in the experiments, according to the report prepared by the Challenge Host showing the results from each task. Their comments are shown in brackets in the following paragraphs.

The setup and stable hovering for Task 1 were achieved in a fast and smooth manner (*No issues and reasonably fast.*). Our state estimation system only drifted 0.2m after the UAV followed a trajectory of approximately 100m in Task 2 (*The odometry system generally worked well, though not quite as accurately as some other systems in the challenge. Winds impact on the system was fairly minor.*). The collision detection in Task 4 was successful for all the types of obstacles, including the hardest one which consisted in a hanging cable with a diameter of 2cm (*It generally worked well.*). Task 5 was achieved by using joystick commands and covering around 80%

of an inclined plane in less than 2 minutes while maintaining the specified distance parallel to the plane with an error of 0.2m (*A serviceable if slightly slow solution.*).

By far the most interesting task was Task 3, since it involved the interaction of all the key modules from our architecture: visual odometry, robot localization, trajectory planner and trajectory tracker. According to the Challenge Host, *the system appeared the most robust at getting to waypoints, though moved slower than some other solutions*. Our core idea is aimed at achieving a robust system in order to enable the safe integration of UAVs in scenarios where they are currently not able to operate. For this reason, since the only source of information for obstacle avoidance was the frontal vision-based sensor, the concept of blindly flying in any direction was discarded from the beginning. Our system would always turn its front sensor towards the direction of the flight in order to ensure a safe trajectory and avoid any obstacle that it may encounter. That is the reason behind the slowness of our system, which in contrast turned out to be the most robust and the only one which aimed at reaching all the waypoints in Task 3. In fact, our team obtained the second highest score among the five challenger teams in the *Benchmarking* round of experiments.

6.4 EuRoC Free-Style

Apart from the *Benchmarking* experiments, another set of flight tests was performed in the Challenge Host testbed within the context of the *Free-Style* experiments. These tests were chosen by our team in order to validate specific functionalities according to our project use case, which is the safe introduction of aerial robots in aircraft manufacturing plants. These tests included tasks related to the interaction of the aerial robot with other agents in its environment, such as human workers or other aerial robots. Hence, obstacle detection and avoidance were basic tasks that our system needed to tackle. Figure 6.6 shows two of the experiments we carried out. In these experiments, the VI-Sensor was again the main visual sensor on-board the UAV for *Sensor-based Odometry*, and S-PTAM was also used for *Robot Localization*.

The first experiment aimed at performing autonomous robot navigation while keeping a safety distance from a human worker in the factory, emulated by a mannequin.

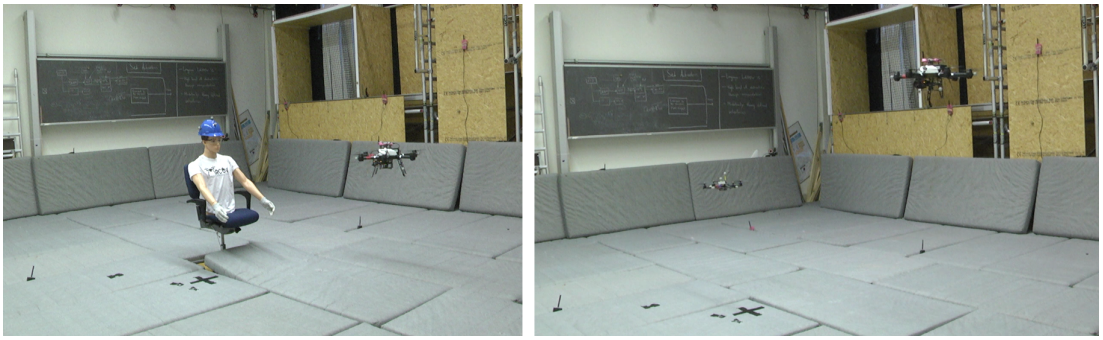

Figure 6.6: The UAV performing obstacle detection and avoidance of a human worker (left) and another smaller UAV (right).

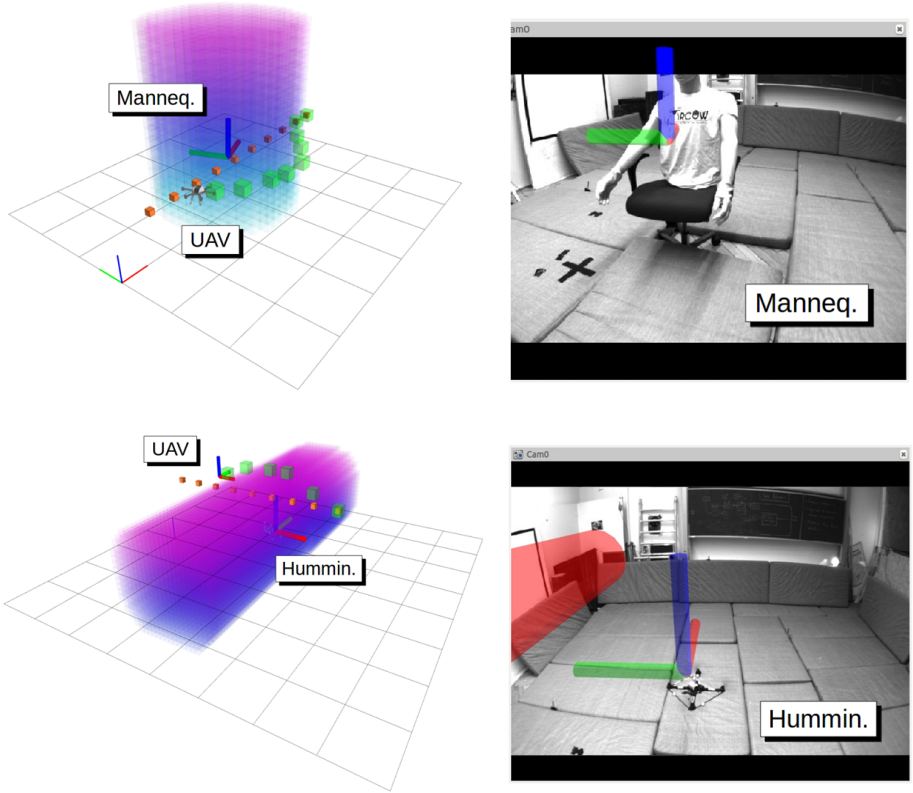

Figure 6.7: Obstacle detection and trajectory replanning results in the mannequin (top) and multi-UAV (bottom) experiments.

A waypoint was provided to the trajectory planner in order to send the UAV 5m forward, enforcing a collision situation where the worker was an obstacle in the original

trajectory commanded to the UAV. Whenever the system detected the worker, the reactive planner automatically recalculated the trajectory in order to always ensure a safety distance from the mannequin, and also change the *yaw* angle so it always remained at sight while it was being avoided, until the UAV returned to the originally generated trajectory. The top part of Figure 6.7 shows the human worker detection and the replanned trajectory around the inflated occupancy matrix centered on the estimated obstacle position. Our target safety distance to maintain from the human worker was set to 1.25m. During the experiment, we achieved an average separation distance from the UAV to the mannequin of 1.35m. Figure 6.8 shows the UAV pose estimation results from the *Robot Localization* module. Left plots show the comparison between the UAV ground-truth captured by the motion capture system (red) and our estimated localization (green). Right plots show the computed error associated with each axis (red) and the RMS error along the trajectory (blue). Changes in *yaw* slightly affected the localization, especially in y axis, but errors, in general, are acceptable, as Table 6.1 shows.

The second experiment involved autonomous conflict avoidance between aerial robots sharing the same space in large factories. A second smaller UAV (AscTec Hummingbird) was used and commanded to perform an off-line pre-planned trajectory, navigating using the motion capture system of the indoor testbed. Again, the trajectory planner was sent a waypoint so that the UAV needed to move 5m forward, and hence the generated trajectory implied a conflict with the one commanded to the smaller UAV, which was sent a perpendicular trajectory. The system detected the obstacle thanks to the visual sensor and then the local planner automatically resolved the conflict by increasing the flight height as it is shown in the bottom part of Figure 6.7. The safety distance to the obstacle was again 1.25m, but the replanning strategy was different due to the height change. During the experiment we achieved an average separation distance (on the xz plane) of 1.32m (with a maximum peak of 1.54m and a minimum peak of 1.14m), very close to the target safe distance. Figure 6.8 shows the UAV pose estimation results, and as it can be observed also in Table 6.1, errors are much lower in this case, especially because the trajectory did not involve considerable changes in *yaw* angle.

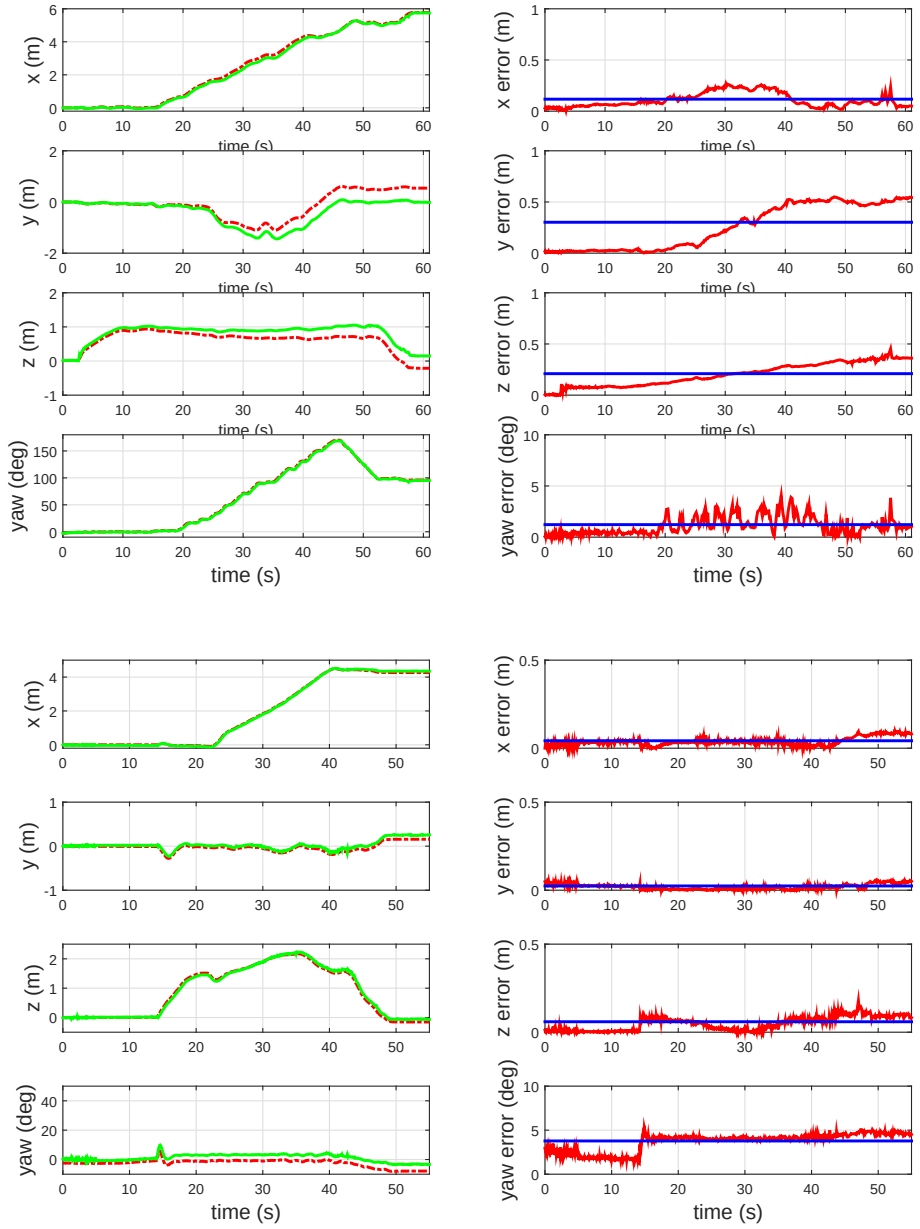

Figure 6.8: Estimated UAV position and orientation in the mannequin (top) and multi-UAV (bottom) experiments.

Our team obtained the highest score among the five challenger teams in the *Free-Style* round, being the only ones to have performed an experiment involving more than one UAV and demonstrating on-line sense and avoid capabilities.

Table 6.1: RMS localization errors in *Free-Style* experiments

	x (m)	y (m)	z (m)	yaw (°)
Mannequin experiment	0.117	0.302	0.209	1.226
Multi-UAV experiment	0.042	0.024	0.059	3.776

6.5 EuRoC Showcase

The main goal of the *Showcase* round is to demonstrate the use cases chosen by our end-user Airbus D&S. The *Benchmarking* and *Free-Style* rounds acted as phases to develop and adapt basic technologies so that in this round the safe introduction of aerial robots in aircraft manufacturing plants can be illustrated. This *Showcase* round has been validated in a controlled environment emulating the scenario of an aircraft manufacturing plant or aircraft assembly line. This scenario was replicated in the indoor testbed of CATEC's facilities, where industrial scenarios have been already replicated in other projects. Figure 6.9 shows both the actual and the replicated environments.

Airbus D&S identified two potential activities where aerial robots could improve productivity by reducing operation times and manufacturing costs: logistic process of light goods, and localization and identification of missing or forgotten tools that could cause damage to structures. The provided UAV for EuRoC's Stage II was modified for the *Showcase* round. The VI-Sensor that was the main source of information for vision-based navigation was replaced by an RGB-D camera. Using this sensor allowed us to obtain further and more reliable depth estimations while working indoors, which is the case in our application. Moreover, a second RGB-D camera was also installed facing backward in order to improve the UAV situational awareness for obstacle detection and avoidance. Both cameras are shown in Figure 6.10. Thanks to this sensor configuration, the on-board trajectory planner was able to compute safe trajectories both in forward and backward directions, depending on where the desired target location is with respect to the UAV current state.

Figure 6.9: Airbus D&S manufacturing plant (top) and replicated environment at CATEC (bottom).

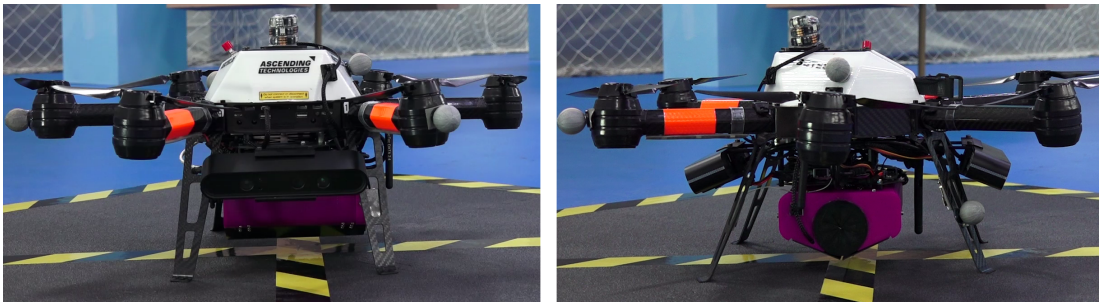

Figure 6.10: RGB-D sensors on-board the UAV and signaling lights.

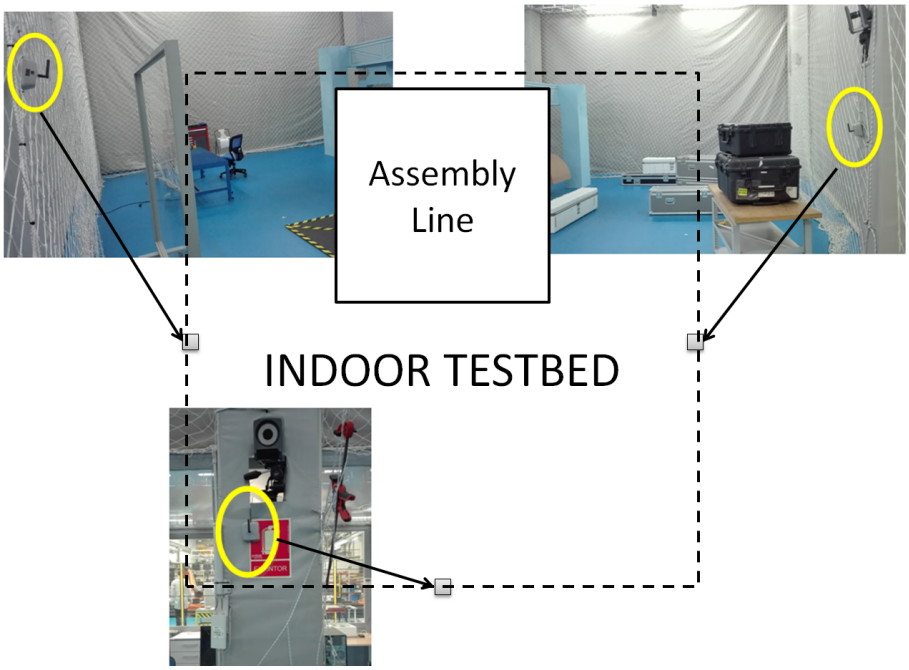

Figure 6.11: UWB beacons installed in the indoor testbed.

In order to increase the awareness of the workers in the aircraft manufacturing plant regarding the operation of a UAV, a series of LED lights were also installed on the aerial platform. Blinking red lights have been placed on top (see Figure 6.10), while fixed red lights are below each of the two front arms.

Another modification in the general navigation system was the integration of range measurements in order to increase the robustness of the UAV localization estimation for long-term operation. Three UWB sensors have been installed in the indoor testbed (see Figure 6.11) in order to provide low-frequency distance measurements with respect to another UWB sensor that was mounted on-board the UAV. Such measurements help to reduce the uncertainty regarding the current UAV pose. In this round of experiments, the *Sensor-based Odometry* was the one introduced in Chapter 3 using an RGB-D camera, and the *Robot Localization* was the particle filter using both RGB-D and UWB sensing as explained in Chapter 4.

Two of the proposed objectives for the *Showcase* round required accurate and long-term localization capabilities, and are the ones explained in detail in this section.

Figure 6.12: Small cargo bay on the UAV (left) and hopper for autonomous delivery (right).

6.5.1 Autonomous Delivery System

This objective deals with the logistic process of light goods in the manufacturing plant. In order to do that, a pick-up and delivery system has been designed and installed on both the UAV and the scenario. A small cargo bay has been designed and 3D printed in order to accommodate small goods for aerial transportation. Besides, a hopper has been placed in the realistic scenario in order to represent one of the delivery points to be installed in the aircraft manufacturing plant, as depicted in Figure 6.12.

A worker that is performing some technical work requests a specific component through the use of a Human Machine Interface (HMI). The logistic operator receives the warning and places the requested component in the UAV's cargo bay. The UAV performs a flight taking-off and landing in a pre-defined location in the scenario, successfully delivering the requested component in the hopper. Different waypoints were commanded to the UAV in order to autonomously take-off, reach the hopper and go back to the take-off location. Both the approach to the hopper and the return to the take-off location involved obstacle detection and local re-planning in order to avoid the blue column that is closer to the take-off location. The UAV successfully delivered the requested component while safely navigating to and from the hopper.

One of the biggest challenges involved in this objective is achieving precise point-to-point autonomous navigation, relying only on data acquired on-board the UAV. The top opening of the hopper is a square of $1x$1m, which enforces a UAV localization error of less than 0.5m in the horizontal plane. Regarding height, it is required to

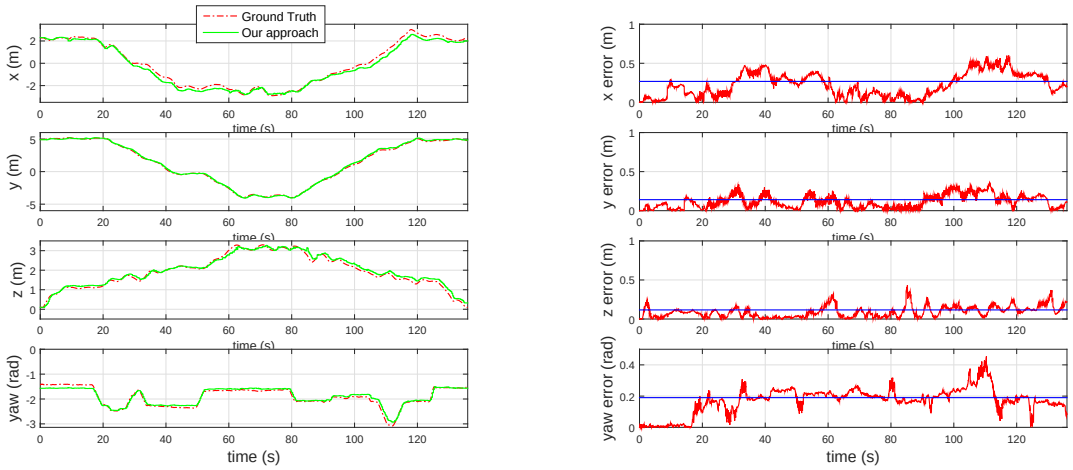

Figure 6.13: Estimated UAV position and orientation in the autonomous delivery experiment.

reach an altitude of at least 3m over the ground, which can be also challenging in a large scenario where obstacles and visual features of reference may be far away from the aerial robot.

Figure 6.13 shows the localization results of this flight, compared with ground-truth data from CATEC's testbed. Left plots show the ground truth (red) and our localization estimations (green), while right plots show errors per axis (red) and RMS errors (blue). It can be seen how x and y errors are below 0.5m for almost the whole trajectory, and especially when the UAV was above the hopper approximately during the middle of the flight (at second 70) at more than 3m of height. RMS errors for each axis are provided in Table 6.2. Errors in z are lower thanks to the use of the ground plane estimator.

6.5.2 Missing Item Detection

This objective involves the end-user use case related to parts monitoring that can be forgotten inside aeronautic components and could cause serious safety issues. The UAV needed to perform a long flight around the whole scenario representing other logistic operations, and at the same time it was able to detect and identify a missing

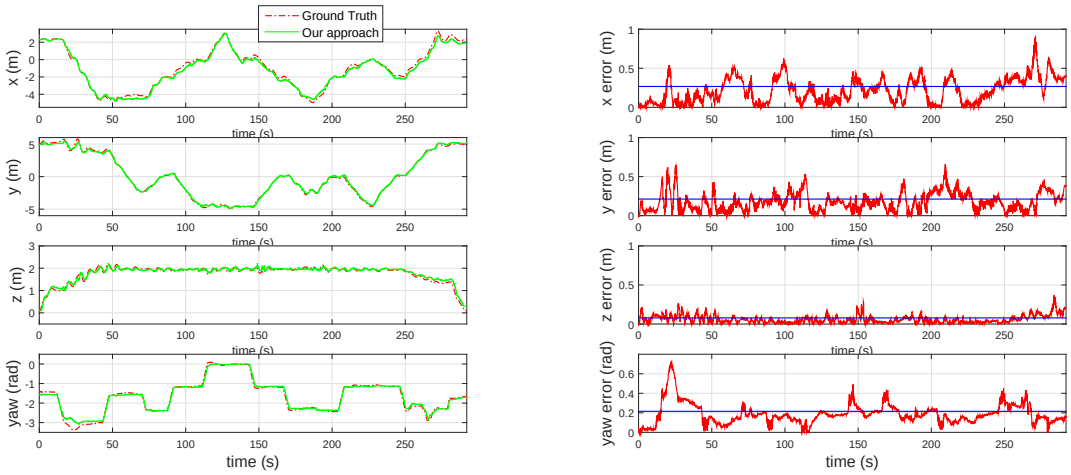

Figure 6.14: Estimated UAV position and orientation in the missing item detection experiment.

object that could be anywhere in the manufacturing plant, based on measurements between additional radio-tags both on the UAV and the missing tool. This radio-based detection algorithm was out of the scope for this dissertation, but precise point-to-point autonomous navigation was again an important pre-requisite for the success of this objective, and hence accurate UAV localization was essential for the task, not only for waypoint navigation, but also for accurate determination of the position of the missing radio-tag.

Figure 6.14 shows the localization results of this flight compared with ground-truth data from CATEC's testbed. Again, left plots show the ground truth (red) and our localization estimations (green), while right plots show errors per axis (red) and RMS errors (blue). It can be seen how the position estimations closely follow the ground truth during the whole flight, which roughly took 5 minutes. RMS errors in x and y are below 0.3m, while z errors are again much lower due to the fact that we are exploiting the sensor point clouds to estimate the ground plane. Table 6.2 shows these results. It can be noted how errors are slightly higher than those shown in the autonomous delivery experiment, since the flight duration of this experiment was also considerably longer. Nevertheless, the UAV is able to successfully reach the specified

waypoints in a robust manner and safely navigate detecting and avoiding possible obstacles on its way.

Table 6.2: RMS localization errors in *Showcase* experiments

	x (m)	y (m)	z (m)	yaw (rad)
Autonomous delivery	0.27	0.14	0.12	0.19
Missing item detection	0.27	0.21	0.08	0.22

6.6 Conclusions

This section describes the framework in which we have tested the performance of the localization approach developed in the context of this dissertation. State estimation of the UAV is the first step of a data processing pipeline that also integrates trajectory planning, obstacle detection and avoidance and automatic control of the aerial vehicle. Performing all these processing steps on-line and on-board the UAV is a very difficult challenge that we have successfully accomplished, thanks to different optimizations in terms of computational efficiency of the algorithms described throughout this document.

The EuRoC project provides a very convenient context in which we have demonstrated accurate, robust and safe autonomous navigation of a small UAV in GPS-denied areas. Moreover, the competitive nature of the project has allowed us to compare our progress and performance with that from other research teams all over Europe, outperforming most of them during the competition.

Chapter 7

Discussion and Conclusions

7.1 Conclusions of this Dissertation

In this dissertation, the problem of robot localization has been thoroughly discussed, with a particular focus on aerial robots. Several techniques have been researched, developed and applied to solve the problem of state estimation of noisy dynamic systems, primarily based on 3D measurements of the robot environment using a variety of on-board sensors. Different sensing approaches and data processing techniques have been also discussed and explained, along with comprehensive experimental results in order to demonstrate their effectiveness and shortcomings. In this way, subsequent modifications and improvements in the on-board system and the data processing approaches have been built on the experience gained after extensive field testing along the past years, as detailed in Chapters 3 and 4.

The final system configuration is an implementation of a particle filter suitable for safe, reliable, computationally efficient and long-term operation of aerial robots in 3D environments. The proposed methodology makes use of a multi-sensor suite that integrates a visual odometry algorithm and point clouds obtained from a 3D imaging sensor (a stereo or RGB-D camera) and distance measurements from radio-based sensors (UWB beacons) to overcome the limitations of each sensing modality by its own in order to build a robust approach.

Moreover, a mapping algorithm is proposed in order to be able to deploy the proposed localization approach in any custom environment, as described in Chapter 5. The novel mapping method exploits the synergies between radio-based and 3D imaging sensing to build an accurate 3D map of the environment and to localize the radio sensors into such map.

The developments included in this dissertation have been extensively tested using a UAV platform to perform autonomous navigation in GPS-denied areas. Experimental results show a strong overall system performance as well as the feasibility of the approach, both in accuracy and computational efficiency. The robustness of the approach has been demonstrated in different campaigns during controlled tests at CATEC's indoor testbed, and especially in the context of the EuRoC project through the demonstration of multiple experiments performing on-line localization estimation, as explained in Chapter 6. The validation of the approach has been presented using a motion capture system as a source of ground-truth data for UAV position and orientation.

7.2 Lessons Learned

Field testing is one of the most expensive tasks when it comes to developing an autonomous system because the aerial platform must be fully operational before carrying out flight experiments, a space for testing must be accessible, and specific qualified staff must be available (including a safety pilot). Besides, if testing outdoors, the weather must behave and, depending on national regulations, a valid license for the UAV pilot might be required. Fortunately, the latter requirement did not play a major role in this dissertation, since all the experiments took place in indoor environments.

An important factor to take into consideration when testing aerial robots is their flight autonomy. Degradation of batteries is a significant issue that we have encountered during extensive test campaigns, leaving us with limited flight times. Moreover, the uniqueness of the aerial platform under test (a research prototype from Ascending Technologies) made us depend heavily on the UAV manufacturer, whose late response forced us to manage custom battery replacements.

Simulation environments such as Gazebo[1] have been very useful for testing the behavior and integration of different modules, especially those related to motion planning. However, simulations are still limited in order to provide accurate information regarding real world scenarios or behaviors of actual systems.

A very convenient tool that has been extensively used during almost all the testing experiments was the *node_manager* developed by Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie (FKIE)[2]. It is a graphical interface to manage nodes, topics, services, parameters and launch files in a ROS network. It seamlessly handles the discovery of running ROS masters in the local network and the synchronization of the local ROS master to the discovered remote masters. This is very useful because all the algorithms are running in the on-board computer, and at the same time, the ground computer can run visualization and debugging tools in order to watch live sensor data, or check the expected behavior of specific algorithms that are under test.

The ability to evaluate the performance of the aerial robot in an experiment is a key factor in field testing. Sometimes a tight time schedule has forced us to carry out extensive test campaigns, during which it was not unusual to adjust several parameters *on the fly*. This leaves room for subjectivity, since human errors can easily arise. It should be a good idea to develop automated methods to quantitatively assess recorded data in order to evaluate the aerial robot performance.

7.3 Future Work

While the system performs well, there is still room for improvement in its performance and the development process.

The proposed localization approach provides a reliable and accurate 4D (x, y, z and ψ) estimation during several minutes of flight. In view of these results, we think this algorithm could be used for longer periods of time, but this should be evaluated in the future with longer experiments.

[1] http://gazebosim.org
[2] http://wiki.ros.org/node_manager_fkie

Future work will also consider performing visual odometry using the current UAV platform setup, which counts with two RGB-D cameras, since it is currently based only on the front facing camera. Currently, the rear-facing camera is only used for obstacle detection, but it could incorporate valuable information in order to improve the robustness of the odometry algorithm, especially when the scene in front of the UAV does not show enough texture or the objects may be at higher distances.

Our state estimation algorithms assume that the vehicle moves relatively slowly (around 30 cm/s). As the vehicle flies faster, the algorithms will likely need to handle larger amounts of motion blur. This would imply changes in the processing pipeline, especially in the visual odometry approach, in order to reduce uncertainties when frame matching is not enough to estimate a valid pose change. Nevertheless, the particle filter has been designed to handle this situation due to the reliability of the radio-based measurements, but this needs to be tested during high-speed motions of the UAV platform.

Although the presented approach is able to estimate the robot localization with small errors, it could be improved with other localization inputs such as visual place recognition, in order to reduce uncertainties when revisiting a location. This could be useful in the visual odometry pipeline and also in the mapping approach.

Moreover, the integration of a low-weight altimeter can reduce the computational load of the processing pipeline regarding height estimation, since this is currently performed from 3D point cloud processing. The inclusion of distances measurements from the UAV to the ground can be easily implemented in the particle filter.

Developing this system for an autonomous robot has allowed us to test a platform that is intended to operate in real situations. Future work will also focus on taking into account that such systems would perform autonomous navigation in any scenario, not only the ones that have been considered so far, and which in some cases might represent simplified versions of real world environments.

These proposed improvements should not be taken lightly in order to consider requirements with respect to the smoothness of the localization estimation, the computational efficiency of the whole processing pipeline and the UAV payload requirements.

Bibliography

Acevedo, J. J., Arrue, B. C., Díaz-Báñez, J. M., Ventura, I., Maza, I., and Ollero, A. (2013). Decentralized strategy to ensure information propagation in area monitoring missions with a team of UAVs under limited communications. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 565–574.

Achtelik, M. W., Lynen, S., Weiss, S., Kneip, L., Chli, M., and Siegwart, R. (2012). Visual-inertial SLAM for a small helicopter in large outdoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2651–2652.

Alahi, A., Ortiz, R., and Vandergheynst, P. (2012). Freak: Fast retina keypoint. In *IEEE conference on Computer vision and pattern recognition (CVPR)*, pages 510–517. IEEE.

Angeli, A., Filliat, D., Doncieux, S., and Meyer, J.-A. (2008). Fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions on Robotics*, 24(5):1027–1037.

Asus (2017). Xtion PRO LIVE. `https://www.asus.com/es/3D-Sensor/Xtion_PRO_LIVE`. [Online; accessed 01-May-2017].

Bajcsy, R. and Lieberman, L. (1976). Texture gradient as a depth cue. *Computer Graphics and Image Processing*, 5(1):52–67.

Besl, P. J. and McKay, N. D. (1992). Method for registration of 3-D shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics.

Blanco, J. L., Gonzalez, J., and Fernandez-Madrigal, J. A. (2008). A pure probabilistic approach to range-only SLAM. In *IEEE International Conference on Robotics and Automation*, pages 1436–1441.

Bolognesi, M., Furini, A., Russo, V., Pellegrinelli, A., and Russo, P. (2015). Testing the low-cost RPAS potential in 3D cultural heritage reconstruction. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(5):229.

Boots, B. and Gordon, G. (2013). A spectral learning approach to range-only SLAM. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 19–26, Atlanta, Georgia, USA. PMLR.

Boudjit, K., Larbes, C., and Alouache, M. (2008). Control of flight operation of a quad rotor AR.Drone using depth map from Microsoft Kinect sensor. *International Journal of Engineering and Innovative Technology (IJEIT)*, 3:15–19.

Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298.

Cary, L. (2011). Unmanned Aircraft Systems (UAS). *International Civil Aviation Authority (ICAO)*, pages Cir 328, AN/190.

Cavoukian, A. (2012). *Privacy and drones: Unmanned aerial vehicles*. Information and Privacy Commissioner of Ontario, Canada.

Cesetti, A., Frontoni, E., Mancini, A., Zingaretti, P., and Longhi, S. (2010). A Vision-Based Guidance System for UAV Navigation and Safe Landing using Natural Landmarks. *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, U.S.A. June 8–10, 2009*, pages 233–257.

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155.

Chen, Z. and Birchfield, S. T. (2006). Qualitative vision-based mobile robot navigation. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)*, pages 2686–2692.

Chong, C.-C., Watanabe, F., and Inamura, H. (2006). Potential of uwb technology for the next generation wireless communications. In *IEEE Ninth International Symposium on Spread Spectrum Techniques and Applications*, pages 422–429.

CNN (2016). CNN receives FAA approval to fly UAS over people in the United States. `http://cnnpressroom.blogs.cnn.com/2016/08/29/cnn-receives-faa-approval-to-fly-uas-over-people-in-the-united-states`. [Online; accessed 01-May-2017].

Cory, R. and Tedrake, R. (2008). Experiments in fixed-wing UAV perching. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, page 7256.

Cox, I. J. (1991). Blanche - an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204.

Cui, J. Q., Phang, S. K., Ang, K. Z., Wang, F., Dong, X., Ke, Y., Lai, S., Li, K., Li, X., Lin, J., et al. (2016). Search and rescue using multiple drones in post-disaster situation. *Unmanned Systems*, 4(01):83–96.

Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

DJI (2016). Guidance: a revolutionary visual sensing system for aerial platforms. `https://www.dji.com/guidance`. [Online; accessed 01-May-2017].

Droeschel, D., Nieuwenhuisen, M., Beul, M., Holz, D., Stückler, J., and Behnke, S. (2016). Multilayered mapping and navigation for autonomous micro aerial vehicles. *Journal of Field Robotics*, 33(4):451–475.

Dryanovski, I., Valenti, R. G., and Xiao, J. (2013). Fast visual odometry and mapping from RGB-D data. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2305–2310.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An Evaluation of the RGB-D SLAM System. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1691–1696.

EYIF (2017). The most innovative European drone companies get awarded at the European Parliament. `http://younginnovator.eu/2017/01/the-most-innovative-european-drone-companies-get-awarded-at-the-european-parliament`. [Online; accessed 01-May-2017].

Ezequiel, C. A. F., Cua, M., Libatique, N. C., Tangonan, G. L., Alampay, R., Labuguen, R. T., Favila, C. M., Honrado, J. L. E., Canos, V., Devaney, C., et al. (2014). UAV aerial imaging applications for post-disaster assessment, environmental management and infrastructure development. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 274–283. IEEE.

Fabresse, F., Caballero, F., Merino, L., and Ollero, A. (2016). Active perception for 3D Range-only Simultaneous Localization and Mapping with UAVs. In *Proceedings of the International Conference on Unmanned Aircraft Systems, ICUAS*, pages 1–6.

Fabresse, F. R., Caballero, F., Maza, I., and Ollero, A. (2013). Undelayed 3D RO-SLAM based on Gaussian-mixture and reduced spherical parametrization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1555–1561.

Fiala, M. (2004). Vision guided control of multiple robots. In *First Canadian Conference on Computer and Robot Vision*, pages 241–246. IEEE.

Fox, D. (2001). KLD-sampling: Adaptive particle filters. In *Neural Information Processing Systems (NIPS)*, volume 14, pages 713–720.

Fraundorfer, F. and Scaramuzza, D. (2012). Visual odometry: Part II: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90.

Gao, X.-S., Hou, X.-R., Tang, J., and Cheng, H.-F. (2003). Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943.

Garcia, J. and Zalevsky, Z. (2008). Range mapping using speckle decorrelation (US Patent 7,433,024). [Online; accessed 01-May-2017].

Geiger, A., Ziegler, J., and Stiller, C. (2011). StereoScan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968.

Geng, J. (2011). Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*, 3(2):128–160.

George, M. and Sukkarieh, S. (2005). Tightly coupled INS/GPS with bias estimation for UAV applications. In *Proceedings of Australiasian Conference on Robotics and Automation (ACRA)*.

Gezici, S., Tian, Z., Giannakis, G. B., Kobayashi, H., Molisch, A. F., Poor, H. V., and Sahinoglu, Z. (2005). Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *IEEE signal processing magazine*, 22(4):70–84.

González, J., Blanco, J.-L., Galindo, C., Ortiz-de Galisteo, A., Fernandez-Madrigal, J.-A., Moreno, F. A., and Martínez, J. L. (2009). Mobile robot localization based on ultra-wide-band ranging: A particle filter approach. *Robotics and autonomous systems*, 57(5):496–507.

Grisetti, G., Kuemmerle, R., Stachniss, C., and Burgard, W. (2010). A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43.

Gu, Y., Lo, A., and Niemegeers, I. (2009). A survey of indoor positioning systems for wireless personal networks. *IEEE Communications surveys & tutorials*, 11(1):13–32.

Guo, K., Qiu, Z., Miao, C., Zaini, A. H., Chen, C.-L., Meng, W., and Xie, L. (2016). Ultra-wideband-based localization for quadcopter navigation. *Unmanned Systems*, 4(01):23–34.

Gupte, S., Mohandas, P. I. T., and Conrad, J. M. (2012). A survey of quadrotor unmanned aerial vehicles. In *Proceedings of IEEE Southeastcon*, pages 1–6.

Hai, D., Li, Y., Zhang, H., and Li, X. (2010). Simultaneous localization and mapping of robot in wireless sensor network. In *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, volume 3, pages 173 –178.

Han, J., Shao, L., Xu, D., and Shotton, J. (2013). Enhanced Computer Vision with Microsoft Kinect Sensor: A Review. *IEEE transactions on cybernetics*, 43(5):1318–1334.

Hausamann, D., Zirnig, W., Schreier, G., and Strobl, P. (2005). Monitoring of gas pipelines–a civil UAV application. *Aircraft Engineering and Aerospace Technology*, 77(5):352–360.

Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 807–814 vol. 2.

Honegger, D., Meier, L., Tanskanen, P., and Pollefeys, M. (2013). An open source and open hardware embedded metric optical flow CMOS camera for indoor and outdoor applications. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1736–1741.

Horn, B. K. and Brooks, M. J. (1989). *Shape from shading*. MIT press.

Hornung, A., Wurm, K. M., and Bennewitz, M. (2010). Humanoid Robot Localization in Complex Indoor Environments. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

Howard, R. M. and Kaminer, I. (1995). Survey of unmanned air vehicles. In *Proceedings of the American Control Conference*, volume 5, pages 2950–2953.

Intel (2015). Intel Keynote CES 2015: Firefly Drones and Intel RealSense Technology. `http://www.intel.com/content/www/us/en/events/ces-2015-intel-keynote-realsense-technology-asctec-drones-video.html`. [Online; accessed 01-May-2017].

JAXA (2017). Asteroid Explorer "HAYABUSA" (MUSES-C). `http://global.jaxa.jp/projects/sat/muses_c`. [Online; accessed 01-May-2017].

Jenkins, D. and Vasigh, B. (2013). The Economic Impact of Unmanned Aircraft Systems Integration in the United States. *Association for Unmanned Vehicle Systems International (AUVSI)*.

Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45.

Kantor, G. and Singh, S. (2002). Preliminary results in range-only localization and mapping. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 2, pages 1818–1823.

Kehagias, A., Djugash, J., and Singh, S. (2006). Range-only SLAM with interpolated range data. Technical report, Carnegie Mellon University (CMU-RI-TR-06-26).

Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for RGB-D cameras. In *IEEE International Conference on Robotics and Automation*, pages 3748–3754.

Khalajmehrabadi, A., Gatsis, N., and Akopian, D. (2017). Modern WLAN Fingerprinting Indoor Positioning Methods and Deployment Challenges. *IEEE Communications Surveys Tutorials*, PP(99):1–1.

Kitt, B., Geiger, A., and Lategahn, H. (2010). Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 486–492.

Kurt-Yavuz, Z. and Yavuz, S. (2012). A comparison of EKF, UKF, FastSLAM2.0, and UKF-based FastSLAM algorithms. In *IEEE 16th International Conference on Intelligent Engineering Systems (INES)*, pages 37 –43.

Labbe, M. and Michaud, F. (2014). Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2661–2666.

Lad, M. (2004). Ultra-Wideband: The Next Generation Personal Area NEtwork Technology. *Patni Computer Systems Limited*.

Lee, K. S., Ovinis, M., Nagarajan, T., Seulin, R., and Morel, O. (2015). Autonomous patrol and surveillance system using unmanned aerial vehicles. In *2015 IEEE 15th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1291–1297.

Lepetit, V., Moreno-Noguer, F., and Fua, P. (2008). EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166.

Leutenegger, S., Chli, M., and Siegwart, R. Y. (2011). BRISK: Binary Robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555.

Li, J., Cheng, L., Wu, H., Xiong, L., and Wang, D. (2012). An overview of the simultaneous localization and mapping on mobile robot. In *Proceedings of International Conference on Modelling, Identification Control (ICMIC)*, pages 358 –364.

Lim, H., Park, J., Lee, D., and Kim, H. J. (2012). Build Your Own Quadrotor: Open-Source Projects on Unmanned Aerial Vehicles. *IEEE Robotics Automation Magazine*, 19(3):33–45.

Lin, C. E. and Yang, S. K. (2014). Camera gimbal tracking from UAV flight control. In *CACS International Automatic Control Conference*, pages 319–322.

Liu, H., Darabi, H., Banerjee, P., and Liu, J. (2007). Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1067–1080.

Lourakis, M. I. and Argyros, A. A. (2009). SBA: A software package for generic sparse bundle adjustment. *ACM Transactions on Mathematical Software (TOMS)*, 36(1):2.

Low, K.-L. (2004). Linear least-squares optimization for point-to-plane ICP surface registration. *Chapel Hill, University of North Carolina*, 4.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P., and Milford, M. J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19.

Lupashin, S., Hehn, M., Mueller, M. W., Schoellig, A. P., Sherback, M., and D'Andrea, R. (2014). A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics*, 24(1):41–54.

Mao, G., Drake, S., and Anderson, B. D. O. (2007). Design of an Extended Kalman Filter for UAV Localization. In *Information, Decision and Control*, pages 224–229.

Marconi, L., Basile, F., Caprari, G., Carloni, R., Chiacchio, P., Hurzeler, C., Lippiello, V., Naldi, R., Nikolic, J., Siciliano, B., et al. (2012). Aerial service robotics: The AIRobots perspective. In *2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 64–69.

Marden, S. and Guivant, J. (2012). Improving the Performance of ICP for Real-Time Applications using an Approximate Nearest Neighbour Search. In *Australasian Conference on Robotics and Automation*.

Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 300–307.

Mautz, R. and Tilch, S. (2011). Survey of optical indoor positioning systems. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7. IEEE.

Maza, I., Caballero, F., Capitán, J., Martínez-de Dios, J. R., and Ollero, A. (2011). Experimental results in multi-UAV coordination for disaster management and civil security applications. *Journal of Intelligent & Robotic Systems*, 61(1):563–585.

Mebarki, R., Lippiello, V., and Siciliano, B. (2016). Vision-based and IMU-aided scale factor-free linear velocity estimator. *Autonomous Robots*, pages 1–15.

Merino, L., Caballero, F., Martínez-de Dios, J. R., Maza, I., and Ollero, A. (2012). An unmanned aircraft system for automatic forest fire monitoring and measurement. *Journal of Intelligent & Robotic Systems*, 65(1):533–548.

Mesas-Carrascosa, F. J., Notario-García, M. D., de Larriva, J. E. M., de la Orden, M. S., and Porras, A. G.-F. (2014). Validation of measurements of land plot area using UAV imagery. *International Journal of Applied Earth Observation and Geoinformation*, 33:270–279.

Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V. (2010). The GRASP Multiple Micro-UAV Testbed. *IEEE Robotics & Automation Magazine*, 17(3):56–65.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., et al. (2008). Junior: The Stanford entry in the Urban Challenge. *Journal of Field Robotics*, 25(9):569–597.

Moravec, H. and Elfes, A. (1985). High resolution maps from wide angle sonar. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121.

Moré, J. J., Garbow, B. S., and Hillstrom, K. E. (1980). User guide for MINPACK-1. Technical report, CM-P00068642.

Mur-Artal, R., Montiel, J. M. M., and Tardos, J. D. (2015). ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Nanotron (2017). Swarm bee ER. `http://www.nanotronshop.com/swarm-c-29.html`. [Online; accessed 01-May-2017].

Nayar, S. K. and Nakagawa, Y. (1994). Shape from focus. *IEEE Transactions on Pattern analysis and machine intelligence*, 16(8):824–831.

Newman, P. and Leonard, J. (2003). Pure range-only sub-sea SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1921–1926.

Nex, F. and Remondino, F. (2014). UAV for 3D mapping applications: a review. *Applied Geomatics*, 6(1):1–15.

Nieuwenhuisen, M., Droeschel, D., Beul, M., and Behnke, S. (2014). Obstacle detection and navigation planning for autonomous micro aerial vehicles. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1040–1047. IEEE.

Nuchter, A., Lingemann, K., and Hertzberg, J. (2007). Cached k-d Tree Search for ICP Algorithms. In *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pages 419–426.

Oleynikova, H., Burri, M., Lynen, S., and Siegwart, R. (2015). Real-time visual-inertial localization for aerial and ground robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3079–3085.

Orbbec (2017). Products - Orbbec. `https://orbbec3d.com/products`. [Online; accessed 01-May-2017].

Orsag, M., Korpela, C., and Oh, P. (2013). Modeling and Control of MM-UAV: Mobile Manipulating Unmanned Aerial Vehicle. *Journal of Intelligent & Robotic Systems*, 69(1):227–240.

Paz, L. M., Piniés, P., Tardós, J. D., and Neira, J. (2008). Large-scale 6-DOF SLAM with stereo-in-hand. *IEEE transactions on robotics*, 24(5):946–957.

Pedersen, H. K. and Cooke, N. J. (2006). From battle plans to football plays: Extending military team cognition to football. *International Journal of Sport and Exercise Psychology*, 4(4):422–446.

Pinies, P., Lupton, T., Sukkarieh, S., and Tardos, J. D. (2007). Inertial Aiding of Inverse Depth SLAM using a Monocular Camera. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2797–2802.

Pire, T., Fischer, T., Civera, J., De Cristóforis, P., and Berlles, J. J. (2015). Stereo parallel tracking and mapping for robot localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378.

Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. (2013). Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148.

Puri, A. (2005). A survey of unmanned aerial vehicles (UAV) for traffic surveillance. *Department of computer science and engineering, University of South Florida*.

Purvis, K. B., Astrom, K. J., and Khammash, M. (2008). Estimation and optimal configurations for localization using cooperative uavs. *IEEE Transactions on Control Systems Technology*, 16(5):947–958.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5.

Raja, A. K. and Pang, Z. (2016). High accuracy indoor localization for robot-based fine-grain inspection of smart buildings. In *IEEE International Conference on Industrial Technology (ICIT)*, pages 2010–2015.

Reid, W., Goktogan, A. H., and Sukkarieh, S. (2014). Moving MAMMOTH: Stable Motion for a Reconfigurable Wheel-on-Leg Rover. In *Proceedings of Australasian Conference on Robotics and Automation*, pages 1–10, Melbourne.

Rekleitis, I., Meger, D., and Dudek, G. (2006). Simultaneous planning, localization, and mapping in a camera sensor network. *Robotics and Autonomous Systems*, 54(11):921–932.

Revercomb, H. E., Smith, W. L., Best, F. A., Giroux, J., LaPorte, D. D., Knuteson, R. O., Werner, M. W., Anderson, J. R., Ciganovich, N., Cline, R. W., et al. (1996). Airborne and ground-based Fourier transform spectrometers for meteorology: HIS, AERI, and the new AERI-UAV. In *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, pages 106–117. International Society for Optics and Photonics.

Rosten, E. and Drummond, T. (2006). Machine Learning for High Speed Corner Detection. In *Proceedings of the 9th European Conference on Computer Vision (ECCV) - Volume Part 1*, pages 430–443, Berlin.

Royer, E., Lhuillier, M., Dhome, M., and Lavest, J.-M. (2007). Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571.

Ruiz, J., Diaz-Mas, L., Perez, F., and Viguria, A. (2013). Evaluating the accuracy of DEM generation algorithms from UAV imagery. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40:333–337.

Rullan-Lara, J.-L., Salazar, S., and Lozano, R. (2011). Uav real-time location using a wireless sensor network. In *8th Workshop on Positioning Navigation and Communication (WPNC)*, pages 18–23. IEEE.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the icp algorithm. In *Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE.

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217.

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

Saari, H., Pellikka, I., Pesonen, L., Tuominen, S., Heikkilä, J., Holmlund, C., Mäkynen, J., Ojala, K., and Antila, T. (2011). Unmanned Aerial Vehicle (UAV) operated spectral camera system for forest and agriculture applications. In *SPIE Remote Sensing*, pages 81740H–81740H. International Society for Optics and Photonics.

Salvi, J., Matabosch, C., Fofi, D., and Forest, J. (2007). A review of recent range image registration methods with accuracy evaluation. *Image and Vision computing*, 25(5):578–596.

Sansoni, G., Trebeschi, M., and Docchio, F. (2009). State-of-the-art and applications of 3D imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, 9(1):568–601.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92.

Scharstein, D., Szeliski, R., and Zabih, R. (2001). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV)*, pages 131–140.

Schmid, K., Tomic, T., Ruess, F., Hirschmüller, H., and Suppa, M. (2013). Stereo vision based indoor/outdoor navigation for flying robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3955–3962.

Schneith (2014). Calibrating the VI Sensor. `https://github.com/ethz-asl/kalibr/wiki/calibrating-the-vi-sensor`. [Online; accessed 01-May-2017].

Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-ICP. *Robotics: science and systems*, 2(4).

Shepard, D. P., Bhatti, J. A., Humphreys, T. E., and Fansler, A. A. (2012). Evaluation of smart grid and civilian uav vulnerability to gps spoofing attacks. In *Proceedings of the ION GNSS Meeting*, volume 3.

Siebert, S. and Teizer, J. (2014). Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system. *Automation in Construction*, 41:1–14.

Sünderhauf, N. and Protzel, P. (2011). BRIEF-Gist - closing the loop by simple means. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1234–1241.

Thrun, S. (2011). *Sebastian Thrun: Google's driverless car*. TED.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT Press.

Thrun, S., Fox, D., Burgard, W., and Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128(1-2):99–141.

Tiemann, J., Schweikowski, F., and Wietfeld, C. (2015). Design of an UWB indoor-positioning system for UAV navigation in GNSS-denied environments. In *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–7. IEEE.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. In *International workshop on vision algorithms*, pages 298–372. Springer.

Wang, B., Chen, X., Wang, Q., Liu, L., Zhang, H., and Li, B. (2010). Power line inspection with a flying robot. In *1st International Conference on Applied Robotics for the Power Industry (CARPI)*, pages 1–6. IEEE.

Wang, Z. M., Miao, D. H., and Du, Z. J. (2009). Simultaneous localization and mapping for mobile robot based on an improved particle filter algorithm. In *International Conference on Mechatronics and Automation (ICMA)*, pages 1106 –1110.

Weiss, S., Scaramuzza, D., and Siegwart, R. (2011). Monocular-SLAM-Based Navigation for Autonomous Micro Helicopters in GPS-denied Environments. *Journal of Field Robotics*, 28(6):854–874.

Xu, Z., Schwarte, R., Heinol, H.-G., Buxbaum, B., and Ringbeck, T. (1998). Smart pixel: photonic mixer device (PMD). In *International Conference on Mechatronics & Machine Vision*, pages 259–264.

Yang, P. (2012). Efficient particle filter algorithm for ultrasonic sensor-based 2D range-only simultaneous localisation and mapping application. *IET Wireless Sensor Systems*, 2(4):394 –401.

Zhan, P., Yu, K., and Swindlehurst, A. L. (2011). Wireless relay communications with unmanned aerial vehicles: Performance and optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 47(3):2068–2085.