UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II
SCUOLA DI DOTTORATO IN
INGEGNERIA INFORMATICA ed AUTOMATICA
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE
DELL'INFORMAZIONE

The Role of Vision Algorithms for Micro Aerial Vehicles

Giuseppe Loianno

giuseppe.loianno@unina.it

In Partial Fulfillment of the Requirements for the Degree of
PHILOSOPHIAE DOCTOR in
Computer Science and Automation Engineering

March 2014

TUTOR
Dr. Vincenzo Lippiello
Prof. Vijay Kumar

COORDINATOR
Prof. Francesco Garofalo

UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

SCUOLA DI DOTTORATO IN
INGEGNERIA INFORMATICA ED AUTOMATICA

Date: **March 2014**

Author: **Giuseppe Loianno**

Title: **The Role of Vision Algorithms for Micro Aerial Vehicles**

Department: **DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE**

Degree: **PHILOSOPHIAE DOCTOR**

.... to my family

"Learning never exhausts the mind." - Leonardo Da Vinci

Acknowledgments

This work is the fruit of three years of work in the field of Micro Aerial Vehicles. The first acknowledgment is for my family, who always supported me, even in the period where I didn't believe I could accomplish this working experience.

I have to thank my first mentor Prof. Dr. Bruno Siciliano, who introduced me to robotics and gave me the possibility to spend part of my research outside my country. He believed in my skills and he always encouraged me to give the best. A special thank is given to my supervisor Dr. Vincenzo Lippiello, who shared his expertise in the field and he helped me to organize the first part of my research during the AIRobots project. During this experience, I met many people from different countries; this gave me the working opportunity to learn new concepts and share ideas.

Finally, I'm very grateful to Prof. Dr. Vijay Kumar. He gave me many precious suggestions from his huge experience in the robotics field. He was able to host me in his laboratory for 1 year. I found there a really nice environment to cooperate and develop my skills. He encouraged me and gave me the right energy to produce always the best i could every single day. I have to thank all my friends and especially those of the GRASP Lab who were able to make me feel home and with whom, I had lots of fruitful discussions. *Thank you*!

The research leading to these results has been partially supported by the AIRobots collaborative project, which has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement ICT-248669. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

Philadelphia, February 2014

Giuseppe Loianno

Introduction

The term *robot* derives from the term *robota* which means executive labour in Slav languages. As well, *robotics* is commonly defined as the science studying the *intelligent connection between perception and action* [104]. The first definition of *robot* was established by Asimov. He was inspired by science fiction and it was defined as the science which is based on three fundamental laws

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey the orders given by human beings, except when such orders would conflict with the first law.
- A robot must protect its own existence, as long as such protection does not conflict with the first and the second law.

Human have always tried to build new machines to help themselves in the execution of several tasks, and then to completely replace themselves, especially in the most dangerous works. Moreover, the idea to have machines just able to solve simple human tasks was a limitation. Thus, the idea to have machines with a high degree of autonomy and able to make decisions matured. Nowadays, robots are widely used in industrial applications for such works where there could be more risk for human life, more cost per hour and more stress for his body. The connotation of a robot for industrial applications is that of operating in a *structured environment* whose geometrical characteristics are mostly known a priori. Past Robotic research has been dominated by the use of industrial robots. However, the world's complexity and different applications require a high degree of autonomy to solve *advanced robotics* tasks. The use of new robots, able to operate in *unstructured environments* – where the geometrical characteristics are not known a priori –, even with or in cooperation with humans and move between different locations, is needed. Recent years have seen a huge development of the aerial robotics field. Flying vehicles such as quadrotors, could solve tasks like helping humans in dangerous activities such as inspection, rescue and mapping. Compared to industrial robots, these vehicles, have a limited payload and sensing system setup. Generally, they carry a camera, an IMU and a GPS, which is not available in indoor

environments. Thus, IMU and vision measurements can be combined producing algorithms to solve in a robust way localization, navigation and mapping problems. Then, aerial robots play an important role and have already become really popular for their applicability in different domains like autonomous navigation, 3D environment reconstruction and interaction.

AIRobots

The goal of the AIRobots project is to develop a new generation of aerial service robots capable to support human beings in all those activities which require the ability to interact actively and safely with environments not constrained on ground but, indeed, freely in air, improving the autonomy of aerial robots in *unstructured environments*. The goal is to overcome the "classical" field of aerial robotics by realizing aerial vehicles able to accomplish a large variety of applications, such as inspection of buildings and large infrastructures, sample picking, aerial remote manipulation.

The starting point is an aerial platform whose mechanical configuration allows the vehicle to interact with the environment in a non-destructive way and to hover close to operating points. Rotary-wing aerial vehicles with shrouded propellers represent the basic airframes which are then equipped with appropriate robotic end-effectors and sensors in order to transform the aerial platform into an aerial service robot, a system able to fly and to achieve robotic tasks.

Advanced automatic control algorithms govern the aerial platform which are remotely supervised by the operator with the use of haptic devices. Particular emphasis is given to develop advanced human-in-the-loop and autonomous navigation control strategies relying upon a cooperative and adaptive interaction between the on-board automatic control and the remote operator. Force and visual feedback strategies are investigated too in order to transform the aerial platform in a "flying hand" suitable for aerial manipulation.

The key aspects of the project are

- Aerial service robotics best practice and performance measures. The first goal is to define a series of performance measures both for general aerial service robotic applications and for the robotic inspections scenarios of interest for the end-user. In this respect the system has to be designed to be robust, flexible, adaptable, portable, safe, intelligent, effective and economic in achieving the desired operations.

- System design and control strategies for aerial robots physically interacting with the human world. The service robotics explicitly requires the ability to interact with the environment in terms of contact between the aircraft and objects, e.g. docking and un-docking operations required to put sensors in

contact with the object to be inspected, takeoff and landing. This feature requires the design of innovative robust control strategies.

- New contribution to human-robot interaction and communication developing an advanced human-robot interface for the purpose of endowing the system with advanced action capabilities. Ideally the aerial service robot represents a "flying hand" that allows the human to act as if he/she were directly on the site, allowing a level of interaction between the human and the environment that has never been reached before in the field of aerial robotics.

- Aerial navigation in loosely structured and cluttered environments. During the inspection of the desired infrastructure the robot is required to fly in an environment which is uncertain and only partially structured because, usually, no reliable layouts and drawings of the surroundings are available. To support these features, advanced cognitive capabilities are required, and in particular the role played by vision is of paramount importance.

Outline

This work investigates the research topics included in the last key aspect. The use of vision and other onboard sensors such as IMU and GPS play a fundamental to provide high level degree of autonomy to flying vehicles. In detail, the outline of this thesis is organized as follows

- Chapter 1 is a general introduction of the aerial robotic field, the quadrotor platform, the use of onboard sensors like cameras and IMU for autonomous navigation. A discussion about camera modeling, current state of art on vision based control, navigation, environment reconstruction and sensor fusion is presented.

- Chapter 2 presents vision based control algorithms useful for reactive control like collision avoidance, perching and grasping tasks. Two main contributions are presented based on relative depth map and image based visual servoing respectively.

- Chapter 3 discusses the use of vision algorithms for localization and mapping. Compared to the previous chapter, the vision algorithm is more complex involving vehicle's poses estimation and environment reconstruction. An algorithm based on RGB-D sensors for localization, extendable to localization of multiple vehicles, is presented. Moreover, an environment representation for planning purposes, applied to industrial environments, is introduced.

- Chapter 4 introduces the possibility to combine vision measurements and IMU to estimate the motion of the vehicle. A new contribution based on

Pareto Optimization, which overcome classical Kalman filtering techniques, is presented.

- Chapter 5 contains conclusion, remarks and proposals for possible developments.

A collection of all videos related to this thesis work is available online [1].

[1] wpage.unina.it/giuseppe.loianno/videos/phd_thesis_videos.

Contents

List of Figures

List of Tables

Glossary

ASV Aerial Service Vehicle. 11, 55

IMU Inertial Measurement Unit. xi–xiii, xix, xx, 2, 7–12, 14, 16–19, 23, 33, 34, 36, 60, 69–72, 79, 83, 85

MAV Micro Aerial Vehicle. 1–4, 7, 12, 20

UAV Unmanned Aerial Vehicle. xix, 1, 2, 9, 11

Chapter 1

State of the Art

1.1 Aerial Robots: The Quadrotor Platform

UAVs have gained enormous commercial potential especially due to recent research progresses. Recent developments in term of electronic components have contributed to high density power storage, integrated miniature actuators and MEMS4 technology sensors, giving the possibility to realize small autonomous vehicles in both military and civilian applications. Military applications currently represent the most important part of the unmanned flying vehicle market, and this industrial sector is growing strongly. A good classification of the different available platforms based on flying principle and propulsion mode is provided in [15] and summarized in fig.(1.1). In the motorized heavier-than-air category, a new generation of MAV5 (Micro Aerial Vehicle) with a wingspan less than 15 cm and less than 100 grams in mass has emerged. Most of these vehicles include stabilization sensors and small cameras. The Black Widow6 MAV is a 15 cm span, fixed-wing aircraft with an embedded color camera flying at 48 km/h for around 30 minutes, and a maximum communication range of 2 km. In the same category, birdlInsect-like MAVs seem to be the perfect solution for fast navigation in narrow spaces and perhaps the best approach to miniaturization. The class of VTOL (Vertical Take Off and Landing) systems have specific characteristics which allow the execution of task difficult to accomplish with other flying models as mentioned in [15]. The main advantage is the ability to vertical, stationary and low speed flight. This class of vehicles with different configurations probably represent currently the most promising flying concept seen in terms of miniaturization. The quadrotor configuration is the most interesting and used solution, in this class of vehicles. Main disadvantages are space and energy requirements. However this vehicle concept offers a better payload and it is easy to build and to control. Moreover, the costs have contributed to locate this platform in the consumer-grade range technology. It is made by four identical propellers located at vertices of a square.

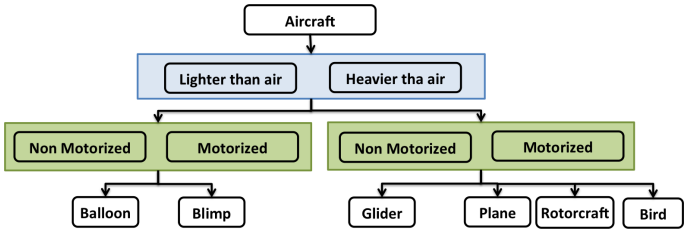


Figure 1.1: UAV classification.

Each propeller j is able to generate a force f_j along its main axis. Considering an inertial reference frame and a frame centered in the center of mass of the vehicle, the quadrotor model is described in the inertial reference frame, as the position and orientation of the body frame respect to the inertial one. The motion model of the MAV according to [113] is

$$\begin{aligned} m\ddot{\boldsymbol{x}} &= -R\tau e_3 + mge_3, \\ \dot{R} &= R\hat{\Omega}, \\ J\dot{\Omega}_i + \Omega \times J\Omega &= M, \end{aligned} \tag{1.1}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ is the Cartesian position of the vehicle expressed in the inertial frame, $m \in \mathbb{R}$ is the mass, $\Omega \in \mathbb{R}^3$ is the angular velocity in the body-fixed frame and $J \in \mathbb{R}^3$ is the inertia matrix with respect to the body frame. The hat symbol $\hat{}$ is defined by the condition $\hat{x}y = x \times y$ for all $x, y \in \mathbb{R}^3$, g is the gravity acceleration and $e_3 = [0\ 0\ 1]^T$. The total moment $M_i \in \mathbb{R}^3$ along all axes of the body-fixed frame and the thrust $\tau \in \mathbb{R}$ are control inputs of the plant. The total thrust, $\tau = \sum_{j=1}^{4} f_j$, acts in the direction of the z axis of the body-fixed frame, which is orthogonal to the plane defined by the centers of the four propellers.

1.2 Vision for Aerial Robotics

The recent years have seen a growing interest on MAVs applications in several environments. The capabilities of Micro Aerial Vehicles (MAVs) are rapidly expanding to include surveillance [84], construction [66], manipulation of slung loads [107], collaborative transportation [85, 106], and mapping of unknown environments using aerodynamic effects [25]. For indoor autonomous navigation the obstacle avoidance is one of the most relevant drawback, due to unavailability of the GPS signal and of a detailed environment map. A number of control strategies have been developed based on other on-board sensors like cameras, radar, lasers, sonars and IMU. However, the most promising approaches make use of visual sensors. For environment interaction and manipulation, Aerial vehicles' flight duration are

limited by the energy density of batteries and the speed of aerial manipulation is restricted to quasi-static interactions with the environment. An aerial vehicle endowed with capabilities traditionally ascribed to raptors, such as perching and dynamic grasping, would be instrumental towards mitigating the energy-density restriction and speeding-up interaction with the environment. In this context, a vision sensor, due to its limited payload, can be a useful device to detect object, control robot motion and speed up interaction with the environment.

1.2.1 Camera Sensor

The task of the camera as a vision sensor is to measure the intensity of light reflected by an object. To this end, a photosite element named pixel is employed to transform light energy in electric energy. Different types of sensors are available based on the principle exploited to realize the energy transformation. The most used are CCD (Charge Coupled Device) and CMOS (Complementary Metal Oxide Semiconductor) sensors based on photoelectric effect of semiconductors. A CCD sensor consists of a rectangular array of photosites. Due to photoelectric effect, when a photon hits the semiconductor surface, a number of free electrons are created, so that each element accumulates a charge depending on the time integral of the incident illumination over the photosensitive element. The charge is then passed to the output amplifier and the element discharged. A CMOS sensor consists of a rectangular array of photodiodes. The junction of each photodiode is precharged and it is discharged when hits by photons. An amplifier integrated in each pixel can transform this charge into a voltage. The main different between the two sensors is that CMOS, respect to CCD sensors, are non integrating devices, measuring the throughput and not the volume. In this way the influence of neighboring pixels is prevented, avoiding blooming which is a typical problem with CCD sensors. Several reasons motivating the use of vision sensors in the MAV field

- They are typically inexpensive sensors
- They are useful like human eyes to obtain an idea of the environment structure while flying
- The frame rate of vision algorithm is generally compatible for control purposes
- Vision algorithms are reliable and precise
- Camera is a lightweight sensor so it can easily be mounted onboard the vehicles
- Camera sensor can be used for different scopes

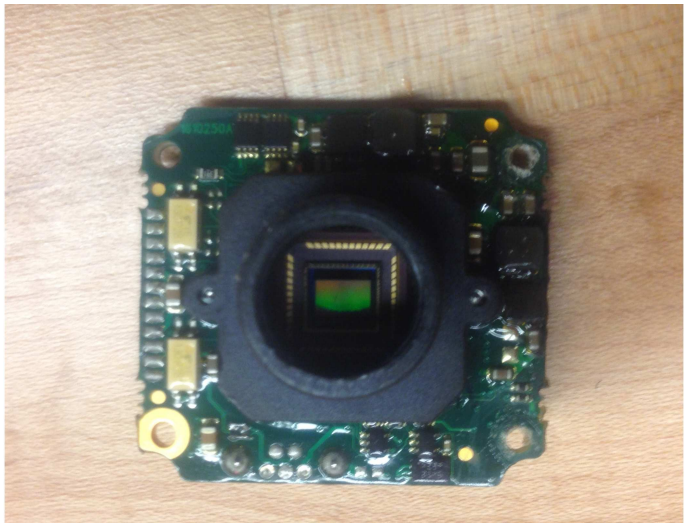
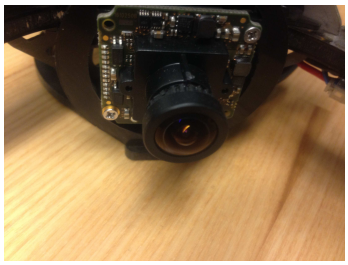


Figure 1.2: Camera sensor



(a) Classical lens.

(b) Fisheye lens.

Figure 1.3: Two different type of lenses.

Figure 1.2 shows a typical camera, while 1.3 shows two camera lenses, with different fields of view. The lens is generally responsible to direct the incoming light controlling the direction of propagation. Generally this is obtained by diffraction, refraction and reflection. In this work different camera are applied to the MAV domain. In particular in the first part the attention focus on the use of monocular system for reactive control, while in the second part RGB-D sensors and stereo camera configurations are used with the purpose to obtain a high level environment map.

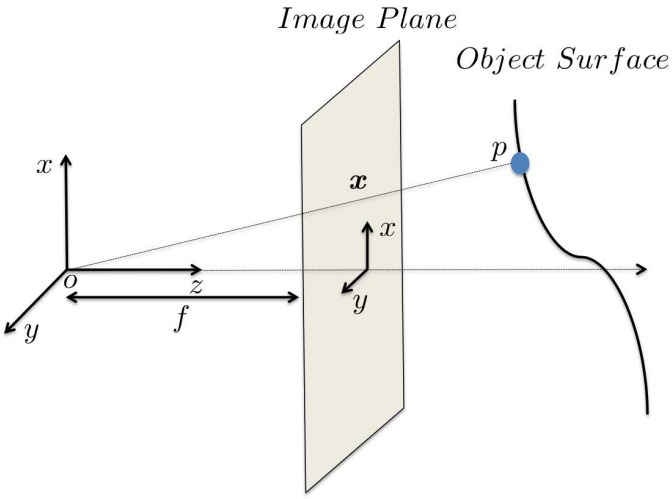


Figure 1.4: Pinhole camera model.

1.2.2 Camera Model

In this part, an overview of the vision system and the adopted camera model, is presented. The following nomenclature will be used. Let $T \in SE(3)$ be the homogeneous transformation matrix from the camera frame to the world frame, f denote a focal length, s_α indicates the pixel coordinate transformation in the α direction, c_α be the center image pixel in the α direction, and λ be an arbitrary scaling factor. In the presented work, the camera is modeled using a standard pinhole perspective camera model as shown in 1.4 so that a generic point in the world, $p = [X,\ Y,\ Z,\ 1]^T$, is projected onto the image plane, $[x',\ y',\ 1]^T$, according to [75] such that

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K P_0 T^{-1} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad K = \begin{bmatrix} f s_x & 0 & c_x \\ 0 & f s_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad P_0 = \begin{bmatrix} \mathcal{I}_{3\times 3} & \mathbf{0}_{3\times 1} \end{bmatrix}. \tag{1.2}$$

The calibrated image coordinates are defined as,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}, \tag{1.3}$$

which are equivalent to the transformation and projection of points in the world to an image plane with unity focal length and a centered image coordinate system. Other camera models exist, more suitable for other cameras typology like omnidi-

rectional cameras [98]. However in this context only the perspective camera model has been presented due to its general use and simplicity.

1.2.3 Vision for Reactive Control

One of the primary objectives in aerial robotics, is the possibility, when the robot flies from an initial position to a final one, to avoid possible obstacles during the path. Several methods based on visual collision avoidance have been proposed. When a stereo camera system is available, an image couple can be employed as in [55] to compute distances towards detected objects based on triangulation. However, stereo systems require a high payload and onboard computational capacity. Several biologically inspired approach have also been presented. In [114] it is shown that fruit flies avoid obstacles when they turn away from the region with a high level of Optical Flow (OF). On the other hand, in [109] it is found out that honeybees try balancing the amount of lateral OF in order to stay equidistant from the flanking walls.

Different studies in the last years have concerned with the use of Optical Flow for obstacle avoidance. In some approaches the average intensity of the left and right OF vectors is balanced, according to the fact that if the left optical flow is larger than the right one, it means that the object is closer to the left side than the right one, and viceversa. A nonlinear control strategy for obstacle avoidance based on the OF is presented in [60], while autopilots for lateral obstacle avoidance of an hovercraft using two one-dimensional sensors pointing at $\pm 90^{\circ}$ have been developed in [109] and [100]. A single-camera frontal collision-avoidance strategy computing the divergence of the OF is proposed in [128], where an increase of the OF divergence indicate the presence of a frontal obstacle.

The optical flow has also been used for implementing altitude control for MAVs, e.g regulating the altitude of a helicopter using two downward optical flow sensors as in [96]. In this last, constant speed obtained by a constant pitch angle implies that the amount of OF is constant so that the vehicle stays at a constant height above ground [128, 48].

In [99] two different strategies, with and without the adoption of the OF, based on the *Time to Contact* –time needed to obtain a collision between the obstacle and the vehicle, while it is moving with a translational speed– have been proposed.

The *Depth Map* (DM) of the environment can be computed using the OF and GPS measurements. In [27, 26] an intuitive 3D map providing obstacle locations is provided using only OF and GPS data. A lateral obstacle avoidance algorithm for a wheeled robot has been proposed in [124], where a depth map obtained from the OF evaluated with an omnidirectional camera has been used. In [82] a real-time algorithm to compute the *Relative Depth Map* (RDM) from the OF independently of the performed motion, while in [126] the RDP is employed for the navigation through indoor corridors in the case of linear motion.

The challenge in the MAV field, is not only avoid obstacles, but give the vehicle the capability to interact with the environment enabling autonomous grasping or perching. Acquiring, transporting and deploying payloads while maintaining a significant velocity are important since they would save MAVs time and energy by minimizing required flight time. Nature provides many examples of energy and time efficient creatures that provide inspiration for the field of robotics. Raptors are excellent aerial hunters and are able to conserve energy by perching on a wide variety of objects while colugos use their ability to glide in order to save time [19]. Further, perching can conserve energy that would have otherwise been expended hovering. Beyond energy-efficiency, high-speed grasping would be particularly useful if a MAV was needed to quickly acquire or deploy sensors, materials, or robots. The robot must be able to detect the object of interest and use visual feedback to control the robot's motion. However, to maintain agility, the robot must have low inertia (*i.e.* minimal sensor payload) and consider the dynamics of the system. Another limitation is the poor understanding of the perception-action loops required for agile flight and manipulation. One can observe that visual feedback is used to close the control loop while dynamically grasping prey (see Figure 1.5). In scenarios like this, a monocular camera is an ideal sensor, especially when combined with an IMU [88, 122], and motivates either Position Based Visual Servoing (PBVS) or Image Based Visual Servoing (IBVS) [57]. PBVS requires an explicit estimation of the pose of the robot in the inertial frame while IBVS acts directly using feedback from the image coordinates. In particular, a single monocular camera is sufficient for visual servoing when there is some known geometry or structure in the environment.

It is natural to look to nature for inspiration when approaching such design challenges. From video footage it is clear that raptors sweep their legs and claws backwards while capturing prey, thereby reducing the relative velocity between the claws and the prey [13]. This allows the bird, without slowing down, to have a near-zero relative velocity between the claw and the prey. This method can inspire how to enable high-speed aerial grasping and manipulation for MAVs.

There are many excellent tutorials on visual servoing [40, 57, 29, 30]; however, most approaches assume first-order or fully-actuated systems. For example, [115] demonstrated robustness to camera calibration, but only considered a first-order system. Stability was proven for second order systems, but assumed full actuation [34]. More recently, [52] and [51] leveraged a spherical camera model and utilized backstepping to design non-linear controllers for a specific class of underactuated second-order systems. As is typical in backstepping, however, it is necessary to assume that the inner control loops are significantly faster than the outer ones. There have been some preliminary efforts towards autonomous landing, but an estimate of velocity in the inertial frame is obtained using an external motion capture system [64]. Thus, there is a lack of IBVS controllers which can handle the dynamic motion required for aggressive grasping and perching.

Figure 1.5: A Red Kite swoops down and uses visual feedback to approach, grasp, and retrieve food on the ground [116] (top). A bald eagle uses a similar strategy to hunt prey in the water [13] (bottom).

Therefore, the major goal is to ascribe to aerial vehicles the ability to autonomously and dynamically fly above, grasp, or perch on a target. In particular, a quadrotor platform is considered, which is appealing, as mentioned, due to its mechanical simplicity, its agility, its ability to hover, and its well-understood dynamics [107]. The system is underactuated; however, it is possible to design controllers that guarantee convergence from almost any point on $SE(3)$, the Euclidean motion group in three dimensions [65]. Similar controllers have been derived for a quadrotor carrying a cable-suspended payload [107]. However, both of these approaches require full knowledge of the state. In order to achieve these goals, the dynamics of the system directly will be expressed in the image plane (rather than in the Cartesian space) to develop an IBVS controller based on visual features of a cylinder [117, 118].

1.2.4 Vision for Pose Estimation and 3D Reconstruction

In previously mentioned works, the visual feedback is used to obtain a reactive control in term of obstacle avoidance and environment interaction. However, vision sensors can are employed to map the surrounding environment. Eventually, the information they provide can fused with the IMU to obtain autonomous control. In this context, vision algorithms are more complex, they are addressed for high level tasks like environment reconstruction and planning. They give a more detailed environment representation, but a less reactive behavior, needed in the previous

described domains. The attention, in this work, is mainly focused on the use of low-cost range sensors for single and multiple platform localization tasks.

These sensors are an attractive alternative to expensive laser scanners or 3D cameras for applications such as indoor navigation and mapping, surveillance, and autonomous robotics. Consumer-grade range sensing technology has led to many devices becoming available on the market like Microsoft Kinect sensor and the ASUS Xtion sensor (PrimeSense, 2010 see Fig. 1.6). The richness of the provided data and the low cost of the sensor have attracted many researchers from the fields of mapping, 3D modeling, and reconstruction. The ASUS Xtion sensor[1] boasts a lower weight than the first generation of RGB-D cameras (around 70g without the external casing), it does not need external power other than the USB connection, and it is very compact. These properties give this device some unique characteristics suitable, for example, for environment mapping and monitoring with UAVs.

There are a number of Simultaneous Localization and Mapping (SLAM) approaches for Micro Aerial Vehicles (MAVs). Good results have been obtained using monocular cameras and IMUs [62], stereo camera configurations [121] and RGB-D sensor systems [102, 101, 53, 39, 11, 91].

In [102, 101], a Kinect and on-board vehicle sensors are used to perform state estimation through Kalman filtering, while in [62], the same filter is used to combine monocular visual information with inertial sensor data solving the scale factor problem. All of these approaches show the feasibility of 3-D SLAM on a computationally-constrained aerial platform. In [53] an RGB-D 3D mapping system utilizes a novel joint optimization algorithm combining visual features and shape-based alignment. In [11] a direct 3D tracking approach is proposed such that an error is based directly on the intensity of pixels. Other algorithms are based on fusing depth maps to a coherent 3D model [91]

Most previous approaches are slow and designed for use on a single platform. However, in some cases, to map large environments and to help the vehicle which is already mapping the environment at that time, it may be necessary to deploy new vehicles which can cooperate in the mapping task. If multiple MAVs can collaborate in mapping tasks, they can cover the same environment in a faster and more reliable way compared to a single vehicle. Moreover, the fused map information can be exploited by every vehicle in order to make decisions on steps and task allocations.

In collaborative mapping, different vehicles can be launched from different initial positions and orientations. In general, any kind of prior knowledge can be considered on the relative pose of vehicles. Thus, the relative pose information should be inferred from different measurements (that may or may not be independent) of the external scene by different vehicles. This problem is addressed in

[1]While this specific sensor is no longer available, there are others under development that are likely to be available in the future.

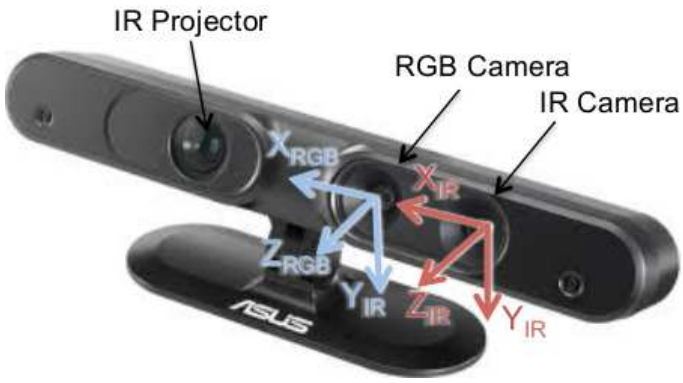


Figure 1.6: Asus Xtion Pro Live Sensor.

[42] where two MAVs map the same environment. Loop closure detection between different cameras, map merging, and concurrent map access are performed on a ground station where maps are replicated. The advantage of this approach is that each vehicle starts its own map. However, the ground station is responsible for map merging and must manage redundant information like the maps built by the individual vehicles, including the overlap in the mapping data. Further, because the maps are based only on monocular vision, the process is not likely to be robust to sudden changes in depth.

There is also work using range sensors with ground robots in 2-D environments [43, 94, 54, 120]. The use of bearing measurements from monocular cameras coupled with IMU has been exploited in [32]. An interesting approach is presented in [127] where 3 synchronous cameras are able to perform localization even in dynamic environments. However, the images are required to be synchronized. Therefore, this approach is difficult to apply in the field of MAVs. In [105], a stochastic approach is presented for cooperative mapping with a Kalman Filter fusing different camera poses and observed landmarks. In chapter 3 a new solution combining monocular SLAM and depth data [73] enabling the localization at 30 Hz is proposed. Moreover this solution is extended to a partially distributed architecture for cooperative localization and mapping, avoiding the map merging problem.

However, the main problem involving general environment reconstruction is the sparsity of the point cloud data representing the environment. This environment representation is unsuitable if it has to be managed by a high-level supervisory control. The world is simply modeled as a set of 3D point clouds presenting the disadvantage of outliers which can affect planning performances. Moreover the sparsity of the representation, due to computational algorithms, would require unacceptable planning time. Thus, a different environment representation strategy

has to be chosen in case of a high-level control system designed for an ASV operating in close interaction with the external environment, like the one presented for the AIRobots project [2, 79, 81] where unmanned service helicopters are equipped with sensors and end-effectors, and capable not only to fly, but also to achieve robotic tasks in proximity and in contact with the surface (e.g. site inspections, simple manipulations, probe testing, etc.).

This application domain is challenging and novel and has not been investigated in depth in the UAV literature, which is mainly focused on free flight tasks an simultaneous localization, mapping, and path planning problems [14, 55, 110, 67, 70, 68]. High-level architectures for UAVs have been proposed in literature [36, 46, 37], but none of these addresses the challenges of the ASV domain proposed in this paper. The aim in this work, is mainly to present an onverview of the architecture adopted in this novel scenario along with the vision based solutions adopted for high level tasks in the AIRobots project, which requirements have been widely discussed in different works [79, 80, 21, 23]. A discretization of the vehicle's workspace with elementary cubes is proposed, in the chapter 3, to model the environment for the proposed high level control architecture.

1.2.5 Sensor Fusion

As previously mentioned, information provided by a vision algorithm can be fused with different sensors like lasers scanners and IMU. In general, the goal is to increase robustness and speed of previous visual localization algorithms. The use of filters combining different sensor data, which are generally provided at different sampling rates, is highly appealing.

Different methods have been studied to combine heterogeneous information sources such as Global Navigation System (GNS), inertial navigation systems, odometry and local radio technologies [49, 93]. Nevertheless, this remains an open research field in robotics and especially in Micro Aerial Vehicles (MAVs) applications, where low cost IMUs have to be combined with one or more cameras information, as well as with the Global Positioning System (GPS) available data. Due to unbounded accumulation of integration errors, position and velocity can be only estimated for no more than few seconds by using only IMU data. On the other side, vision sensors are able to provide positional information with no drift with respect to fixed observed environments. However, the main drawback of this sensors is the huge amount of data to be transmitted and/or elaborated on-line to extract positional information, that generates at least a time delay in the estimation update and a low measurement rate (e.g. 10 Hz) compared to IMU sensors (e.g. 100-200 Hz).

Unequal periodicity of the sampling times of the measurement devices rises significant challenges. A known solution consists in adopting multi-rate filters [10]. Further, the delay that characterizes visual measurements can be addressed by

adopting different techniques based on Kalman filters or its variants [71, 63]. Specifically, in this latter work the authors extrapolate the delayed measurement forward into the present time, and calculates an optimal gain. However, the delay compensation is often achieved by a state augmentation depending on the given delay [56]. In [87] the general Kalman filter formulation is extended by considering both the relative measurements update and the correlation between two consecutive displacements, while a solution to choose the initial state covariance matrix is addressed in [62]. The solution proposed in [10] has been employed in [17] by compensating the delay due to the wireless data communication and image processing to stabilize a MAV with a standard PID controller.

The adoption of a mono-camera system in an unknown environment determines the capability to estimate the egomotion of the vehicle up to a scale factor. By using a sensor fusion techniques combining inertial and visual data, the global scale factor can be estimated achieving an absolute egomotion estimation. The solutions proposed in [61, 67, 69], which are not based on the Kalman filter, combine inertial measurements and consecutive feature matchings to obtain a closed-form solution for scale-factor estimation.

Optimal sensor fusion techniques based on second order moment minimization [8] and Pareto Optimization [9] try to couple heterogeneous sensors such as Ultra-Wideband radio measurements with speed and absolute orientation information. Other works rely on the use of complementary filters and nonlinear estimators as in [50] and [31]. In these latter cases, the vehicle position, velocity and attitude estimation is obtained using a nonlinear dynamic system, where the proof of stability is obtained using the Lyapunov stability theory.

In this work, a new optimal sensor fusion algorithm [72] based on Pareto optimization techniques is proposed in Chapter 4 to combine IMU and camera visual measurements to estimate a vehicle motion. Results show, respect to a Kalman filter approach, an improved estimation at the price of a limited increased computational complexity.

Chapter 2

Vision for Reactive Control

In this chapter two main contributions are presented. First, a new vision-based obstacle avoidance technique [67, 68] for indoor navigation is presented for MAVs applications. The vehicle trajectory is modified according to a repulsive force field generating from the DM of the surrounding environment computed online using the OF. A single onboard omnidirectional camera is assumed to be available. In particular, a new formulation for a closed-form solution for the absolute-scale velocity estimation problem, which are required for the DM estimation, is presented. Starting from the solution proposed in [61], where in addition to inertial measurements the correspondences of an image feature between three image frames (here referred as *visual station*) are required, a new compact formulation is adopted also generalizing to the case of multiple visual station and image features. A dynamic region-of-interest for image feature extraction and a navigation velocity self-limitation control are considered to improve safety during navigation in view of the estimated vehicle velocity. Second, in the image space, a controller for the robot that relies on visual feedback from a monocular camera is proposed [117, 118]. Following this, a description of the hardware used in experiments, particularly the camera system, is provided. Experimental results, which include high-speed vision-based control, are then proposed. For visual control, IBVS and geometric-control literature is proposed, generalizing from a first-order fully actuated system to a higher-order underactuated system. Further,methods to guarantee dynamically feasible trajectory generation in the image space by utilizing the differential flatness property, are demonstrated. Finally, the proposed trajectory generation methods and control laws are verified in simulation and experimentation.

2.1 Optical Flow and Depth Map

The Optical Flow can be defined as the apparent motion of a image features (objects, surfaces, etc.) between two consecutive camera frames caused by the

relative motion between the camera and the scene. It is known that the motion of obstacles observed in an image sequence depends on the distance of the object with respect to the camera, and thus the OF can be profitably exploited estimating the distances of surrounding obstacles. For this reason, OF is often employed in non-stereo visual based obstacle avoidance. However, the estimation of the absolute distance of an obstacle requires the knowledge of the vehicle translational velocity, which is here evaluated with a new closed-form solution based on image correspondences and IMU measurements.

2.1.1 Depth map construction with Optical Flow

In the case of a purely translational motion of the vehicle, assuming that all the objects in the scene are stationary, the translational Optical Flow ω_{T} of an image feature of an observed object depends on the relative velocity between the camera and the object itself $\boldsymbol{v}$ and on the angle between the direction of motion and the observed feature α, as shown in Fig. 2.1, with the following rule:

$$d = \frac{\|\boldsymbol{v}\|}{\omega_{\mathrm{T}}} \sin(\alpha), \tag{2.1}$$

where d is the distance between the object feature and the camera. Therefore, if the vehicle velocity is available, the distance and so the position of the observed obstacle can be estimated. However, in a general case, the motion of the vehicle is composed of a translational part and of a rotational part, namely ω_{T} and ω_{R}, each of which produces a rate of the OF.

The computation of the ω_{T} component can be performed applying a compensation of the rotational effect as described in [126]. With reference to Fig. 2.2, the inertial and the camera reference frames are denoted with $I - x_I y_I z_I$ and $O - xyz$, respectively. Without loss of generality, it is supposed that the camera and the vehicle frames are coincident. The camera velocity $\boldsymbol{v}$ and acceleration $\boldsymbol{a}$, this last provided by the onboard IMU system with a period T, are expressed in camera frame. The orientation of the camera frame, also extracted using the IMU measurements, is referred to the inertial frame and expressed using the well-known Tait-Bryan (Euler) angles roll, pitch, and yaw $\boldsymbol{\phi} = (\varphi, \theta, \psi)$.

Adopting a classical pin-hole camera model (other models can be considered in view of the available hardware, e.g. see [98] for the case of fisheye lens) and assuming known the camera calibration parameters, the image feature vector $\boldsymbol{f} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$, i.e. the position of the observed feature with respect to the camera, can be expressed using the normalized image coordinates X and Y as follows

$$\boldsymbol{f} = z \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = d \cdot \hat{\boldsymbol{f}}, \tag{2.2}$$

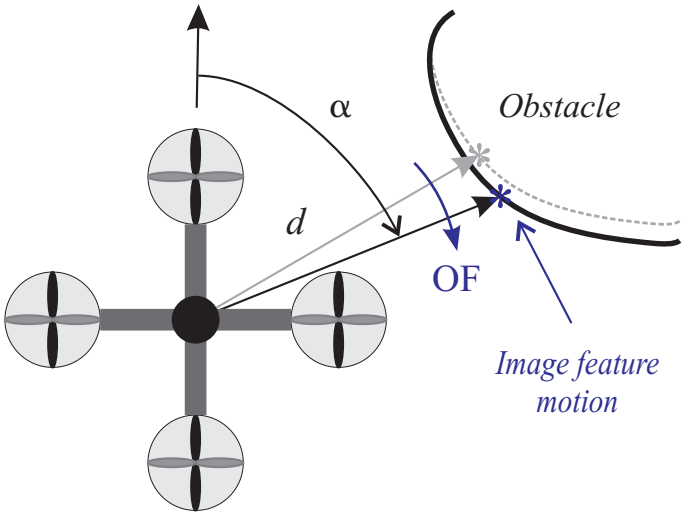


Figure 2.1: Optical flow during a translational motion.

where $d = \|\boldsymbol{f}\|$ is the distance of the feature and $\hat{\boldsymbol{f}}$ is the unit feature vector depending only on visual measurements X and Y.

The image features considered in this paper are corner extracted using the well known Shi-Tomasi corner detector, while the Pyramidal Lucas-Kanade algorithm [74, 16] has been employed to find correspondences between consecutive image frames. Denote with $\hat{\boldsymbol{f}}_1^1$ and $\hat{\boldsymbol{f}}_2^2$ the unit feature vectors of a correspondence between two consecutive images, both represented in the respective reference frames –conventionally, for vectors and matrices the reference frame is indicated as superscript– and with $\boldsymbol{\phi}_{12}$ the corresponding angular changes for the camera orientation. Then, the unit feature vector $\hat{\boldsymbol{f}}_2^1$ representing the position of the image feature measured in frame 2 reported in frame 1 can be evaluated as follows

$$\hat{\boldsymbol{f}}_2^1 = \boldsymbol{R}_2^1 \hat{\boldsymbol{f}}_2^2, \tag{2.3}$$

where $\boldsymbol{R}_2^1 = \boldsymbol{R}(\boldsymbol{\phi}_{12})$ is the rotational matrix representing the rotation performed by the camera in the form

$$\boldsymbol{R}(\boldsymbol{\phi}) = \begin{bmatrix} c_\varphi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\varphi s_\theta c_\psi + s_\phi s_\psi \\ s_\varphi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\varphi s_\theta c_\psi - c_\phi c_\psi \\ -s_\theta & -c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}.$$

The corresponding ω_{T} can be estimated as the angular velocity of the feature vector evaluated in the interval Δt_{12}, between the image frames 1 and 2, given by

$$\omega_{\mathrm{T}} = \frac{\cos^{-1}\left(\hat{\boldsymbol{f}}_1^1 \cdot \hat{\boldsymbol{f}}_2^1\right)}{\Delta t_{12}}. \tag{2.4}$$

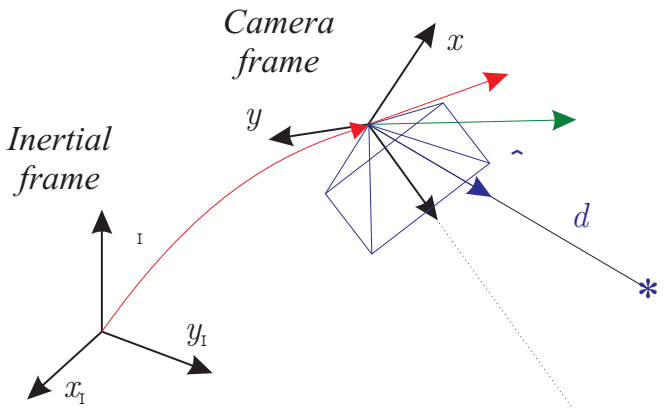


Figure 2.2: Inertial and camera reference frames.

Figure 2.3 shows the ω_T computed in a real indoor scene with an omnidirectional fisheye camera.

For a given vehicle translational velocity $\boldsymbol{v}$, substituting (2.4) in (2.1) and the result in (2.2), the set of all feature vectors $\boldsymbol{f}$ of the available image correspondences can be evaluated, constituting the instant *Depth Map* of the surrounding environment at the time of the image frame acquisition.

2.1.2 Velocity estimation

In this section a generalization of the method proposed in [61] is presented with a more compact analytical formulation, where the extension to a multi-frame multi-feature correspondence is explicitly considered. Without loss of generality, it is assumed that the period of the visual system is N times the period of the IMU system T. This means that between two consecutive images there are N available measures provided by the IMU. Moreover, it is assumed that the IMU and the camera reference frames are coincident –if both are calibrated it is easy to refer IMU data to the camera frame– and that the IMU is ideal, i.e. it provides gravity and bias-free acceleration and gyroscopic measurements. Therefore, only the camera frame will be considered in the rest of the section. Finally, the acceleration $\boldsymbol{a}$ is always expressed in the current camera frame (e.g. $\boldsymbol{a}_j = \boldsymbol{a}_j^j$, where j refers to the camera frame at the time instant t_j).

Considering a camera motion as shown in Fig. 2.4 and assuming that t_k is the last sample time with available visual data, the previous available visual measurements are referred to the sample times t_{k-sN}, with $s \in \mathbb{N}$ (s identifies each visual station). By denoting with $\boldsymbol{r}_i^j$ the relative displacement of the frame i with respect and referred to the frame j and considering a single image feature match between

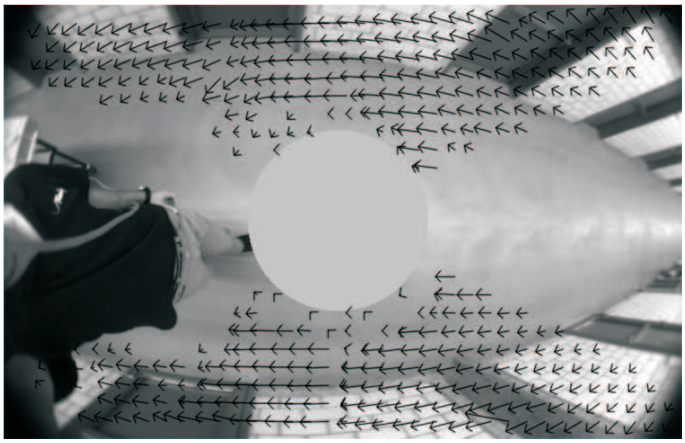


Figure 2.3: Optical Flow estimated in a real scene.

frames k and $k_s = k - sN$, the following relation can be written

$$d_{k_s}\hat{\boldsymbol{f}}_{k_s}^{k_s} = \left(\boldsymbol{R}_{k_s}^{k}\right)^{\mathrm{T}}\left(d_k\hat{\boldsymbol{f}}_k^k - \boldsymbol{r}_{k_s}^k\right), \tag{2.5}$$

where $\boldsymbol{R}_{k_s}^{k} = \left[\begin{array}{ccc}\boldsymbol{r}_x & \boldsymbol{r}_y & \boldsymbol{r}_z\end{array}\right]_{k_s}^{k}$ is the rotational matrix representing the orientation of frame k_s with respect to frame k, and $\boldsymbol{r}_x$, $\boldsymbol{r}_y$, and $\boldsymbol{r}_z$ are the its column vectors. This relative displacement can be expressed in terms of the current velocity, with respect to the current camera frame k, and the integration of acceleration samples between t_{k-sN} and t_k. Let us consider the relative displacement and velocity between two consecutive frames:

$$\boldsymbol{r}_j^{j-1} = \boldsymbol{v}_{j-1}^{j-1}T + \frac{1}{2}\boldsymbol{a}_{j-1}T^2 \tag{2.6}$$

$$\boldsymbol{v}_j^{j-1} = \boldsymbol{v}_{j-1}^{j-1} + \boldsymbol{a}_{j-1}T \tag{2.7}$$

$$\boldsymbol{r}_{j-1}^{j} = -\boldsymbol{R}_{j-1}^{j}\boldsymbol{r}_j^{j-1} = -\boldsymbol{v}_{j-1}T - \frac{1}{2}\boldsymbol{R}_{j-1}^{j}\boldsymbol{a}_{j-1}T^2 \tag{2.8}$$

$$\boldsymbol{v}_j = \boldsymbol{R}_{j-1}^{j}\boldsymbol{v}_j^{j-1} = \boldsymbol{v}_{j-1} + \boldsymbol{R}_{j-1}^{j}\boldsymbol{a}_{j-1}T. \tag{2.9}$$

Replacing (2.9) in (2.8) yelds

$$\boldsymbol{r}_{j-1}^{j} = -\boldsymbol{v}_jT + \frac{1}{2}\boldsymbol{R}_{j-1}^{j}\boldsymbol{a}_{j-1}T^2. \tag{2.10}$$

The whole displacement between two consecutive visual frames can be achieved adding all the displacements corresponding to the intermediate time intervals where only IMU data are available, obtaining

$$\boldsymbol{r}_{k_s}^{k} = -sNT\boldsymbol{v}_k + \frac{1}{2}\bar{\boldsymbol{a}}_{k_s}^{k}T^2, \tag{2.11}$$

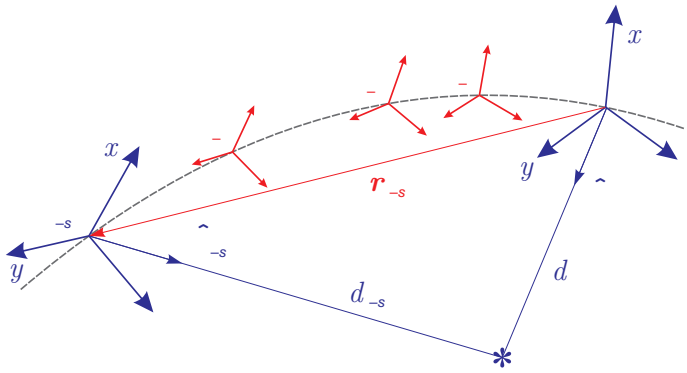


Figure 2.4: Camera (blu) and IMU measurement reference frames.

with

$$\bar{\boldsymbol{a}}_{k_s}^{k} = \sum_{j=1}^{sN} (2(sN-j)+1)\boldsymbol{R}_{k-j}^{k}\boldsymbol{a}_{k-j}, \tag{2.12}$$

which can also be expressed in a recursive formulation, here omitted for brevity.

By plugging (2.11) in (2.5) and considering (2.2), the following system of equations for a one-point image correspondence between frames k and k_s is derived

$$X_{k_s} = \frac{\left(\boldsymbol{r}_{x,k_s}^{k}\right)^{\mathrm{T}}\left(d_k\hat{\boldsymbol{f}}_k^k + sNT\boldsymbol{v}_k - \frac{1}{2}\bar{\boldsymbol{a}}_{k_s}^{k}T^2\right)}{\left(\boldsymbol{r}_{z,k_s}^{k}\right)^{\mathrm{T}}\left(d_k\hat{\boldsymbol{f}}_k^k + sNT\boldsymbol{v}_k - \frac{1}{2}\bar{\boldsymbol{a}}_{k_s}^{k}T^2\right)} \tag{2.13}$$

$$Y_{k_s} = \frac{\left(\boldsymbol{r}_{y,k_s}^{k}\right)^{\mathrm{T}}\left(d_k\hat{\boldsymbol{f}}_k^k + sNT\boldsymbol{v}_k - \frac{1}{2}\bar{\boldsymbol{a}}_{k_s}^{k}T^2\right)}{\left(\boldsymbol{r}_{z,k_s}^{k}\right)^{\mathrm{T}}\left(d_k\hat{\boldsymbol{f}}_k^k + sNT\boldsymbol{v}_k - \frac{1}{2}\bar{\boldsymbol{a}}_{k_s}^{k}T^2\right)}. \tag{2.14}$$

In the general case, by considering $n_s \geq 2$ visual stations and n_f image features, a system of $2n_sn_f$ equation with $3+n_f$ unknowns $\boldsymbol{v}_k$ and $\boldsymbol{d}_k$, where $\boldsymbol{d}_k$ is the n_f vector of distances of each image feature, is achieved. This linear system can be easily arranged in the classical form

$$\boldsymbol{A}\begin{bmatrix}\boldsymbol{v}_k\\ \boldsymbol{d}_k\end{bmatrix} = \boldsymbol{b}, \tag{2.15}$$

that for $n_s = 2$ and $n_f = 1$ becomes a square system of 4 equations in 4 unknowns. However, by increasing n_s and/or n_f, a least-squares solution can be achieved, which is robust to noise, but with some limitations. If n_s is increased, the number of unknowns do not change, i.e. the complexity of the system solution remains the same, and the baseline employed for the triangulation considered in

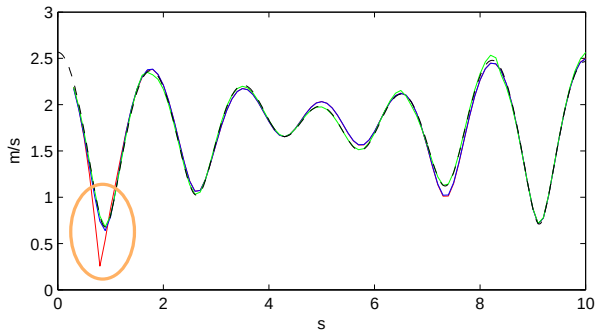


Figure 2.5: A comparison of several cases for the absolute-scale velocity estimation: true value (dark dashed line), case with $n_s = 2$ and $n_f = 1$ (red line), case with $n_s = 2$ and $n_f = 2$ (green line), and case with $n_s = 3$ and $n_f = 1$ (blue line).

the equation system is enlarged. resulting in a well numerical conditioned problem. However, in this case more IMU samples will be integrated, resulting in a bad solution is the quality of the IMU system is poor, as the typical case of MAVs. On the other hand, increasing n_f the same number of IMU data is employed but the number of unknowns increases linearly: the matrix $\boldsymbol{A}$ assumes a sparse conformation and the solution of the system becomes quickly inefficient; the complexity of the image feature matching algorithm increase and becomes less robust (increase the probability of outliers).

Taking into account these considerations, a tradeoff is required (e.g. $n_s = 3$ or 4 is a good IMU system is available, $n_f \leq 3$). A comparison between several cases is showed in Fig. 2.5, where the ideal case with $T = 10$ ms, $N = 10$ is considered. Obviously, best results are achieved when the number of image features are increased, while at the beginning of the trajectory it is noticeable a bad numerical solution for the minimum system case. This last condition happens with a significant frequency for a number of tested trajectories, then this choice it is inadvisable for a real case.

Notice that the proposed solution becomes singular when the velocity of the camera is constant, i.e. when the acceleration value remains zero over the last three camera observation points, and hence the motion remain unobservable. However, this case can be easily detected at runtime monitoring the result of the IMU integration.

2.2 Navigation control

Once estimated the vehicle velocity, the distance of each feature observed in the scene and associated to an OF element can be evaluated and collected together

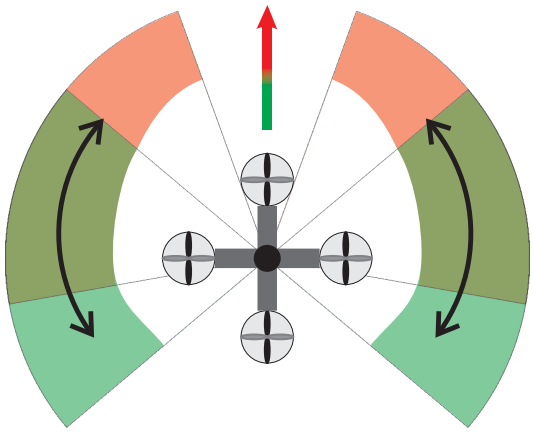


Figure 2.6: Dynamic region of interest.

with the corresponding optical rays. The result is a temporary environmental map, namely Depth Map, which can be fully exploited for lateral obstacle avoidance during the navigation.

2.2.1 Dynamic region-of-interest

The OF computation requires, as explained before, an image feature extraction algorithm and a matching algorithm, that can be computational expensive for the typical processor units available on a MAV. In the case of an omnidirectional camera, the adoption of region-of-interest (RoI) for the image elaboration processes may provide a large benefit in terms of computational requirement, while the main drawback is that the systems becomes "blind" outside the RoI. However, the adoption od a dynamic RoI that is smartly adapted online to the real environmental and navigation conditions may reduce the risk of an unpredicted impact. Observing that, due to the inertial of the system, an obstacle can be avoided only if it is detected as early as possible with respect to the vehicle velocity, the solution proposed is to adopt a RoI that "looks" more forward as the vehicle is moving quickly.

In this paper the RoI is composed of two regions, namely left and right RoI, which are symmetric with respect to the direction of motion. Both regions have a fixed total extension around the vertical axis, but they are rotated in view of an angular offset θ_{of} with respect to the navigation velocity (see Fig. 2.6). Notice that the forward region in the direction of motion is discarded due to numerical inconsistency of the OF along this direction. By denoting with θ_M the maximum offset angle for the RoI, an exponential adaptation law is considered for an offset

angle with respect to the motion direction as follows

$$\theta_{of} = \begin{cases} \theta_M \left(1 - e^{-4\frac{\|\boldsymbol{v}\| - v_m}{v_M - v_m}}\right) & \text{if } \|\boldsymbol{v}\| > v_m \\ 0 & \text{if } \|\boldsymbol{v}\| \leq v_m, \end{cases} \tag{2.16}$$

where v_m and v_M are the minimum and maximum values which can be assumed from the cruise velocity.

Also the vertical extension of the RoI is shaped in view of the offset, symmetrically reducing its range with the increase of θ_{of}. This behavior is required for omnidirectional cameras, that compresses objects extension in the image as far as they are along the direction of motion.

2.2.2 Lateral obstacle avoidance control

The safety of the vehicle during navigation within an indoor environment depends on its capability to avoid unplanned lateral obstacles.

With respect to the dynamic left and right RoI presented above and for each available DM, the distances of the vehicle with respect to the left and right side of the surrounding environment are computed with the following procedure. By denoting with $\hat{\boldsymbol{v}} = \boldsymbol{v}/\|\boldsymbol{v}\|$ the unit vector pointing along the motion direction, the distances of each detected feature, which is characterized by its feature estimated vector $\boldsymbol{f}$, along the motion direction $s_{\hat{\boldsymbol{v}}}(\boldsymbol{f}) = \boldsymbol{f}^{\mathrm{T}} \cdot \hat{\boldsymbol{v}}$ and with respect to the forward axis $d_{\hat{\boldsymbol{v}}}(\boldsymbol{f}) = \|x_{\hat{\boldsymbol{v}}}(\boldsymbol{f})\hat{\boldsymbol{v}} - \boldsymbol{f}\|$ are computed. Then, the vectors of distances from the left $\boldsymbol{d}^L_{\hat{\boldsymbol{v}}}$ and the right $\boldsymbol{d}^R_{\hat{\boldsymbol{v}}}$ sides of the navigation direction are composed using increasing values of $s_{\hat{\boldsymbol{v}}}$ as a sort criteria. Finally the minimum of each distance vector is found and a local spacial average is applied resulting in the minimum mean distances $\bar{d}^L_{\hat{\boldsymbol{v}}}$ and $\bar{d}^R_{\hat{\boldsymbol{v}}}$. Depending on the application, a LP-filter can be considered to reduce discontinuities due to the changing of the observed features.

Assuming d_{ls} as a safety lateral distance, a course correction is obtained through a PD controller acting on the following error

$$e_l = \begin{cases} \frac{\bar{d}^L_{\hat{\boldsymbol{v}}} - \bar{d}^R_{\hat{\boldsymbol{v}}}}{d_{ls}} & \text{if } \bar{d}^L_{\hat{\boldsymbol{v}}} + \bar{d}^R_{\hat{\boldsymbol{v}}} < 2d_{ls} \\ 1 - \frac{\bar{d}^R_{\hat{\boldsymbol{v}}}}{d_{ls}} & \text{if } \bar{d}^L_{\hat{\boldsymbol{v}}} \geq d_{ls}, \bar{d}^R_{\hat{\boldsymbol{v}}} < d_{ls} \\ \frac{\bar{d}^L_{\hat{\boldsymbol{v}}}}{d_{ls}} - 1 & \text{if } \bar{d}^L_{\hat{\boldsymbol{v}}} < d_{ls}, \bar{d}^R_{\hat{\boldsymbol{v}}} \geq d_{ls} \\ 0 & \text{otherwise.} \end{cases} \tag{2.17}$$

Notice that $\bar{d}^L_{\hat{\boldsymbol{v}}} + \bar{d}^R_{\hat{\boldsymbol{v}}} < 2d_{ls}$ means that the vehicle is navigating in a narrow environment, e.g. a corridor, and in this case the previous control tries keeping the vehicle in the middle of the free space, while the following cruise control reduces the vehicle velocity.

2.2.3 Cruise control

The proposed navigation control considers a *cruise velocity* of the vehicle v_c along the direction of motion in the case of free space. However, for the safety of the vehicle, when an obstacle is detected or when the dimension of the space that is free for the motion is reduced, i.e. the minimum distance with respect to the environment d becomes less than a safety distance d_s, a reduction of the navigation velocity is commanded. The module of the navigation velocity is generated applying a virtual control force f_v in the desired direction of motion, which is generated with an exponential law as follows

$$f_v = f_p \left(1 - e^{-4\frac{\|\boldsymbol{v}\|}{v_c}}\right) - \tag{2.18}$$

$$f_{sM} \left(1 - e^{-4\frac{d_s - d}{\gamma_v d_s}}\right) \left(1 - e^{-4\frac{\|\boldsymbol{v}\| - v_m}{v_c - v_m}}\right), \tag{2.19}$$

with

$$f_{sM} = \begin{cases} F_s & \text{if } \|\boldsymbol{v}\| > v_m, d < d_s \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma_v \in (0, 1)$ determines the rate of reduction of the velocity when the distance d becomes less than d_s, v_m is the minimum cruise velocity that has to be assured, and F_s is the maximum braking force.

Eventually, to avoid an obstacles without penalizing excessively the velocity also the motion direction has to be locally corrected. For this purpose, a correction of the planned motion direction is achieved taking into account the presence of lateral obstacles. By denoting with $\boldsymbol{f}_f^l$ and $\boldsymbol{f}_f^r$ the positions of the most advanced feature points, i.e. obstacles, which have been detected on the left and right side of the environment, respectively. To reduce noise effects, these vectors can be computed performing a spatial mean of a certain number of the most advanced feature points. Hence, the vector which points toward a free space direction is computed as follows

$$\boldsymbol{p}_f = \frac{1}{2} \left(\boldsymbol{f}_f^r - \boldsymbol{f}_f^l\right). \tag{2.20}$$

Finally the angular correction of the current motion direction $\Delta\psi$ is computed as the angle required to align the current velocity vector $\boldsymbol{v}$ to $\boldsymbol{v}_f$

$$\Delta\psi = \arccos\left(\frac{\boldsymbol{p}_f^{\mathrm{T}} \boldsymbol{v}}{\|\boldsymbol{p}_f\| \cdot \|\boldsymbol{v}\|}\right). \tag{2.21}$$

2.3 Simulation results

The performance of the proposed DM construction algorithm and of the navigation control has been tested with simulations using the MATLAB/Simulink environment in two different cases, with and without the correction of the planned motion direction.

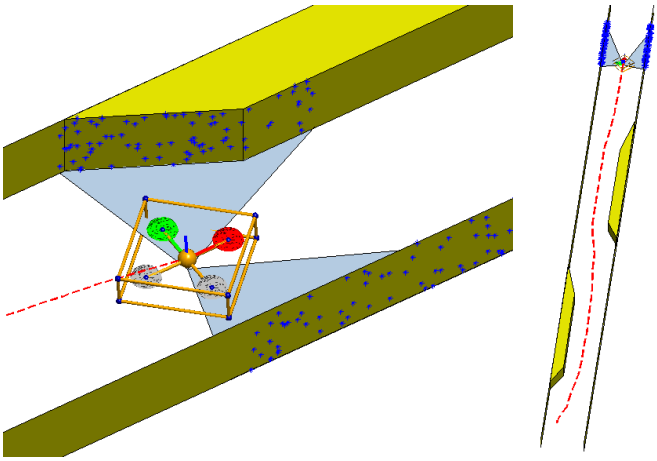


Figure 2.7: Simulated indoor environment.

In Fig. 2.7 a sketch of the employed simulator is showed. The considered indoor environment is similar to a corridor of a total length of 25 m and with a longitudinal shape that changes along the path. In particular the width of the free navigable space varies several times from 2 to 1 m, and vice versa, also changing in its middle line position.

A random occurrence of image features has been considered on both sides of the environment without outliers. Gaussian white noise has been added on image and IMU measurements. For the velocity estimation, the case $n_s = 2$ and $n_f = 2$ has been considered with $T = 0.01$ s and $N = 10$.

The adopted dynamic model of the vehicle can be found in eq.(1.1) and in [14]. The control inputs are the two tilt angles, the angular velocity around the vertical axis and the thrust, while the outputs are the position and the yaw angle. In particular, the vehicle is modeled in the inertial frame as a simple point-mass model using the second Newton's law. The forces acting on the system are the controlled thrust τ and the gravity g, as shown in the first expression in eq.(1.1). The $m = 0.5$ kg is the vehicle mass and $\boldsymbol{R}(\varphi, \theta, \psi)$ is the rotation matrix of the vehicle frame with respect to the inertial frame where angular dependency of the roll, pitch and yaw angles, has been specified. The delay acting on the control angles due to the internal controller action can be modeled as a second order system:

$$L(s) = \frac{\omega^2}{s^2 + 2 \cdot d \cdot \omega \cdot s + \omega^2}, \tag{2.22}$$

where $\omega = 15.92$ rad/s and $d = 1.22$. Supposing that the controller is fast enough and smooth, it is possible to consider the delay acting on forces and not on the angles, so to obtain four linear and decoupled systems respect to the forces as in

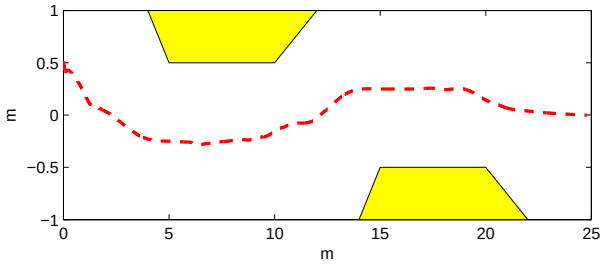


Figure 2.8: Course correction during navigation in view of the detected obstacles.

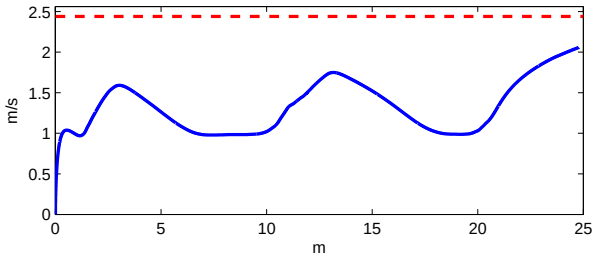


Figure 2.9: Navigation velocity modified in view of the detected obstacles and of the current free space (blue line) and adopted v_c (red dashed line).

in [14]. With respect to these parameters, the PD controller of the lateral obstacle avoidance control has been designed in the frequency domain with the following transfer function:

$$C(s) = \frac{0.008(100s + 1)}{0.001s + 1}. \tag{2.23}$$

Some of the most significant adopted parameters are as follows: $\theta_M = 30°$ for a total lateral angle of view of 80°, $v_c = 2.44\ m/s$, $v_m = v_c/4$, $d_{ls} = d_s = 1.0$ m, $\gamma_v = \gamma_l = 0.25$.

The course correction achieved during the navigation is shown in Fig. 2.8, where also the shape of the environment has been reported. The vehicle starts from the home position that is near to the left side of the environment. The path followed by the vehicle is almost centered in the middle of the available free space left to the vehicle as desired.

In Fig. 2.9 the navigation velocity modified in view of the detected obstacles and of the current free space is shown. As expected, the velocity is reduced when the vehicle is near to obstacles or in a restricted area. The cruise velocity in the narrow part of the environment is decreased, in view of the adopted parameters, to about 1 m/s, while when the available space increases also the velocity increases

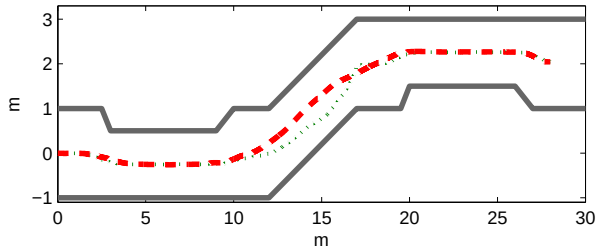


Figure 2.10: Course correction during navigation, with (red dashed line) and without (green dotted line) the motion direction correction, in view of the detected obstacles and of the path deviations (gray lines).

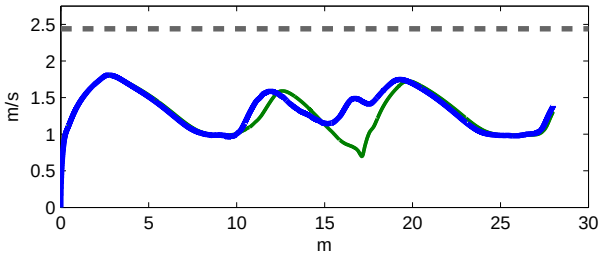


Figure 2.11: Navigation velocity modified in view of the detected obstacles and of the current free space, with (blue line) and without (green line) the motion direction correction, and adopted v_c (gray dashed line).

tending to v_c.

In the second case, a more complex and narrow environment is considered. The course correction achieved during the navigation is shown in Fig. 2.10, where also the shape of the environment has been reported. The vehicle starts from the home position that is near to the left side of the environment. The path followed by the vehicle is almost centered in the middle of the available free space left to the vehicle as desired.

In Fig. 2.11 the navigation velocity, modified in view of the detected obstacles and of the available free space, is shown. As expected, the velocity is reduced when the vehicle is near to obstacles or in a small area. In turn, the velocity in the narrow part of the environment is decreased depending on the chosen parameters, to about 1 m/s. When the available free space increases also the velocity increases towards v_c. The motion direction correction is useful for the vehicle to keep a higher velocity in the narrow part as shown in fig. 2.11, respect to the lateral control approach only.

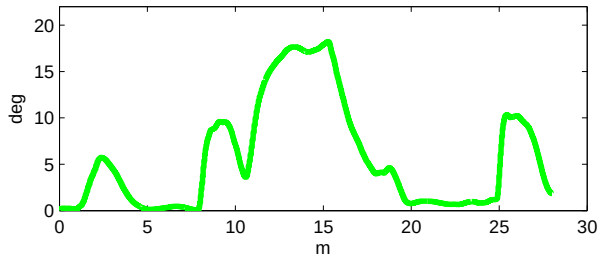


Figure 2.12: Motion direction correction during navigation in view of the detected obstacles and of the path deviations.

Finally, in Fig. 2.12 the motion direction correction which is applied during the navigation is shown. Clearly, the presence of lateral obstacles and an unexpected corridor deviation require suitable corrections to keep the vehicle in the middle of the available space without reducing drastically the velocity. Figures 2.10 and Figure 2.11 show the improvements provided by the adoption of the proposed approach.

2.4 Image based Visual Servoing

The challenge, in this second part, is not to avoid obstacles, but to provide to the vehicle the capability to interact with the environment through the use of vision sensors. The primary contribution is to enable high-speed grasping maneuvers by developing a dynamical model directly in the image space, showing that this is a differentially-flat system with the image features serving as flat outputs, developing a geometric visual controller that considers the second order dynamics (in contrast to most visual servoing controllers that assume first order dynamics), and presenting validation of the methods through both simulations and experiments[1].

2.4.1 Geometry

Let the image features be the points whose rays are tangent to the cylinder and lie in the vertical plane. In contrast to typical visual servoing approaches, these points are now a function of the position of the robot. Therefore, the standard image Jacobian, which assumes the target points are stationary in the inertial frame [29], cannot be used.

[1]It must be noted that grasping maneuvers are predominanty in the sagittal plane and thus developed models and algorithms for motion planning and control are based on a planar model ($x-z$ plane). However, since the experimental system is 3D, a Vicon-based motion capture system will be used to ensure stability for the yaw and the y-axis dynamics. The $x-z$ dynamics will be stabilized through the developed IBVS controller.

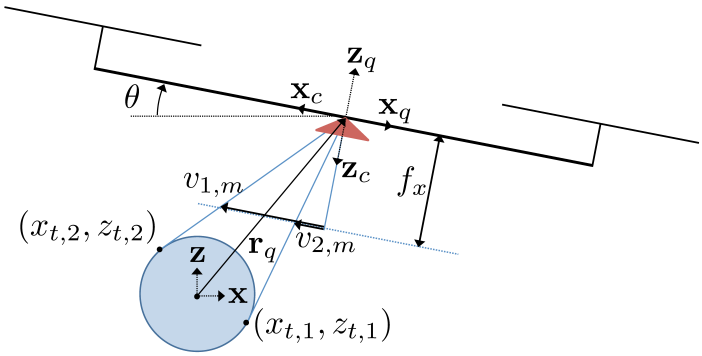


Figure 2.13: It is assumed that the target is located at the origin and the quadrotor is located at $(x_q,\ z_q)$.The focal length of the camera, f_x, defines the location of the image plane relative to the quadrotor and the image coordinates are given by v_1 and v_2. The optical ray tangent to the target intersects the target at $(x_t,\ z_t)$. The coordinate system of the camera is indicated by $\mathbf{x}_c$ and $\mathbf{z}_c$.

In order to formulate the mapping between the image plane and the robot pose, let the target cylinder be centered at the origin, R_t denote the radius of the target cylinder, and $\mathbf{r}_t$ be a tangent point on it as shown in Figure 2.13. With the camera at the same position as the quadrotor, there are two geometric constraints in the inertial frame,

$$\|\mathbf{r}_t\|_2 = R_t \tag{2.24}$$

$$\|\mathbf{r}_q\|_2^2 = \|\mathbf{r}_q - \mathbf{r}_t\|_2^2 + R_t^2 \tag{2.25}$$

where $\|\cdot\|_2$ is the 2-norm in the Euclidean space. These equations have two solutions which represent the two tangent points,

$$\mathbf{r}_{t,i} = \frac{R_t^2}{\|\mathbf{r}_q\|^2}\left(\begin{bmatrix} x_q \\ z_q \end{bmatrix} \pm \begin{bmatrix} -z_q \\ x_q \end{bmatrix}\sqrt{\frac{\|\mathbf{r}_q\|^2}{R_t^2} - 1}\right). \tag{2.26}$$

Unfortunately, the features in the image plane are coupled with the attitude. Thus, the image features would not allow for the necessary attitude-decoupled mapping between the position of the robot and the image features as required for the features to be flat outputs as outlined in section 2.5.1. Similarly to [58], the calibrated image coordinates are mapped to coordinates on a level virtual image plane by rotating the camera coordinate system to a virtual frame where $\theta = 0$.

Then, the virtual calibrated coordinates of the features can be computed using

the position of the quadrotor, (2.26), and

$$\lambda \begin{bmatrix} v_i \\ 0 \\ 1 \end{bmatrix} = P_0 T^{-1} \begin{bmatrix} x_{t,i} \\ 0 \\ z_{t,i} \\ 1 \end{bmatrix} \tag{2.27}$$

with the appropriate transformation, T, and independent of the pitch, θ. The virtual coordinates, $\mathbf{v} = [v_1,\ v_2]^T$, in (2.27) provide two equations which can be solved to determine the robot and camera position as a function of the virtual image coordinates.

The space $S = \{\mathbf{r}_q \in \mathbb{R}^2 \mid 2R_t \leq \|\mathbf{r}_q\| \leq B_r, z_q > R_t\}$, is defined such that the quadrotor's position is bounded below by $2R_t$ and bounded above by B_r, and the quadrotor is always above the cylinder. Then, there exists $V \subset \mathbb{R}^2$ and a smooth global diffeomorphism $\Gamma : S \longrightarrow V$ such that

$$\mathbf{v} = \frac{f_x}{z_q^2 - R_t^2} \begin{bmatrix} x_q z_q + R_t^2 \sqrt{\frac{\|\mathbf{r}_q\|^2}{R_t^2} - 1} \\ x_q z_q - R_t^2 \sqrt{\frac{\|\mathbf{r}_q\|^2}{R_t^2} - 1} \end{bmatrix} \equiv \Gamma(\mathbf{r}_q), \tag{2.28}$$

$$\dot{\mathbf{v}} = \frac{d\Gamma(\mathbf{r}_q)}{dt} = \frac{\partial}{\partial \dot{\mathbf{r}}_q} \left(\frac{d\Gamma(\mathbf{r}_q)}{dt} \right) \dot{\mathbf{r}}_q \equiv J \dot{\mathbf{r}}_q, \tag{2.29}$$

where J is the image Jacobian [125]. Note that J can be expressed as a function of either the image coordinates or the position of the robot by using (2.28) and the fact that Γ is invertible. Having established a mapping between the Cartesian coordinates and the image coordinates, a dynamic model of the quadrotor system directly in the image coordinates is developed.

2.4.2 Dynamics in the Image Plane

The dynamics of this quadrotor system are well known in literature. For simplicity, the robot is restricted to the vertical $(x-z)$ plane, (See [108] for the complete 3-D dynamic model and 1.1)

$$\mathbf{r}_q = \begin{bmatrix} x_q \\ z_q \end{bmatrix}, \quad \mathbf{w}_q = \begin{bmatrix} \mathbf{r}_q \\ \theta \end{bmatrix}$$

where $\mathbf{r}_q$ is the position of the quadrotor and θ is the pitch angle. Then, the dynamics in the inertial frame take the form

$$D\ddot{\mathbf{w}}_q + C\dot{\mathbf{w}}_q + \mathbf{G} = \mathbf{F} \tag{2.30}$$

where $D \in \mathbb{R}^{3\times3}$ is a diagonal inertial tensor because the robot frame is aligned with the principal axes of the inertia. In this case, centripetal and Coriolis terms, $C \in \mathbb{R}^{3\times3}$, are zero. Gravity appears in $\mathbf{G} \in \mathbb{R}^{3\times1}$, and $\mathbf{F} \in \mathbb{R}^{3\times1}$ is

$$\mathbf{F} = \begin{bmatrix} fR\mathbf{e}_2 \\ M \end{bmatrix}$$

where $R \in SO(2)$, $f \in \mathbb{R}$ is the total thrust, $\mathbf{e}_2 = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$, and M is the pitch moment generated from the difference of thrusts between the front and rear rotors. Since the system has three degrees of freedom, given by $\mathbf{w}_q$, and only two control inputs that appear in $\mathbf{F}$, the system is underactuated. Now, $\dot{\mathbf{r}}_q$ and $\ddot{\mathbf{r}}_q$ can be expressed as functions of the image coordinates using the inverse of the image Jacobian, J. Then, the dynamics in (2.30) can be expressed in terms of the image coordinates using

$$\dot{\mathbf{r}}_q = J^{-1}\dot{\mathbf{v}} \tag{2.31}$$

$$\ddot{\mathbf{r}}_q = J^{-1}\ddot{\mathbf{v}} - J^{-1}\dot{J}J^{-1}\dot{\mathbf{v}} \tag{2.32}$$

so that the dynamics in the image coordinates are:

$$\ddot{\mathbf{v}} = \frac{1}{m}J\left[fR\mathbf{e}_2 - \mathbf{G}_{1:2}\right] + \dot{J}J^{-1}\dot{\mathbf{v}} \tag{2.33}$$

$$J_q\ddot{\theta} = M \tag{2.34}$$

where $\mathbf{G}_{1:2}$ denotes the first two elements of $\mathbf{G}$. Equation (2.33) presents the translational dynamics directly in the image coordinates. In the next section, it is demonstrated that $\mathbf{v}$ forms a set of flat outputs for the system, enabling trajectory design directly in the image space.

2.5 Dynamically Feasible Trajectories

2.5.1 Differential Flatness

A system is differentially flat if there exists a change of coordinates which allows the state, $(\mathbf{q}, \dot{\mathbf{q}})$, and control inputs, $\mathbf{u}$, to be written as functions of the flat outputs and their derivatives $(y_i, \dot{y}_i, \ddot{y}_i, ...)$ [89]. If the change of coordinates is a diffeomorphism, trajectories can be planned using the flat outputs and their derivatives in the flat space since there is a unique mapping to the full state space of the dynamic system. A proposed set of flat outputs, in the image space, are the image coordinates, $\mathbf{v}$. These would be convenient since planning dynamically feasible trajectories in the image space, V, would be as simple as planning a sufficiently smooth trajectory in the image coordinates. First, there exists a diffeomorphism between the image coordinates and the position of the robot, namely Γ as defined in (2.28). From (2.33)

$$fR\mathbf{e}_2 = m_q J^{-1}\left(\ddot{\mathbf{v}} - \dot{J}J^{-1}\dot{\mathbf{v}}\right) + \mathbf{G}_{1:2} \tag{2.35}$$

and defining

$$\mathbf{F}_{1:2} = m_q J^{-1}\left(\ddot{\mathbf{v}} - \dot{J}J^{-1}\dot{\mathbf{v}}\right) + \mathbf{G}_{1:2}, \tag{2.36}$$

it can be concluded that

$$f = \|\mathbf{F}_{1:2}\|, \quad \theta = \arctan\left(\frac{F_1}{F_2}\right). \tag{2.37}$$

The derivative of (2.35) reveals that

$$\dot{f} = \mathbf{e}_2^T R^T \dot{\mathbf{F}}_{1:2} \tag{2.38}$$

and

$$\dot{\theta} = \frac{1}{f}\mathbf{e}_1^T R^T \dot{\mathbf{F}}_{1:2}. \tag{2.39}$$

The next derivative provides

$$\ddot{\theta} = \frac{1}{f}\left(\mathbf{e}_1^T R^T \ddot{\mathbf{F}}_{1:2} - 2\dot{f}\dot{\theta}\right) \tag{2.40}$$

and, using (2.34), the pitch moment is

$$M = J_q \frac{1}{f}\left(\mathbf{e}_1^T R^T \ddot{\mathbf{F}}_{1:2} - 2\dot{f}\dot{\theta}\right). \tag{2.41}$$

Upon inspection, it can be noticed that the 4^{th} derivative of the image coordinates appears in (2.41) through the $\ddot{\mathbf{F}}_{1:2}$ term, which means that trajectories in the image plane must be at least 4 times differentiable, or $\mathcal{C}^4$.

2.6 Trajectory generation

The differential flatness analysis in the Euclidean space and further examination of the control inputs reveals that the snap (4^{th} derivative) of the position of the quadrotor appears in the M term through $\ddot{\theta}$. In addition, $\beta^{(4)}$ appears in M through the $\mathbf{r}_s^{(4)}$ term in $\ddot{\theta}$. In the image plane case, the snap of the image coordinates appears in M.

Then, to minimize the norm of the input vector, it is appealing to minimize the following cost functional constructed from the snap of the trajectory:

$$\mathcal{J}_i = \int_{t_0}^{t_f} \left\| y_i^{(4)}(t) \right\|^2 dt \quad \forall\, i \tag{2.42}$$

where y_i denotes the i^{th} flat output. Accordingly, minimum-snap trajectories in the image space is considered. The minimization problem can be solved by choosing a finite dimensional basis for the trajectories and numerically solving a quadratic program (QP) [83]. If only equality constraints are needed, the QP can be solved by a single matrix inversion, and in practice, even the inequality case can be solved fast enough for real-time integration. In the implementation, the

trajectories are precomputed and the robot is controlled (using Vicon) to start at the appropriate starting point in the trajectory. The choice for this approach was motivated by ease-of-implementation and the fact that this allows the same trajectory to be flown numerous times.

The boundary conditions on the trajectories are the same as the observed boundary conditions of the trajectories of the raptors. In particular, a start and finish location are defined, and the position at pickup is defined by the target's location.

See Figure 2.18 for the inertial-frame trajectories that result from planning in the image space.

Having shown that the system is differentially flat with two sets of flat outputs in the image space, and having used the differential flatness property to generate dynamically feasible trajectories, a controller that uses vision to track features in the image space is developed.

2.7 Vision Based Control

Attitude Controller

First, let $R_d \in SO(2)$ denote the desired rotation matrix defined by a desired attitude, θ_d, and recall that R is the rotation matrix defining the current attitude. The angular rate of the robot is Ω, which, in the planar case, is equivalent to $\dot{\theta}$, and the desired angular rate is Ω_d, or $\dot{\theta}_d$. Then, attitude errors are defined

$$e_R = \frac{1}{2}\left(R_d^T R - R^T R_d\right)^\vee = \sin(\theta - \theta_d) \tag{2.43}$$

$$e_\Omega = \Omega - R^T R_d \Omega_d = \dot{\theta} - \dot{\theta}_d. \tag{2.44}$$

where $^\vee$ is the "vee" map as defined in [65]. These errors are similar to [65] but simplified for the planar case. Also, the configuration error function is defined as

$$\Psi\left(R, R_d\right) = \frac{1}{2}^{\mathrm{T}}\left[I - R_d^T R\right]. \tag{2.45}$$

The attitude controller is then given as below.

***Proposition* 1.** *[65, Prop. 1] (Exponential Stability of Attitude Controlled Flight Mode) Consider the control moment defined as*

$$M = -K_R e_R - K_\Omega e_\Omega + J_q \ddot{\theta}_d, \tag{2.46}$$

where K_R and K_Ω are positive scalars. Further, suppose the initial conditions satisfy

$$\Psi(R(0), R_d(0)) < 2 \tag{2.47}$$

$$\|e_\Omega(0)\|^2 < \frac{2}{J_q} k_R \left(2 - \Psi(R(0), R_d(0))\right). \tag{2.48}$$

Then, $(e_R, e_\Omega) = (0, 0)$ *is exponentially stable for the closed-loop system.*

Proof. Follows from [65, Prop. 1]. See Section A.1.1 for more details. □

Position Control

Let errors in the image plane be defined by

$$\mathbf{e}_v = \mathbf{v} - \mathbf{v}_d \tag{2.49}$$

where, as mentioned $\mathbf{v}$ is a vector of the image feature coordinates. Similarly, $\mathbf{v}_d$ is a vector of the desired image feature coordinates. Then, using (2.33), the image space error dynamics are

$$m_q \ddot{\mathbf{e}}_v = fJR\mathbf{e}_2 - J\mathbf{G}_{1:2} + m_q \dot{J} J^{-1} \dot{\mathbf{v}} - m_q \ddot{\mathbf{v}}_d. \tag{2.50}$$

where J is the image Jacobian and $\mathbf{G}_{1:2}$ is the first two components of $\mathbf{G}$. The visual servoing controller is then given as below.

***Proposition* 2.** *(Exponential Stability of Visual Feature Controlled Flight Mode) Consider the total thrust component along the current body frame vertical axis defined by*

$$f = \mathbf{A} \cdot R\mathbf{e}_2. \tag{2.51}$$

where

$$\mathbf{A} = \mathbf{G}_{1:2} + m_q J^{-1} \left[-K_p \mathbf{e}_v - K_d \dot{\mathbf{e}}_v + \ddot{\mathbf{v}}_d\right], \tag{2.52}$$

$K_p > 0$, $K_d > 0$, *and the commanded attitude is given by*

$$R_c \mathbf{e}_2 = \frac{\mathbf{A}}{\|\mathbf{A}\|}. \tag{2.53}$$

Finally, if the assumptions stated in Section A.1 is respected, then the zero equilibrium $(\mathbf{e}_v, \dot{\mathbf{e}}_v, e_R, e_\Omega) = (\mathbf{0}, \mathbf{0}, 0, 0)$ *is locally exponentially stable.*

Proof. See Section A.1.2. □

2.8 Vision System

The quadrotor is equipped with a global shutter Caspa$^{\text{TM}}$ VL camera and Computer on Module from Gumstix [3]. The automatic detection and tracking of the cylinder runs onboard the robot, is based on contour detection using Freeman chain coding, and is obtained using the C++ Visp library [33]. When the object is in the image and $\mathbf{r}_q \in S$, the measured image points from the camera are mapped

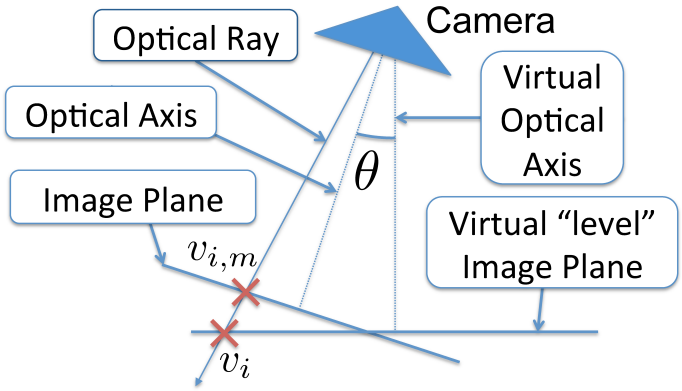


Figure 2.14: The measured image feature points, $v_{i,m}$,which are affected by θ, are projected onto a virtual level image plane to decouple the motion from the attitude of the robot and determine the coordinates v_i.

to the virtual image plane using feedback from the IMU and the transformation shown in Figure 2.14, which is mathematically equivalent to

$$v_i = \tan\left(\arctan\left(v_{i,m}\right) + \theta\right) \tag{2.54}$$

where $v_{i,m}$ is the boundary of the cylinder as measured in the calibrated image.

The points in the virtual plane are filtered to improve the estimate of the image features and their derivatives to compute J and $\dot{J}$. A block diagram of the system is shown in Figure 2.15. Since the visual controller is only designed for motion in the vertical plane, in experimentation, an external motion capture system is used as feedback to stabilize the yaw and out of plane motion. Note that the vision based controller stabilizes motion in the vertical plane as designed.

Having briefly described the experimental platform that's being used, next section presents experimental results to validate the proposed methods of trajectory generation and tracking to achieve dynamic grasping.

2.9 Simulation Results

Using the trajectory generation method outlined in Section 2.6, sample trajectories can be generated directly in the image coordinates, representing a swooping maneuver. It is reasonable to specify a limit on the attitude, which enables the incorporation of linear visibility constraints, rather than requiring non-linear visibility constraints when planning in the Cartesian space. A sample trajectory is shown in Fig. 2.16 (top), where the boundary conditions and intermediate waypoint were computed using Γ, and with the derivatives in the intermediate waypoint left unconstrained. Next, using the generated desired trajectory in the image

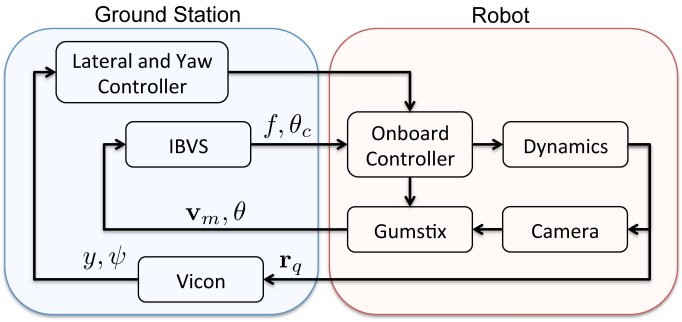


Figure 2.15: A camera captures images of the cylinder, which are sent to the Gumstix Overo Computer on Module (COM) and processed at 65 Hz using blob tracking. The boundaries of the cylinder are undistorted, calibrated, and sent back to a ground station along with the pitch as measured from the IMU. Then, the ground station maps the points to the virtual plane and computes desired control inputs using the IBVS controller. Simultaneously, Vicon feedback is used to close the loop on the roll and yaw of the robot. Then, the desired attitude is sent to the onboard controller, which uses the IMU to control the attitude at 1 kHz.

plane, the controller from Section 2.7 is simulated on the dynamic model given by (2.33)-(2.34). The simulation is started with an initial image coordinate error of $0.10m$, and the resulting trajectory and error are plotted in Fig. 2.16.x

2.10 Experimental Results

The stability of the proposed visual controller is demonstrated through several different experiments including hovering, vertical trajectories, "swooping" trajectories, and hovering above a moving cylinder. Here a sample "swooping" trajectory, which includes components from several of the previously mentioned trajectories, is presented. See Figure 2.17 for the planned and actual trajectories in the virtual image plane, Figure 2.18 for the corresponding estimated and actual position in the inertial frame, Figure 2.19 for a sequence of still images from a sample experiment, and the supplementary video for footage of sample trajectories.

The results of the vision based control are shown in Figures 2.17 and 2.18. In these, a "swooping" trajectory is executed with a variation of 1 m in the z direction and 50 cm in the x direction. The system is stable, and it is possible to notice that the swooping trajectory in the Cartesian space, as shown in Figure 2.18, corresponds to a desired planned and executed trajectory in the image space Figure 2.17. This is an experimental demonstration of the success of the proposed theoretical approach. In the Cartesian space, the error is quite small in the z direction, which presents a larger spatial change compared to x direction.

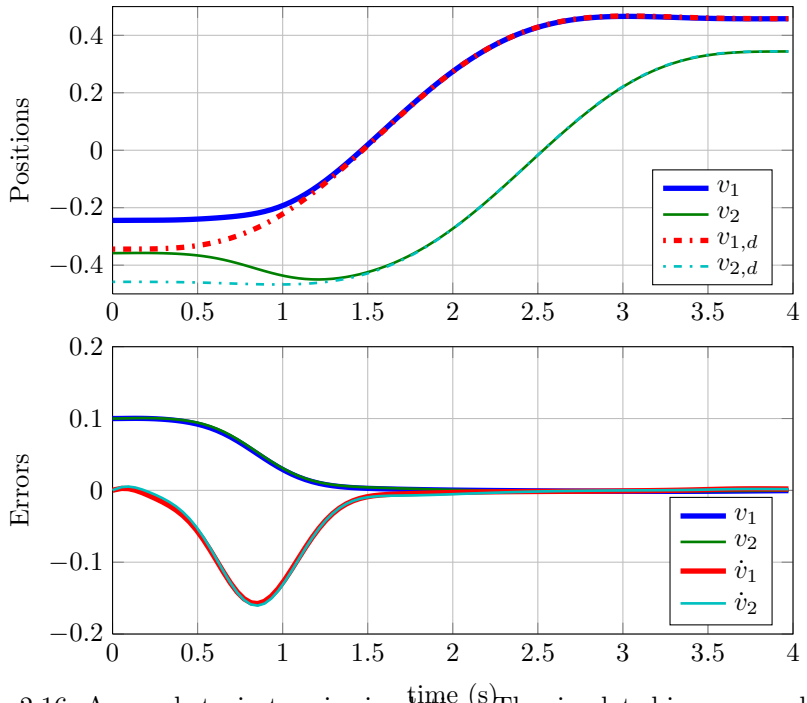


Figure 2.16: A sample trajectory in simulation. The simulated image coordinates, v_i, and the desired coordinates, $v_{i,d}$, are in the top graph where there is an initial error of $0.1m$ in each coordinate. The feature errors and error velocities are in the bottom graph.

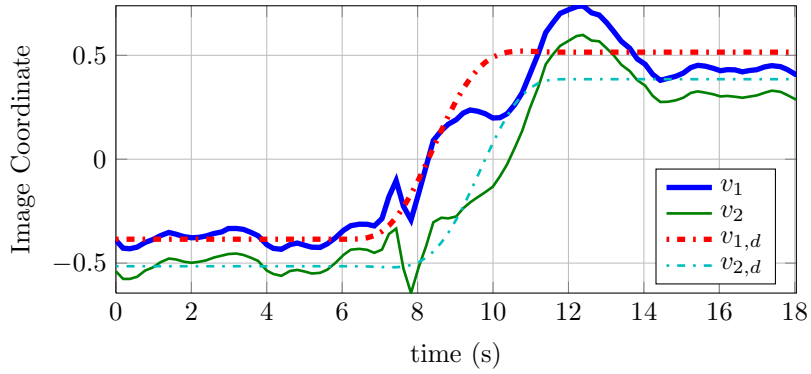


Figure 2.17: Experimental results of the feature coordinates in the virtual plane for a "swooping" trajectory. The feature coordinates are denoted by v_i and the desired trajectory is given by $v_{i,d}$.

Moreover, the z direction is the most challenging from a vision control point of view since the only source of information to recover the scale is the cylinder size.

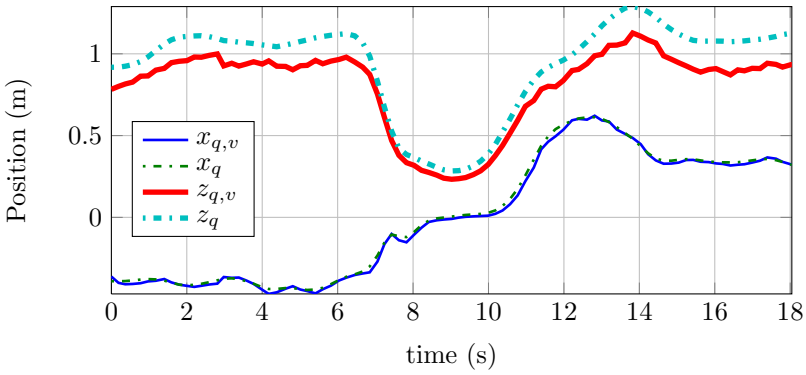


Figure 2.18: Positions in the inertial frame for the experiment in Figure 2.17. The vision estimates of the position (using Γ) are denoted by the "v" subscript. The ground truth only has the "q" subscript.

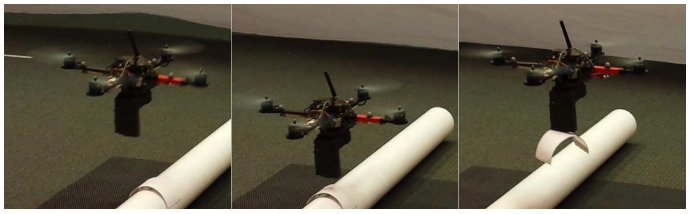


Figure 2.19: mages from a sample "swooping" trajectory using the vision-based controller developed in this paper.

Limitations and Future Challenges

It is also important to recognize that the experimental trajectories for the vision-based control (Figures 2.17 and 2.18) are not as fast as the trajectories with control feedback in the inertial space, which demonstrated aggressive grasping at speeds up to 3 m/s, shown in [119]. There are several reasons: the feedback is only from sensors onboard the robot (in contrast with an external motion capture system), the rate of feedback is nearly half in the vision-based case since a space, weight, and power constrained camera and computer are used, the position feedback loop is now closed using the onboard IMU, and the camera has a limited field of view. Although high speed visual control has been demonstrated earlier [103], it has not been achieved on space, weight, and computationally-constrained platforms. Thus, it is natural to expect trajectories that are not as aggressive. The main goal is to show the feasibility of the proposed approach with a minimal sensor suite. Further, it can be noticed that trajectory tracking is not perfect and attribute this to modeling errors such as distortion from the camera lens and external disturbances

such as ground effect and the disturbed aerodynamics after the target is captured. Future work and the advancement of technology will help to reduce the limitations with the goal of eventually achieving similar performance to the experiments in a structured environment.

In the vision-based case, currently effort are concentrated to stabilize the lateral dynamics. This is mainly because the lateral velocity is not observable from the features selected. In future work, the feedback will be augmented with optical flow for velocity estimates, and perhaps extend the feature points to be tangent lines (parallel to the axis of the cylinder), which would help provide an estimate of the roll of the robot.

The current vision approach requires the radius of the cylinder to be known a priori. In many cases, however, proper identification of the cylinder may lead to a good estimate of the size. For example, there are many common cylinders of similar or standard size such as railings and pipes. Additionally, once there is one successful grasp, the desired location of image features can be recorded to enable future grasping without needing to determine the size of the cylinder. Thus, this approach will not be difficult to generalize to grasping of unknown cylinders.

Perching

As researchers continue to develop quadrotors, the added ability to perch will be critical in extending mission time. Unlike grasping and perching using fixed wing vehicles, the two tasks are very similar for quadrotors. The only difference for a quadrotor is that the planned trajectory would stop at the bottom of the swooping behavior in order to perch. Using the proposed trajectory methods and control schemes, this task would be a simple extension of the current work.

Chapter 3

Vision for Pose Estimation and 3D Reconstruction

The combination of synchronized data between the RGB and the depth sensors for multiple robots is exploited to create a complete framework for localization and mapping. The central idea is to decompose the problem into (a) a centralized monocular SLAM problem with sparse representation involving features that are tracked by individual vehicles avoiding map merging scaling issues like in [42]; and (b) the problem of associating robot poses and depths with features to creates a dense 3-D map, a problem that can be solved in a distributed way. By decomposing the problem in this way, the computational bottleneck of 3-D RGB-D cooperative SLAM is avoided and an increased robustness to noise in depth which is typical in outdoor or brightly lit environments is achieved. Specifically, the algorithm is fast, robust, and lends itself to real-time computation with 30 Hz pose estimates for feedback for control. Second, it allows dense 3-D mapping. Third, it allows multiple robots to localize to the same coordinate system. It provides a more efficient way of sharing a common high-resolution RGB map of the environment, avoiding redundancy. The framework has been developed under ROS [6] and is available online[1] Finally, in the last part of this chapter a different environment representation is introduced for a high-level supervisory control, to solve the problem of large amount of data and sparsity related to a point cloud representation.

3.1 Visual Egomotion

The ASUS Xtion sensor is employed to localize the vehicle in the environment by coupling the monocular multi-map visual odometry algorithm proposed in [28]

[1] `https://github.com/loiannog/PTAMM_RGBD_cooperative`.

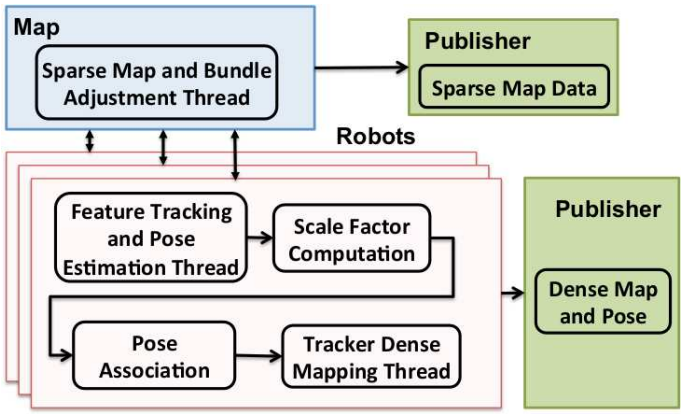


Figure 3.1: Framework representation. The software that runs on robots (shaded pink) is based on a distributed algorithm while all other modules are centralized.

with depth data provided by the Infrared (IR) camera obtaining a real-time visual SLAM algorithm and a dense colored map. A multi-threaded programming approach allows fast localization and cooperative mapping at an average rate of 30 Hz suitable for real-time applications. A schematic representation of the approach is given in Fig. 3.1. A brief description of the monocular algorithm and its improvements are given in the following.

3.1.1 Visual Framework

The SLAM task is split into two parallel tasks, namely the tracking task and the mapping task, which are executed in parallel threads.

The tracking task (shaded pink and labeled "robots" in Fig. 3.1) is responsible for the tracking of salient features in the camera image determining camera position. This is done with the following steps: first, a simple motion model (using constant velocity in the experiments) is applied to predict the new pose of the camera. Then, the stored map points are projected into the camera frame, and the corresponding features' FAST corners [95] are searched to solve the data association problem. The orientation and position of the camera is refined such that the total error between the observed point features and the projection of the map points into the actual frame is minimized.

In parallel, the mapping task (blue box in Fig. 3.1) uses a subset of all camera images called keyframes to build a 3D point map of the environment. After adding a new keyframe, a batch optimization is applied to refine both the map points and the keyframe poses. The main goal is the so-called *bundle adjustment* which involves the minimization of the total back-projected map error.

The adopted algorithm, to reduce computational effort, does not use an EKF based state estimation and does not consider any uncertainties, both for the camera and the feature location. As demonstrated in [111], the keyframe SLAM outperforms the classical filter-based SLAM approach in [35]. The lack of modeling uncertainties is compensated by the use of a large number of features and the global batch optimization. The algorithm is fast and reliable, and the map is very accurate. This version of the framework provides enhancements to the original version of PTAM, allowing users to save the state of a map and corresponding keyframes to disk, as well as initialize a new map.

3.1.2 Visual Framework Extensions

In addition to enhancements to the interface, new extensions have been developed

Scale factor estimation

The missing scale factor in monocular visual odometry has been estimated as shown in Fig. 3.1 using the procedure mentioned in the next section to obtain a coherent absolute pose and compensate for the odometry drift.

Depth Initialization

The user can decide to initialize the map associating each extracted feature to the corresponding depth, without performing a planar motion required by the visual SLAM.

User interface

The user can use a simple interface to visualize mapping results, setup parameters, and remotely control the algorithm's behavior from a ground-station. Moreover, the 6 DoF pose is published as a pose with a covariance estimation calculated from bundle adjustment.

Features

While the lowest level features are included for tracking, the user can choose to omit them in map-handling, storing features only in the highest 3 pyramidal levels. This is useful in similar-structured environments since features extracted at higher pyramidal levels are more robust to scene. Moreover, this speeds up the keyframe insertion. Finally, corner extraction utilizes the AGAST corner detector [78], which is faster than the original FAST [95].

3.1.3 Cooperative Mapping

The user can define multiple trackers, which allow the possibility to carry multi-camera pose estimation. Each tracker is a boost thread[2] running in parallel with all other trackers (see Fig. 3.1). The relocalizer function, based on [123], identifies the pose of extra-cameras in the actual map. When a new camera is introduced, the keyframe descriptors, extracted from subsampled images (80×60), are compared to the current camera image descriptor (each comparison takes around 0.016 ms on a single core machine) to find the one which minimizes the sum of square differences. This keyframe is accepted as a match, and the camera position is resumed to that of the keyframe. The rotation of the camera is estimated using a direct second order minimization. Naturally, the new cameras start in a confined location, the subset for searching can be reduced. Then, all cameras can cooperate to build the same map. The employed locking strategy uses shared and upgradable locks which allows other threads to simultaneously read the data except for the negligible time when the map is updated. The only requirement is that subsequent trackers must register their camera poses to the maps established coordinate system. This registration is exactly what is required to relocalize by a lost camera during the normal run of single camera tracking. The registration is made using appearance, not structure, so that a camera can be localized without first building its own map. The presented strategy is useful in case different vehicles should contribute to an existing map. A new thread is instantiated for every new vehicle. Finally, it should be pointed out that if dedicated data structures are implemented, the mapping part could be placed on a ground station while the tracking could run on-board the vehicle providing a distributed characteristic to the framework.

3.2 Scale Factor Estimation

The monocular SLAM framework can provide the translational motion of the sensor in the environment up to a scale factor, since a single camera is employed. The scale factor parameter is estimated through the combination of depth data and visual data since the RGB and the depth images can be hardware synchronized. An analysis, based on public datasets, is provided to validate the effectiveness of the proposed estimation approach.

3.2.1 3D Point Cloud Generation

The measurement of the depth data is achieved by a triangulation process [45], during which the IR projector and the IR camera (see Fig. 3.2) generate a disparity image. For each pixel, the distance to the sensor can then be retrieved from the corresponding disparity. In the following, a depth coordinate system, which has an

[2]Boost library: www.boost.org

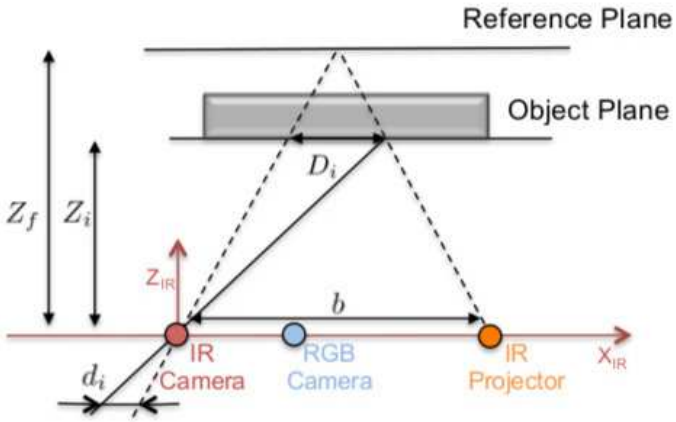


Figure 3.2: Geometry of IR Camera. When a speckle is projected on an object whose distance to the sensor is smaller or larger than the one of the reference plane, the position of the speckle in the infrared image will be shifted in the direction of the baseline between the laser projector and the perspective center of the infrared camera (red).These shifts are measured for all speckles by an image correlation procedure, which yields a disparity image.

origin at the perspective center of the infrared camera sketched in red in Fig. (1.6) and in Fig. (3.2), is considered. The Y axis is orthogonal to X and Z axis making a right-handed reference system. The RGB coordinate system is aligned with the infrared reference system.

Suppose that an object is on the reference plane at distance Z_f with respect to the sensor, and the corresponding speckle is captured on the image plane of the infrared camera as shown in Fig. 3.2. If the object is shifted with respect to the sensor, the i-th speckle on the image plane is displaced in the X direction, which corresponds to a disparity d in the image space. The disparity d_i is strictly related to D_i, the displacement of a generic point i on the image, to Z_f representing the distance from the reference plane, and to Z_i (depth), which denotes the distance of a point i with the following relations

$$\frac{D_i}{b} = \frac{Z_f - Z_i}{Z_f} \tag{3.1}$$

$$\frac{D_i}{f} = \frac{d_i}{Z_i}. \tag{3.2}$$

By solving for D in (3.2) and substituting into (3.1), the following relation is obtained

$$Z_i = \frac{Z_f}{1 + \frac{Z_f}{fb} d_i}, \tag{3.3}$$

where b represents the base length between the IR projector and the infrared camera, and f is the focal length. All these parameters can be determined with a suitable sensor calibration procedure. The other two coordinates of the object's 3D position can be determined by the classical perspective projection model as follows

$$X_i = -\frac{Z_i}{f}(x_i - x_c + \delta_x),\ Y_i = -\frac{Z_i}{f}(y_i - y_c + \delta_y), \tag{3.4}$$

where x_c, y_c represent the coordinates of the principal point, δ_x, δ_y the lens distortion correction parameters, and x_i, y_i the corresponding normalized image coordinates.

3.2.2 Scale factor computation

The intrinsic and the extrinsic calibration parameters are supposed to be available for both the IR and for the RGB camera. Let $\mathbf{p}_i^{IR}$ be the i-th point, expressed in the IR frame, which belongs to the point cloud set

$$\mathbf{P}^{IR} = \left[\ \mathbf{p}_1^{IR}, \cdots, \mathbf{p}_n^{IR}\ \right]^{\mathrm{T}}, \tag{3.5}$$

where $\mathbf{p}_i^{IR} = [X_i\ Y_i\ Z_i]^T$. The previous point cloud can be represented in the RGB frame as follows

$$\mathbf{p}_i^{RGB} = \mathbf{t}_{IR}^{RGB} + \mathbf{R}_{IR}^{RGB}\mathbf{p}_i^{IR}, \tag{3.6}$$

where $\mathbf{R}_{IR}^{RGB}$, $\mathbf{t}_{IR}^{RGB}$ are the rotation matrix and the translation vector of the IR camera with respect to the RGB camera, respectively, which are provided by the camera calibration procedure.

The depth values corresponding to each feature extracted within the visual framework have to be identified to evaluate the 3D point cloud using eqs. (3.3) and (3.4). The ratio between the Euclidean distance of each 3D point, generated in the visual framework

$$\Delta p_i^{RGB} = \|\mathbf{p}_i^{RGB}\| \tag{3.7}$$

and the same distance computed for the depth points

$$\Delta p_{v,i} = \|\mathbf{p}_{v,i}\| \tag{3.8}$$

with $i = 1, \ldots, n$, gives a set of scale factors

$$\mathbf{s} = \left[\ s_1, \cdots, s_n\ \right]^{\mathrm{T}},\ s_i = \frac{\Delta p_i^{RGB}}{\Delta p_{v,i}}. \tag{3.9}$$

The current estimation of the scale factor corresponds to the mean value of $\mathbf{s}$. Then, a recursive procedure (see bottom part in Algorithm 1), deletes scale factors that are far away from the mean by more than twice the standard deviation. A schematic representation of the estimation process for a sample frame is given in Algorithm 1.

Algorithm 1 Scale_Factor_Estimation(RGB_{image},$Depth_{image}$)

extract_image_features_set(RGB_{image})→ $\mathbf{F} = \left[\ \mathbf{f}_1, \cdots, \mathbf{f}_n \ \right]^T$;
compute_3D_points(RGB_{image}, $\mathbf{f}$) → $\mathbf{p}kinect_schema_{v,i}$;
compute_depth_points($Depth_{image}$) $\mathbf{p}_i^{IR}$;
rotate and_translate($Depth_{image}$) → $\mathbf{p}_i^{RGB} = \mathbf{t}_{IR}^{RGB} + \mathbf{R}_{IR}^{RGB}\mathbf{p}_i^{IR}$;
for $\forall \mathbf{f}_i$ **do**
  $\mathbf{f_i} \rightarrow depth_i$;
  $\Delta p_i^{RGB} = \|\mathbf{p}_i^{RGB}\|$;
  $\Delta p_{v,i} = \|\mathbf{p}_{v,i}\|$;
  $s_i = \frac{\Delta p_i^{RGB}}{\Delta p_{v,i}}$;
end for
$\mathbf{s} = \left[\ s_1, \cdots, s_n \ \right]^T$;
is_deleted = true;
while is_deleted **do**
  is_deleted = false;
  s_f = mean($\mathbf{s}$);
  σ = std($\mathbf{s}$);
  for $\forall i$ **do**
    if abs($s_i - s_f$) $\geq 2\sigma$ **then**
      delete s_i;
      is_deleted = true;
    end if
  end for
end while
return s_f

Notice that the combination of the depth and visual data is strictly required only for the first frame. Moreover, s_i values could be computed only for the extracted features. However, since visual odometry algorithms compute the camera path incrementally, the errors introduced by each new frame-to-frame motion accumulate over time. The scale computation at each frame, as shown later, strongly reduces the odometry drift, helping the bundle adjustment procedure. The proposed solution is computationally inexpensive, the scale factor estimation requires an average of 7 ms/frame to be computed, and it allows the evaluation of the environment map directly from the cloud points given by the IR/depth image.

3.2.3 Scale factor performance analysis

The performance of the previous procedure to compute the scale factor has been tested using a third party RGB-D dataset [112]. The dataset provides synchronized images and the ground truth for each trajectory from a motion capture system. Given a set of estimated poses, $P_e = \{P_{e_i} \in \mathrm{SE}(3), i = 1, \cdots n\}$ and ground truth poses, provided by a motion capture system, $P_{mc} = \{P_{mc_i} \in \mathrm{SE}(3), i = 1, \cdots n\}$, where n is the total number of samples, the root mean square error (RMSE) of the translation relative pose error (RPE) and the RMSE of the translation absolute

trajectory error (ATE) with respect to the motion capture system are evaluated. The RMSE of the translation RPE is defined as

$$\mathrm{RMSE}_{\mathrm{RPE}} := \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left\|\mathrm{trans}\left(\left(P_{mc_i}^{-1}P_{mc_{i+\Delta}}\right)^{-1}\left(P_{e_i}^{-1}P_{e_{i+\Delta}}\right)\right)\right\|^2} \tag{3.10}$$

where Δ is generally chosen as 1 in order to obtain an estimation of the drift per frame, trans is the translation component of the error, and $m = n - \Delta$. This metric essentially evaluates an error position increment with respect to a ground truth. It allows us to compare the RGB visual SLAM (without absolute scale) with the proposed approach. It can be considered as a direct measurement of how the depth affects the RGB SLAM. The RMSE of the translation ATE is defined as

$$\mathrm{RMSE}_{\mathrm{ATE}} := \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|\Delta P_{i_t}\|^2} \tag{3.11}$$

where ΔP_{i_t} represent the translation part of the error $\Delta P_i = P_{e_i} - P_{mc_i}$, supposing that both sequences are synchronized and defined in the same reference frame. Table 3.1 summarizes the RMSE of the translation RPE for RGB only and RGB with the estimated scale factor in different textured scenes. The result is a large error reduction of around 75%, which confirms the validity of the presented approach to overcome the scaling problem. Table 3.2 summarizes the RMSE of the

Table 3.1: Comparison of the RMSE translation drift (RPE) (m/s) between the RGB only and the proposed RGB+Depth approach.

Dataset	Distance	RGB	RGB+Depth
fr1/xyz	mixed	0.095986	0.037778
fr1/floor	mixed	0.144712	0.049756
fr2/xyz	mixed	0.078383	0.011542
fr3/nostruct/text	near	0.138694	0.027775
fr3/nostruct/text	far	0.107612	0.028689
fr3/struct/text	near	0.254303	0.023267
fr3/struct/text	far	0.120925	0.017481
avg.improvement		–	75%

translation of ATE for RGB+Depth in different textured scenes in two different cases. In the first column, the scale factor is estimated only at the first frame, keeping it constant along the whole motion, while in the second, it is estimated frame by frame. The final error is of the same magnitude of the one presented in Table 3.1. As expected, the estimation frame by frame contributes to an average error reduction of 30%. It is worthwhile to have a complete idea of the system performances comapred to the current literature, even if out of the scope of this

work. Thus, the last column in Table 3.2 provides some results from one of the most used open source RGB-D SLAM algorithms [39] based on the online datasets. The results show the competitiveness of the presented approach obtaining comparable errors, but with a higher speed, since the presented approach can work at the frame-rate. The presented analysis shows benefits in terms of RPE and ATE

Table 3.2: RMSE of the ATE (m) for the RGB+Depth algorithm in the two mentioned cases and comparison with RGBD-SLAM.

Dataset	Distance	RGB+D 1 fr.	RGB+Depth	RGBD-SLAM
fr1/xyz	mixed	0.031556	0.026185	0.013473
fr1/floor	mixed	0.035546	0.02561	0.035169
fr2/xyz	mixed	0.029919	0.027226	0.026112
fr3/nostruct/text	near	0.032	0.028351	0.087121
fr3/nostruct/text	far	0.060984	0.025456	0.014108
fr3/struct/text	near	0.024541	0.022574	0.034389
fr3/struct/text	far	0.064303	0.014131	0.013496
avg.improvement		–	30%	–

helping the local-global *bundle adjustment* to reduce the drift. Fig. 3.3 shows that the matching between 3D points of the monocular algorithm and corresponding depth points in a sample frame using the estimated scale factor is reliable. In Fig. 3.4, the full scale factor's variation for the fourth experiment is represented.

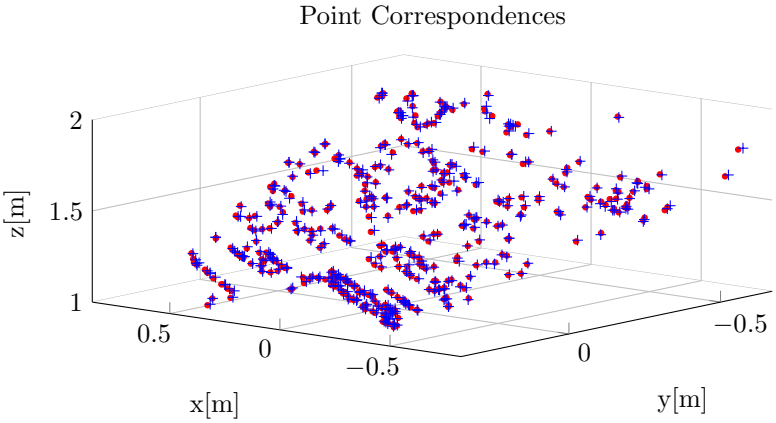


Figure 3.3: Correspondences between 3D points of the monocular algorithm (red points) and corresponding depth points (blue crosses), with the estimated scale factor, in a sample frame in the fourth dataset.

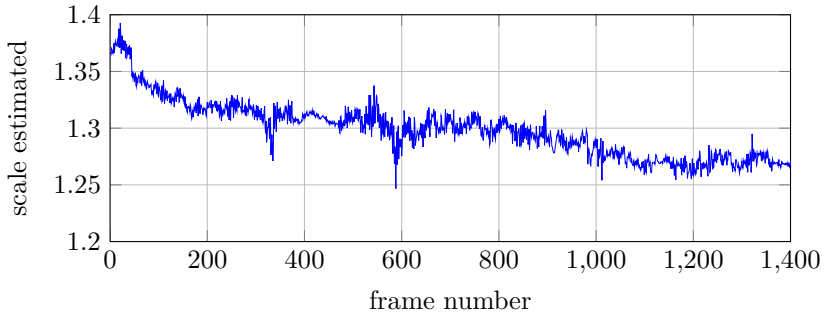


Figure 3.4: A representative plot showing the history of estimated scale factors obtained from the fourth dataset.

3.3 3D Mapping and Reconstruction

The environment map can be generated either using the depth values, suitably synchronized with respect to the tracker pose (see Fig. 3.1 for a representation), or using the sparse map provided by the visual odometry framework. Both maps and pose estimator run in separate threads improving the reliability of the proposed algorithm on a multi-core platform. In the case of a distributed use of the framework, the sparse map is centralized, while the dense map is distributed and defined on the tracker side.

3.3.1 Dense mapping

During the algorithm initialization, when the scale factor estimation is performed, the absolute position and orientation are computed and an appropriate synchronization is performed to associate them to the point clouds generated from the IR sensor. The result is a colored map given by a combination of the RGB image pixel colors and the depth values. Generally, if a depth image resolution 640×480 pixels is considered, $300,000$ points are generated. To avoid an excessive memory allocation, which can affect computation performance, without reducing the spatial accuracy of the environment map, a new multi-resolution approach is proposed for point cloud sampling avoid unnecessary memory allocation in the case of a large environment. As shown in Section 3.2, the disparity is directly related to the point cloud data. The disparity image is subsampled using a virtual image grid, which is composed of rectangles of size $D_R \times D_C$ pixels (see Fig. 3.5). In particular, for each rectangle of the grid, a spatial sampling with a step $\Delta_R \times \Delta_C$ is chosen accordingly to the following law

$$\Delta_i = \Delta_{i,min} + \frac{\Delta_{i,max} - \Delta_{i,min}}{Z_{max} - Z_{min}}(Z_{max} - Z_C) \tag{3.12}$$

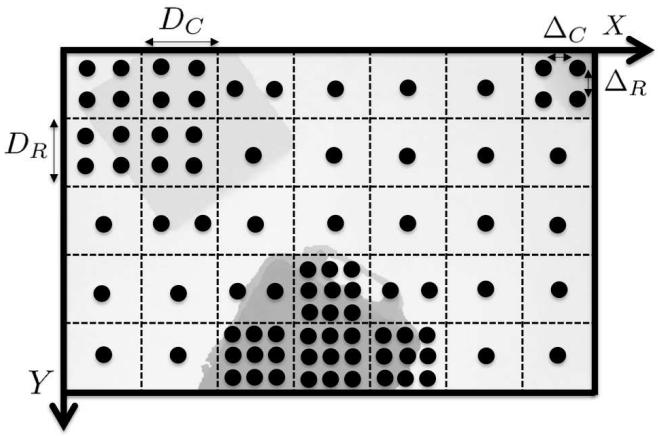


Figure 3.5: Disparity image subsampled. Grey regions are subsampled with a smaller interval compared to white ones, since a higher grey level indicates a closer object in the camera frame.

with $i = R, C$, where Z_C is chosen as the average distance between the rectangle and the environment expressed into the IR frame. The terms $\Delta_{i,min}$ and $\Delta_{i,max}$, represent the minimum and maximum values for the spatial sampling distance, respectively. For each rectangle, the number of points added to the global map varies as shown with a linear law between a maximum distance Z_{max} and the minimum distance Z_{min}, which the sensor is able to detect.

Moreover, time sampling is also adopted depending on the sensor's linear and angular velocity in order to avoid explicit useless points added when the platform is almost fixed. In detail, the map publishing thread takes care of the map streaming and pose visualization in real-time, while the map data storage is updated with new points according to the following time laws

$$F = F_{max} - (F_{max} - F_{min}) \cdot \max\left(\frac{v}{v_{max}}, \frac{\omega}{\omega_{max}}\right) \tag{3.13}$$

where F is the map update frequency, F_{max} and F_{min} are the maximum and minimum publication update frequencies, v and ω are the mean linear and angular velocity norms, respectively, performed on the last k time instants

$$v = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\Delta T}(p_i - p_{i-1}), \quad \omega = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\Delta T}\theta_{i-1}^{i}, \tag{3.14}$$

where ΔT is the time interval between two consecutive measurements, p_i is the position norm at time instant i, while θ is obtained from the axis-angle orientation representation of two consecutive time rotations.

3.3.2 Sparse mapping

A problem with these sensors is their efficiency in outdoor environments, where depth generation is compromised due to outside light intensity. For this reason, the presented framework gives the opportunity to use, instead of IR camera points, 3D points in the fixed frame generated by the visual SLAM algorithm [28]. All the points create, in this case an absolute environment map.

3.4 Experimental Results

The OpenNI driver [4] has been employed for the sensor interfacing, which provides the capability of choosing between different configurations in terms of image resolution and update frequency. In the considered experimental case study, the RGB and depth data are streamed synchronized with a frequency of 30 Hz and a maximum resolution of 640 $\times$ 480 pixels. The proposed framework has been encapsulated in a ROS [6] node and the time synchronization between RGB image and depth is realized via the ROS message synchronization mechanism.

3.4.1 Hand held performance evaluation

To show the effectiveness of the proposed approach the PRISMA Lab has been reconstructed. Figure 3.6 shows the original room (on the top) and the corresponding dense colored map (on the bottom), while the sensor trajectory is depicted with a blue line inside the map environment.

Moreover, to evaluate the effectiveness of the proposed scale factor estimation method, the sensor has been moved along a trajectory of about 8 m long, which is shown in Fig. 3.7, estimating the scale only at the first frame. In order to provide a ground truth for the proposed egomotion estimation algorithm, an OptiTrack motion capture system [5] composed of ten S250e cameras has been employed to track the sensor during its motion at 250 Hz.

The time history of the norm of the motion estimation error with respect to the ground truth shown in Fig. 3.8 highlights a 15 cm peak over a mean of about 3.8 cm, which confirms the effectiveness of the proposed approach. In fact, the sensor accuracy for the depth measurement at 4 m of distance is declared in 3 cm, and thus the performance of the proposed approach is in line with the sensor intrinsic performance. The positional norm error is decreasing after 30 seconds since the camera is back to previous mapped positions, where keyframes have already been instantiated. The results can be seen in a video available at the link mentioned in the Introduction.

Figure 3.6: Dense colored map reconstruction: on the top the real environment; on the bottom the achieved map. The blue line indicates the sensor motion within the environment as measured by the visual egomotion estimation algorithm with the proposed scale factor computation. The map can be published up to 20 Hz ($F_{max} = 20$ Hz and $F_{min} = 10$ Hz), with $\Delta_{R,max} = 10$, $\Delta_{R,min} = 5$, $\Delta_{C,max} = 20$, $\Delta_{C,min} = 10$.

3.4.2 Flying performance evaluation

In this section, system setup and experimental results are provided based on different flying trials. The platform is the Asctec [1] Hummingbird (see Fig. 3.9), equipped with an Intel i5, 1.8 GHz computer and a downward facing ASUS Xtion, mounted in the ventral part of the vehicle. The sensor case has been removed, and the USB cable was shortened to reduce weight. The experimental results, are based on data collected at the GRASP Lab at the University of Pennsylvania [86]. The vehicle was controlled to fly in the $x - y$ direction of Vicon's reference frame, within a 2.5 $\times$ 2 m area. The altitude varies from 0.5 m, which represents the landing altitude, to 2.5 m. The maximum speed was set to 0.6 m/s and the maximum acceleration at 0.8 m/s due to safety reasons. The variation in altitude was performed in order to verify the robustness of the scale factor estimation on a flying vehicle platform. Different waypoint trajectories, relying on the Vicon motion capture system [7] for control feedback, are performed. The collected dataset is

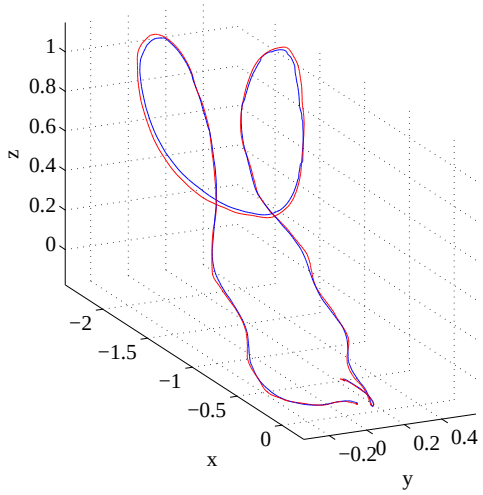


Figure 3.7: Path trajectory of the sensor (in red) and the corresponding ground-truth (in blue) provided by the Optitrack motion capture system.

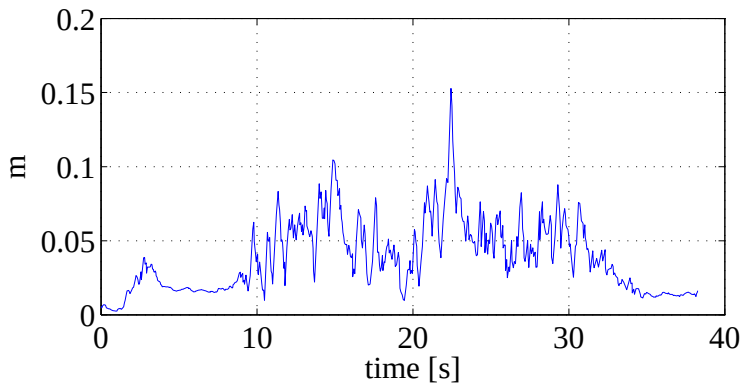


Figure 3.8: Time history of the positional norm error for the path of Fig. 3.7.

running, using Ubuntu operating system, on a Mac-Book Pro i7 2.8 GHz.

In all the experiments reported here, a vehicle starts the map and a second one is introduced at the occurrence. It re-localizes using the existing sparse map and contributes to map's expansion. Absolute error positions are provided in Table 3.3. The results show again, an error improvement when the scale is continuously estimated. Moreover, the error presents the same magnitude error compared to the datasets analyzed in Section 3.2, despite the vehicles' vibrations, which can

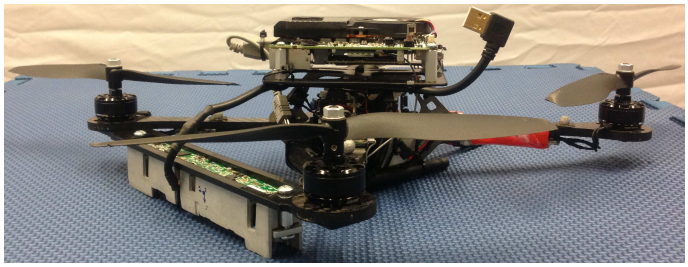
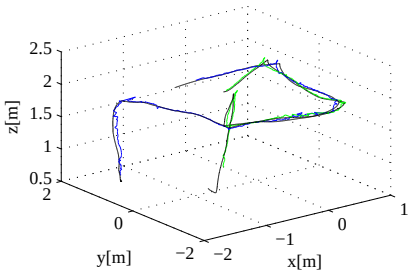


Figure 3.9: An Asctec Hummingbird equipped with an Intel i5, 1.8 GHz and a downward pointing ASUS Xtion.

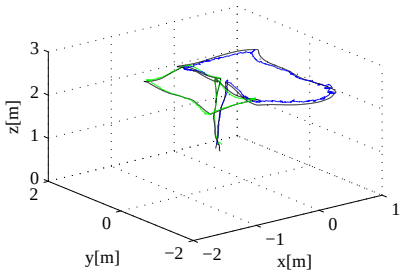
increase depth noise. The results confirm the applicability of the proposed localization algorithm and scale factor estimation. Even in the case of fast altitude changes, like in the third experiment and in the landing phase, the estimation of the scale factor is still reliable. As seen from the experimental results, the average RMSE error for both vehicles is about 0.07 m which makes it feasible for use for autonomous flight and for feedback control. In the second and third experiments, a higher error for the first vehicle can be noticed, which is the result of very quick motions from abrupt direction changes in the trajectory. The reader can easily notice the behavior in the attached video submission. Finally, in Fig. (3.11) trajectories and two different environment representations, provided by the user interface, are shown for the third experiment.

Table 3.3: RMSE of the ATE (m) for the RGB+Depth during flight mapping, estimating depth only at first frame (first column) and continuously (second column).

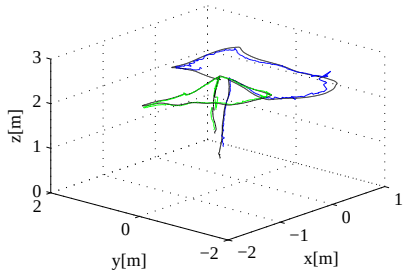
Experiment number	Vehicle num.	RGB+D 1 fr.	RGB+Depth
1	1	0.070783	0.057762
1	2	0.058007	0.048910
2	1	0.086844	0.084125
2	2	0.07817	0.056320
3	1	0.091035	0.080299
3	2	0.107347	0.086656
4	1	0.0710701	0.053526
4	2	0.073651	0.063978
avg.improvement		–	20%



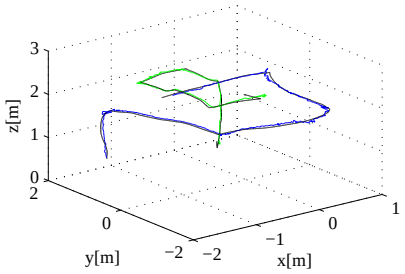
(a) In this experiment, the second vehicle starts and lands from a different position with respect to the first one, and it follows part of the first vehicle's trajectory.



(b) In this experiment, both vehicles start from the same position. The first one is mapping one side of the room while the second vehicle is mapping the opposite side.

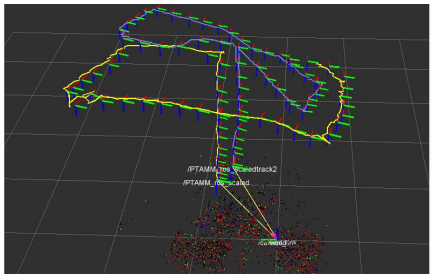


(c) The first vehicle is mapping one side of the room while the second one is mapping the opposite side with the additional difficulty that during its motion, the second vehicle quickly reduces its altitude by about 0.8 m and raises back up to 0.8 m before landing around 4 s.

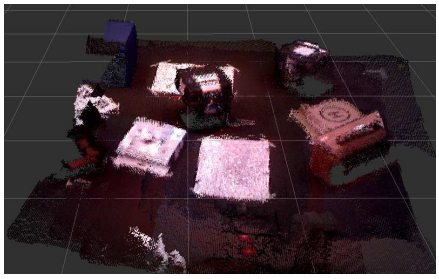


(d) The second vehicle starts when the first vehicle approaches the landing position. In this case, the relocalization acts in the part where the grid is not visible. This experiment demonstrates the effectiveness of the relocalizer and the scale recovery with the proposed sensor in a generic mapped environment.

Figure 3.10: Trajectories performed during the four experiments. First vehicle (blue line), corresponding ground truth (black line), second vehicle (green dashed line), and its ground truth (black dashed line). The average absolute error is 0.05 m except for the second and third experiments where a slight increased error is noticeable in the blue trajectory due to loss of vehicle control. Respect to the ground truth both vehicles present good performances estimating the scale in case of abrupt change in altitude.



(a) Trajectories, keyframes, and sparse map from two vehicles.



(b) Dense map made by one vehicle in previous experiments with $D_R = 2$, $D_C = 4$, $\Delta_{R,max} = 10$, $\Delta_{C,max} = 10$, $\Delta_{R,min} = 5$, $\Delta_{C,min} = 5$, $Z_{max} = 3$ m, $Z_{min} = 0.5$ m, $F_{max} = 20$ Hz, $F_{min} = 10$ Hz.

Figure 3.11: Trajectories and environment representations.

3.5 High Level Environment Representation

The maps previously described are an unsuitable representation of an environment to be managed by a high-level supervisory control. The real-time constraint of a flight control requires that the environment has to be represented with aggregate data to reduce the elaboration time, i.e., to maximize control system reactivity as mentioned in Section 1.2.4.

3.5.1 High level Architecture

Figure 3.12: Robotic Platform: ducted-fan ASV

Moreover, in the proposed scenario, beyond classical UAV tasks (take off, land, hovering, flyTo), the autonomous system should orchestrate a new set of operations like wall approach, docking, undocking, wall scanning etc.. These operations represent different operative modes, each associated with a different controller

with specific control laws and performance the high-level control system should be aware of. Each switch from one operative mode to the other should be suitably prepared and planned to keep smooth control trajectories. Since the system flies close to the obstacles in cluttered and unknown environments, fast planning engines are required to generate (or to adjust) trajectories in real-time. On the other hand, the system should be able to regulate the trade off between fast planning and accurateness of the generated trajectories depending on the operative mode and the context. The planning/executive system should be able to manage sliding autonomy, from autonomous to teleoperated mode, depending on the humans' interventions, since the system operates with the man in the loop.

The proposed approaches are validated by means of real experiments employing different platforms to interact and inspect the surrounding environment. A short overview of the proposed architecture is provided in the following. For more details, the reader can refer to [20, 22, 24]. In this context the aim is to present a general overview of the high level architecture emphasizing the role of visual environment representation and showing the experimental tasks that can be accomplished.

3.5.2 System Requirements and Architecture

The applicative scenario described so far requires a high-level control system with the following features:

- The air vehicle operates in close interaction with the environment, hence reactive, adaptive, and flexible planning/replanning capabilities are needed
- Both autonomous and human-in-the-loop control modalities should be supported to allow human interventions and teleoperation
- High-level control strategies should be defined taking into account the low-level operative modes and constraints

In particular, the high-level system should orchestrate the activations of a set of low-level controllers, modeled as hybrid automata [90], switching to the appropriate controller according to the operative mode and the task (see Figure 3.13) feeding the selected controller with suitable data (e.g. state and references).
The layered architecture depicted in Figure 3.13 is proposed to match these requirements. Two layers are distinguished: the high-level supervisory system is responsible for user interaction, task planning, path planning, execution monitoring; the low-level supervisory system manages the low-level execution of control primitives setting the controllers and providing control references. The architecture is detailed in Figure 3.14.

The robot activities are represented at different levels of abstraction: *mission-level tasks* representing mission goals; *macro-actions* representing primitive tasks

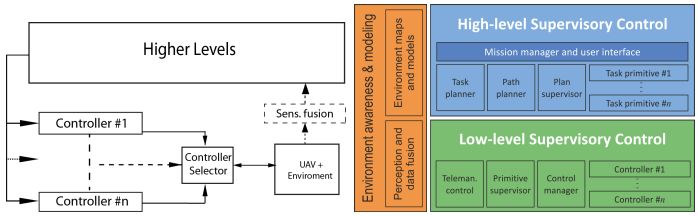


Figure 3.13: Interaction between the high level system and the low-level controllers (left); the high level control system is composed of high-level and low-level supervisory systems.

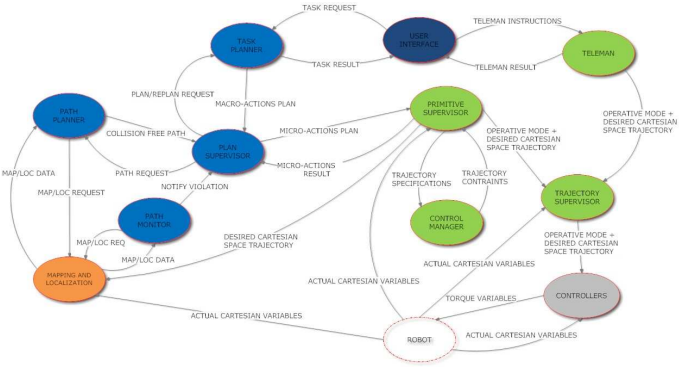


Figure 3.14: High Level Architecture: high level, low level, and reactive level modules are respectively in blue, green, and gray

(e.g. $TakeOff$). At a lower level of abstraction the set of commands (micro-actions) that can be sent to the low-level supervisory system, i.e. to the *Primitive Supervisory* (PR), is introduced. The *Task Planner* (TP) provides a plan composed of *macro-actions* (see Table 1). The *User* module (US) allows us to specify high-level goals (e.g. $Inspect(p)$) or lower level tasks (e.g. $TakeOff$) or to directly teleoperate. That is, the user can continuously interact with the system both by providing new high-level tasks/actions and by adjusting the low-level execution in a mixed-initiative control modality. Each task/goal is delivered to the TP which expands a task into an abstract plan composed of *macro-actions*. This plan is then sent to the *Plan Supervisor* (PS) for high-level execution. Each task or macro-action can be interrupted and preempted by new tasks provided by the user, provoking task replanning. That is, high-level mixed-initiative control is managed trough mixed-initiative planning [41].

The PS generates, for each macro-action in the high-level plan, a set of *micro-actions* to be executed by the PR. Each *macro-action* is further decomposed into a sequence of *micro-actions* which are endowed with detailed information about

$TakeOff(Pos)$	Take off from the current pose and hover in the pose Pos
$Land(Pos)$	Land from the current position to Pos
$Hover(Pos)$	Keep the pose Pos
$MoveTo(Pos)$	Move from the current pose to Pos
$MoveCircular(Pos, I)$	Circular movement around P with radius in I
$Scan(Srf)$	Scan the surface Srf
$Inspect(Obj, P)$	Observe the object Obj in position P
$Brake(C)$	Execute a hard brake from the current position
$Approach(P)$	Approach the target position P
$Dock(P)$	Dock to a target position P
$UnDock(C)$	Undock from the current position
$Manipulate(Obj, P)$	Manipulate an object Obj in position P

Table 3.4: Macro actions considered in the operative domain.

the associated geometrical paths. The PR exploits the *Control Manager* (CM) to select the low-level controller responsible for the micro-action execution. This module is the main responsible for the high-level/low-level control integration: given the operative constraints provided by the high-level supervisor and given the low-level controller features, the CM is to decide the best controller for the execution. Finally, the PR generates the control trajectory passing it to the *Trajectory Supervisor* (TS) to generate control references at a suitable frequency.

The PR exploits concatenations of fifth-order polynomials to provide smooth trajectories between waypoints [76] while ensuring the velocity and tolerance constraints as explained further below in this section. Depending on the required reactivity, the PS regulates the number of geometric waypoints to be processed by the PR. When a micro-action fails, the PS can either call the PP to generate an alternative path or call the TP to generate a different plan of macro-actions. Furthermore, it can be interrupted by the *Path Monitor* (PM) which checks for trajectory deviations and unexpected obstacles. Finally, the operator can always switch to a manual control mode, in this case the TS should monitor the trajectory provided by the *Teleman*. Once the autonomous control is restored, a replanning process is needed to recover the execution of the current task.

3.5.3 3D Mapping

The data environment for mapping and path planning is a 3D occupancy grid-map of cells which is run-time generated given the robot pose and the 3D point clouds extracted from the vision system. The pose estimation, which is a fundamental pre-requisite to generate the high-level map for the high-level control architecture, is recovered differently according to the type of platform used, as it will be better explained in next sections. Given the pose, the associated point cloud map should be suitably processed into a 3D occupancy grid. This is obtained by discretizing the vehicle's workspace with elementary cubes of equal size. A vehicle of $50 \times 50 \times 20$ cm is employed, and cubes of 10 cm are used. The number of points into a cube is

a direct measure of the probability that an object is located in that region. Two goals are simultaneously achieved:

- the dimension of data that the supervisory control has to elaborate online remains limited
- the presence of outliers do not affect the reliability of the system, thanks to the adoption of suitable thresholds for the occupancy detection

For each cube it is stored: the number of inliers (3D triangulated points) fell into the cube volume, the last camera position which an inlier had been collected, and the state of the cube. The number of inliers represents the number of different points from which the same obstacle has been detected. Each cell can be associated with one of the following values: $free$, *occupied*, *obstacle*, *target*, *ignored* or *unknown*. Initially, each cube is set to $free$. When a 3D point is detected to belong to a given cube, the value of the corresponding cube is set to *occupied*. When the number of points inside a cube reaches a given threshold, the state is set to *obstacle*. On the other hand, when a target is identified, the corresponding cube is set to *target*. Moreover, from each position that had generated a valid target view point, all the cubes laying along the optical rays are set to *ignored*.

For wide environments, a sparse representation of the occupancy grid map is associated with a spatial/temporal vanishing criterion. This determines whether an occupancy cube is still reliable or if it has to be discarded (depending on the distance traveled by the vehicle and/or on the time last after its previous update). In fact, due to the drift of the vehicle pose estimation, obstacles which have been observed a long time before or far from the current position cannot be considered reliable anymore in the current map representation, therefore they should be refreshed. With these solutions the reliability and scalability of the map representation can be suitably tuned.

3.6 Case Studies

In this section, experimental results on planning, replanning, and obstacle avoidance, both in real-world scenarios and in simulated environments are presented. In particular, the system at work is described in two case studies representing, respectively, a physical inspection task, where docking and manipulation activities are necessary, and a visual inspection task, where a scan of an unknown wall is deployed avoiding a physical obstacle(s) placed on the wall with unknown size, number, and position.

3.6.1 Experimental Setting

In order to validate the presented architecture have been deployed in two different scenarios. In the first scenario two experiments are introduced. In the former, a

single-module ducted-fan vehicle [79] should avoid an unknown obstacle to perform the inspection task, which consists in keeping a fixed the distance from the unknown wall while a superficial scanning path is executed. In the latter, two connected ducted-fan modules, which have been connected to increase the overall vehicle's payload and controllability along the approaching direction, should move towards a wall, dock on it, and perform a manipulation task (e.g. writing some worlds on the wall). For this goal, a small delta-parallel manipulator is mounted on the dual-module version of the vehicle to enable docking and interaction operations.

Both the vehicles are equipped with a visual sensor and an ATOM board 1.6GHz, which gives the possibility to stream the data on an INTEL Core i7 platform, 1.6GHz, 4GB RAM, UBUNTU 10.04. This architecture enables us to stream compressed images on a ground station at 15Hz or elaborate all onboard at 5Hz.

In the second scenario a quadrotor platform from ETH Zurich [92] is employed to perform the visual inspection of an industrial-boiler texturized wall with the added difficulty to avoid a unknown obstacle placed on it. The platform is equipped with a stereo camera and embedded IMU. An onboard FPGA module enables the system to speed up some basic operations like feature extractions and sensor fusion. The remaining operations are executed on a ground station INTEL Core i5 platform 2.8GHz, 4GB RAM, Ubuntu 10.04.

In both scenarios, the low-level and the high-level supervisors are connected with the robotic platform exploiting the ROS framework.

3.6.2 Obstacle Avoidance and Interaction with the Environment

Two scenarios are considered. In the first scenario, planning, replanning on-the-fly, and obstacle avoidance are introduced, while in the second scenario physical interaction with the environment (i.e. docking/undocking and manipulation) is tested. Notice that, for each of these scenarios, take off and landing are manually managed, hence the operator switches from teleoperation to autonomous mode and viceversa to, respectively, start and end the mission. Moreover, the vision plays in this tasks a fundamental role in all the previus mentioned scenarios to identify obstacles, provide information on the surface to inspect and interact.

Obstacle Avoidance The architecture has been tested in a real environment of $4 \times 4 \times 3$ m sizeconsidering the two environments depicted in Figure 3.15 (up and down). The base reference frame is located in the bottom-left part of the environment depicted in the Figure 3.15. In the two testing scenarios, the task was to inspect a target point from the initial pose. In the first test the target point is at $(250, 100, 75)$ cm, with 90 deg of orientation, from the pose $(25, 100, 50)$ cm,

with 0 deg of orientation; in the second test the target point is at $(250, 325, 75)$ cm, with 90 deg of orientation, from the pose $(25, 70, 50)$ cm, with 0 deg of orientation. The maximum speed was set to 0.3 m/s for the two tests. For each test, an unknown obstacles is placed along the initial planned path, but not visible due to the initial orientation of the camera. In this way, the obstacles can be detected during the motion provoking task/path replanning, escape, or brake, depending on the vehicle state (distance from the obstacles and current velocity).

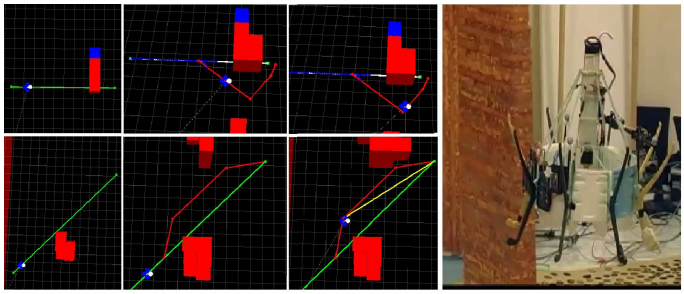


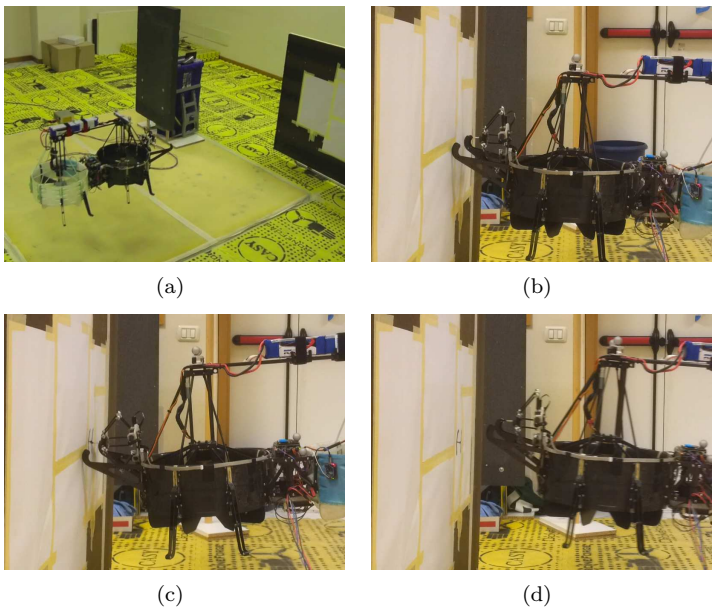
Figure 3.15: Planning and replanning: initially, the system generates the green path, once the obstacle is detected a new path is generated on the fly (left); shot of the real platform during the plan execution (right).

For each scenario, each test is exectued 10 times collecting mean, max, min, and standard deviation (STD) of: time spent during planning (Tp), time spent in replanning (Tr), number of replanning episodes (Nr), length of the executed path (Lp), and total time for execution (including replanning time) (Te).

Table 2 reports the results for the two scenarios (Test 1 and Test 2). For both these settings, initially, the obstacles are not visible, hence the generated plan is simple and planning time is low (see the left side of Figure 3.15). The control pose feedback of the vehicle is either obtained by using LIBVISIO2 [47, 38] coupled with a Kalman filter or, alternatively, by directly deploying an OptiTrack [5] motion capture system. Once the obstacles are discovered on the fly, replanning is invoked to adjust on-line the trajectory (i.e. without hovering during the replanning phase). Replanning and execution time are slightly higher in the first scenario which is more complex. Instead, Tr seems negligible when compared with Te. The final trajectory length (Te) is similar in both the settings and comparable with the distance between the starting and target point, hence the final trajectory seems not affected by the continuous replanning process. In these tests, Tp and Tr are mainly due to path and trajectory planning (while task decomposition is negligible). During these tests, experienced brake or escape episodes were never happened.

	Test 1				Test 2			
	Mean	**STD**	**Max**	**Min**	**Mean**	**STD**	**Max**	**Min**
Tp	0.075	0.014	0.08	0.04	0.017	0.002	0.03	0.01
Tr	0.614	0.41	1.20	0.01	0.067	0.04	0.11	0.005
Te	60.5	10.12	75	42	49.9	8.18	60	40
Lp	14.4	1.54	18	12	13.18	1.11	15	11

Table 3.5: Planning and execution results (in seconds) in the real scenario.



(a) (b) (c) (d)

Figure 3.16: Physical inspection: (a) the robot flies towards the target; (b) docking maneuver; (c) manipulation; (d) undocking maneuver.

Interaction with the Environment In the second experiment, the architecture has been tested in docking and manipulation maneuvers. In this setting, the goal is to execute a simple manipulation (i.e. write a couple of letters) on a target wall located in a given position. The initial experimental setting is depicted in Figure 3.17. Here, the target wall is positioned at (150,180, 50) cm at 0 deg of orientation from the robot initial pose. Also in this case, the maximum speed is set to 0.3 m/s. The experiments starts with a teleoperated take-off, then the autonomous mode is enabled and the user provides the robot with a manipulation task associated with the relative position of the target panel. The abstract manipulation task is decomposed into a sequence of macro actions: $MoveTo(Approach)$, $Dock(Target)$, $Manipulate()$, $Undock(Target)$ to be autonomously executed and

monitored by the PS. The $MoveTo(Approach)$ action should bring the vehicle at a close distance form the target wall to enable the docking maneuver $Dock(Target)$. Initially, since some obstacles are not visible, the approach operation is associated with a simple first path and trajectory. This planning step is fast (0.05 seconds), however, as soon as the environment has been reconstructed by the mapping process a new path is needed to avoid the obstacles and this is on-line generated by adjusting the first path (see Figure 3.16(a)). In this setting, during our tests, at least one replanning steps is needed to adjust the trajectory (with Tr aligned with the one in Table 1 and 2). Once the approach position has been reached (this phase takes about 25 seconds in our tests), the path monitor communicates that the approach position has been reached, hence the PS can start the docking operation. This maneuver is managed by a specialized controller till the wall contact is not reached and stabilized (see Figure 3.16(b)). Once the vehicle reaches a stable contact with the vertical surface of the panel, the manipulation task can be executed by the PS. In this case the manipulator was endowed with a pen and the task was to write "Hi" on the docking panel. Finally, after the $Manipulate()$ action, the PS can switch to the undocking phase (see Figure 3.16(d)). At this point the mission has been accomplished, then the operator can finally switch to the teleoperated mode to land.

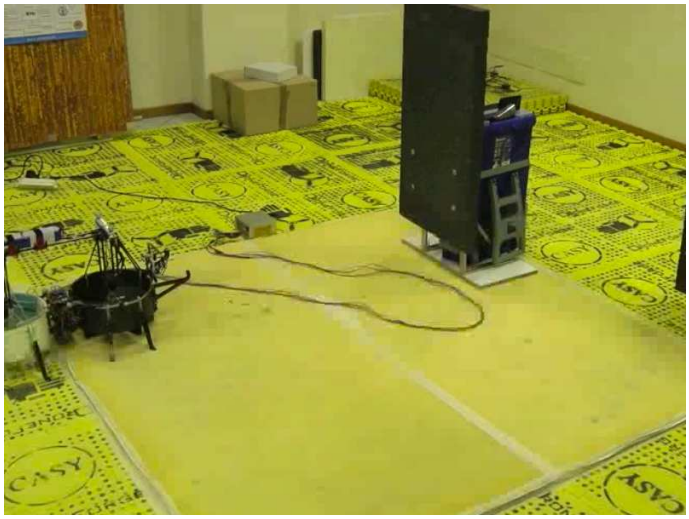
This scenario illustrates the strict connection needed between different control layers and the smooth control switching behavior between different operative modes. Overall, the system task/path planning/replanning performance shows to be compatible with the operative scenario requirements.

3.6.3 Visual Inspection

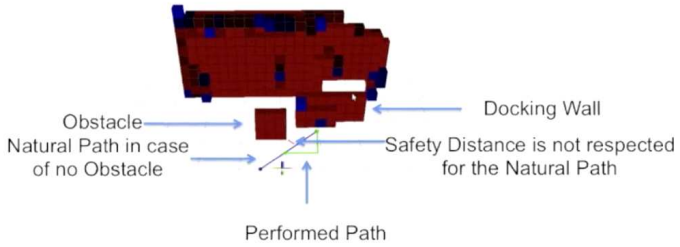
A second testing scenario concerns a visual inspection task (see Figure 3.18). In this case the system is to detect a wall in the environment and then generate an inspection path that allows to scan the surface of the wall. The inspection task exploits a fast incremental clustering algorithm of the 3D point cloud sequence provided by the stereo vision system [38].

The inspection task is based on three scanning steps. In the first stage, a preliminary, low-resolution, scan is performed by observing the wall that must be inspected from a far position (see Figure 3.19(a)). This preliminary phase allow us to detect a possible wall surface obtaining a rough shape and extension of the wall. This approximate shape of the a wall is obtained once a sufficient number of points are collected by the mapping process described in Section 3.5.3. That is, the 3D grid-map of cells, acquired during this first step, is employed to estimate the wall pose with respect to the vehicle.

Once the approximate position of the wall has been defined, the PP can generate a first inspection path that allows us to gather a more detailed and accurate scan of the observed structure. The planned trajectory is a serpentine that is



(a)



(b)

Figure 3.17: Physical inspection: environment and robot initial position (a), robot position and target wall (b).

shaped with respect to the estimated wall. During this second scan action, the wall structure, shape, position, and size are updated, as shown in Figure 3.19(b). Moreover, obstacles which are present on the wall are detected by using an affinity test with respect to the current statistics of the estimated wall.

Finally, a third planning step is required to generate the inspection trajectory that keeps a constant distance with respect to wall avoiding obstacles which have been detected during the previous step or during the motion (a replanning action will be activated in this case).

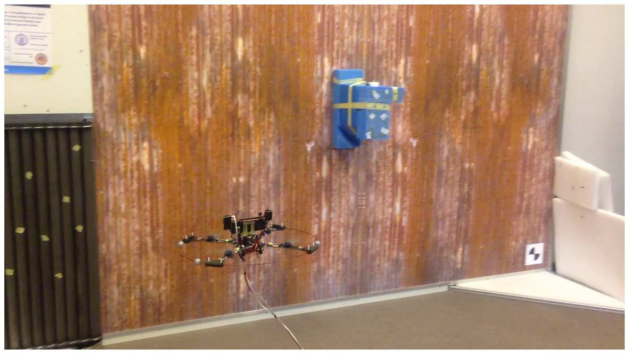
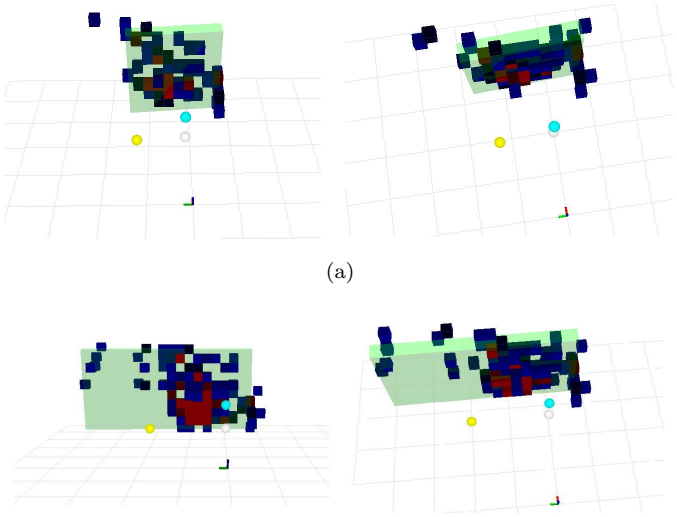


Figure 3.18: Wall to be inspected w5th a boiler-like texture.



(a)

(b)

Figure 3.19: Main phases of the inspection task.

Different experiments at different speeds in the range $0.1 - 0.4\ m/s$ have been performed with a full success ratio. The goal was the scan of a $2.5 \times 1.5\ m^2$ surface with path steps of $0.5\ m$ distant in the first inspection phase and $0.25\ m$ in the second inspection phase from the wall (see Figure 3.18). Figure 3.20 shows two different trajectories generated during the performed experiments.

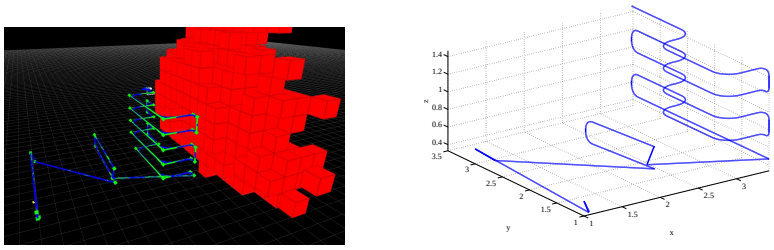


Figure 3.20: Paths performed during inspection task experiments of the environment shown in Figure 3.18. On the left: shot of the path planning interface where both the performed (blue) and the planned path (green) are shown. On the right: executed trajectory with metric scale indication.

Res/Env	LL		LH		HL		HH	
	Mean	**STD**	**Mean**	**STD**	**Mean**	**STD**	**Mean**	**STD**
Tp	0.21	0.11	0.39	0.03	0.25	0.10	0.31	0.14
Tr	0.12	0.03	0.07	0.01	0.20	0.04	0.23	0.03
Te	308.39	3.1	211.88	2.4	718.57	5.2	720.45	7.6
Lp	79.09	13.76	78.04	9.63	86.79	12.65	85.24	13.12
Nr	0.9	0.21	0.3	0.12	3.4	1.71	2.5	1.10

Table 3.6: Planning and execution results(time in seconds, length in meters)

3.6.4 Planning and execution

The planning and execution system has been tested in simulated environments. To test continuous replanning, a larger space of dimension $100 \times 100 \times 50\ m^3$ is considered with 4 and 9 obstacles. To decouple replanning from map building, a known map associated with a visibility horizon (not visible obstacles are detected on the fly causing replanning), is assumed. For each test, the task was to inspect a target point in pose $(90, 90, 5, 90)$ starting from *hovering* in the pose $(5, 5, 5, 0)$ (in meters); the robot maximum and minimum velocity was set at $0.5\,m/s$ and $0.1\,m/s$ respectively. By changing the visibility horizon (green cells in Fig. 3.15) of the planner (15 or 25 m) and the complexity of the environment (4 or 9 obstacles) 4 scenarios were obtained. Table 2 collects means and STD of 10 tests for each entry (time and length are in *sec.* and m, LL, HL, etc. are for Low complexity and Low visibility, High complexity and Low visibility). Here, it is noticeable that Tp increases with the obstacles (HL,HH) and decreases with short visibility (LL,HL). Indeed, in these cases the planning problem is simpler. However, short visibility is associated with additional replanning time which, in turn, decreases with the number of obstacles. The lower the replanning time, the lower is the execution time and the shorter the executed path. A similar effect is due to visibility: short visibility causes frequent replanning events (Nr) and longer paths

(Lp) and execution times (Te). Furthermore, the variance is enhanced with short visibility that enhances the uncertainty. In these tests, the task planning time is usually negligible (Tp and Tr mainly due to path and trajectory). Also in this case, brakes or escapes were never experienced.

Chapter 4

Sensor Fusion of Visual and Inertial Measurements

In this chapter, a new optimal sensor fusion algorithm based on Pareto optimization techniques is proposed to combine IMU and camera visual measurements to estimate a vehicle motion. Advantages of the proposed method are that no-prior assumption about robot motion model is required, and that the proposed formulation allows a multi-rate sensor fusion. A comparison of the proposed technique with respect to a Kalman filter approach shows an improved estimation at the price of a limited increased computational complexity.

4.1 Problem Formulation

It is supposed that the orientation of the camera reference frame with respect to the inertial unit frame is known. Without loss of generality, the base reference frame is assumed coincident with the first IMU frame. The differential vision system provides the position displacement with respect to the previous vehicle position, i.e., only a differential positional measurement is available, with a sampling rate T_V. On the other hand, the IMU provides the linear acceleration and attitude of the vehicle with a sampling time $T_i \leq T_V$. The latter can be recovered by fusing the accelerometer and rate gyro measurements in a standard complementary attitude filter, similarly to what presented [12], by using a gradient descent method as in [77]. The orientation of the vehicle is represented with the well-known Tait-Bryan (Euler) angles of roll, pitch, and yaw $\boldsymbol{\phi} = (\varphi, \theta, \psi)$, which yields to the rotation matrix $\boldsymbol{R}$, i.e.

$$\boldsymbol{R}(\boldsymbol{\phi}) = \begin{bmatrix} c_\varphi c_\theta & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & s_\varphi s_\psi + c_\varphi s_\theta c_\psi \\ s_\varphi c_\theta & s_\varphi s_\theta s_\psi + c_\varphi c_\psi & s_\varphi s_\theta c_\psi - c_\varphi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix},$$

where $s_x = \sin(x)$ and $c_x = \cos(x)$. Moreover, the measurement of the Euler angles

$$\tilde{\boldsymbol{\phi}} = \left(\tilde{\varphi}, \tilde{\theta}, \tilde{\psi}\right) \tag{4.1}$$

can be modeled as

$$\begin{aligned} \tilde{\varphi} &= \varphi + \omega_\varphi \\ \tilde{\theta} &= \theta + \omega_\theta \\ \tilde{\psi} &= \psi + \omega_\psi, \end{aligned} \tag{4.2}$$

where ω_φ, ω_θ, and ω_ψ represent angle Gaussian white noises with zero mean and variance σ_ϕ^2, σ_θ^2, and σ_ψ^2, respectively.

Finally, the linear acceleration of the vehicle $\tilde{\boldsymbol{a}}$ with respect to the fixed frame is given by

$$\tilde{\boldsymbol{a}} = \boldsymbol{R}(\tilde{\boldsymbol{\phi}})\tilde{\boldsymbol{a}}_I = \boldsymbol{R}(\tilde{\boldsymbol{\phi}})\left(\boldsymbol{a}_I + \boldsymbol{\omega}_{a_i}\right) - \boldsymbol{g}, \tag{4.3}$$

where $\boldsymbol{a}_I = [\ a_{I,x} \quad a_{I,y} \quad a_{I,z}\ ]^{\mathrm{T}}$ is the acceleration provided by the IMU and expressed in the current robot frame, $\boldsymbol{\omega}_{a_I} = [\ \omega_{a_I,x} \quad \omega_{a_I,y} \quad \omega_{a_I,z}\ ]^{\mathrm{T}}$ is a Gaussian white noise with known variance $\boldsymbol{\sigma}^2_{a_I} = [\ \sigma^2_{a_I,x} \quad \sigma^2_{a_I,y} \quad \sigma^2_{a_I,z}\ ]^{\mathrm{T}}$ and $\boldsymbol{g} = [\ 0 \quad 0 \quad 9.81\ ]^{\mathrm{T}}$ is the gravity vector that can be subtracted from the inertial measurement given that the absolute orientation is known.

The measurement $\widetilde{\delta \boldsymbol{p}}_V = \delta \boldsymbol{p} + \boldsymbol{\omega}_V$, that is provided by the vision system, represents the robot displacement performed during the last sampling period T_V, expressed in the fixed frame with respect to last visual frame, where $\delta \boldsymbol{p}$ is the effective unknown displacement and $\boldsymbol{\omega}_V = [\ \omega_{V,x} \quad \omega_{V,y} \quad \omega_{V,z}\ ]^{\mathrm{T}}$ is white noise with known variance $\boldsymbol{\sigma}^2_V = [\ \sigma^2_{V,x} \quad \sigma^2_{V,y} \quad \sigma^2_{V,z}\ ]^{\mathrm{T}}$ and bias $\mathbf{b_V} = [\ b_{V_x} \quad b_{V_y} \quad b_{V_z}\ ]^{\mathrm{T}}$. Well-known visual odometry techniques (e.g. see [97] and [44]) can be considered to compute $\widetilde{\delta \boldsymbol{p}}_V$.

With the proposed approach, the estimation of the vehicle displacement $\widehat{\delta \boldsymbol{p}} = [\ \widehat{\delta x} \quad \widehat{\delta y} \quad \widehat{\delta z}\ ]^{\mathrm{T}}$, as performed between the sampling times $k-1$ and k, is evaluated by a convex combination of the visual measurement and the inertial displacement estimation as:

$$\widehat{\delta x}(k) = (1 - \beta_{x,k})\widetilde{\delta x}_V(k) + \beta_{x,k}\widehat{\delta x}_a(k) \tag{4.4}$$

$$\widehat{\delta y}(k) = (1 - \beta_{y,k})\widetilde{\delta y}_V(k) + \beta_{y,k}\widehat{\delta y}_a(k) \tag{4.5}$$

$$\widehat{\delta z}(k) = (1 - \beta_{z,k})\widetilde{\delta z}_V(k) + \beta_{z,k}\widehat{\delta z}_a(k), \tag{4.6}$$

The term $\widehat{\delta \boldsymbol{p}}_a = [\ \widehat{\delta x}_a \quad \widehat{\delta y}_a \quad \widehat{\delta z}_a\ ]^{\mathrm{T}}$ is the estimation of the position displacement obtained from the inertial data, which will be characterized both in the synchronous and asynchronous case in the following sections. The weight factors $\beta_{x,k}$, $\beta_{y,k}$, and $\beta_{z,k}$ are the unknown parameters that have to be (optimally) chosen at each sampling time by a Pareto optimization, as described in the following.

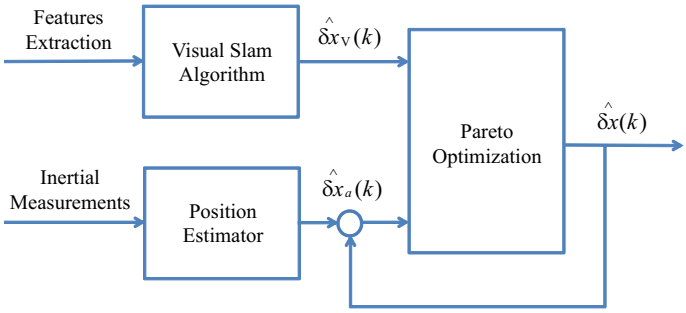


Figure 4.1: Working schema of the proposed Pareto optimization algorithm.

Accordingly, the absolute position estimation of the vehicle at the time instant k can be computed as

$$\widehat{\boldsymbol{p}}(k) = \widehat{\boldsymbol{p}}(k-N) + \widehat{\delta \boldsymbol{p}}(k), \tag{4.7}$$

where $N = 1$ in case of synchronous measurements.

Without loss of generality, only the estimation of the x component will be described. Analogous results can be achieved for the y and z motion components.

Figure 4.1 shows the working principle of the sensor fusion algorithm valid both in synchronous and asynchronous measurements cases. The difference between two cases relies on the correction on inertial position estimation that, as it will shown in next subsections, in the synchronous case is done at every step time, while in the asynchronous case only where there is visual measurement availability.

4.1.1 Synchronous measurements

Consider the case of synchronous measurements, $T = T_V = T_i$. Then the IMU measurements are used at the same frequency of the vision system. By starting from the inertial measurements, the estimation of the positional displacement by a forward Euler integration is obtained

$$\widehat{v}_x(k) = \widehat{v}_x(k-1) + T\tilde{a}_x(k) \tag{4.8}$$

$$\widehat{\delta x_a}(k) = T\widehat{v}_x(k). \tag{4.9}$$

By plugging (4.8) into (4.9) and approximating $\widehat{v}_x(k-1) \cong \widehat{\delta x}(k-1)/T$, the estimation of the position displacement obtained by the inertial measurement is

$$\widehat{\delta x_a}(k) = \widehat{\delta x}(k-1) + T^2\tilde{a}_x(k). \tag{4.10}$$

Thus, the estimated position depends on the estimated position at time $k-1$ and on the acceleration measurement.

4.1.2 Asynchronous measurements

In the asynchronous case, it is assumed that the vision system provides the vehicle pose estimation at lower frequency with respect to the inertial system, $T_V = NT_i$, with $N \in \mathbb{N}$ being the scale factor relating the IMU and vision frequencies.
Exploiting classical Taylor expansion the position at time k can be written as a function of the acceleration a_x at time $k-1$ as

$$x(k) = x(k-1) + T_i \delta v_x\,(k-1) + \frac{T_i^2}{2} a_x(k-1), \tag{4.11}$$

where δv_x represents the velocity at instant time $k-1$.

Thus, the absolute position estimation obtained by the inertial sensor at time instant k can be expressed as a function of the optimal position estimation at time $k-N$ as

$$\widehat{x}_a(k) = \widehat{x}(k-N) + T_i \sum_{j=k-N}^{k-1} \widehat{\delta v}_x\,(j) + \frac{T_i^2}{2} \sum_{j=k-N}^{k-1} a_x(j), \tag{4.12}$$

with the velocity estimation $\widehat{\delta v}_x$ given by Euler integration formula

$$\widehat{\delta v}_x\,(j) = \widehat{\delta v}_x(j-1) + T_i \tilde{a}_x(j-1)$$

The differential displacement from inertial measurement between two consecutive optimization time instants is given subtracting $\widehat{x}_a(k-N)$ to Eq. (4.12) obtaining:

$$\begin{aligned} \widehat{\delta x}_a(k) =& \widehat{\delta x}(k-N) + T_i \sum_{j=k-N}^{k-1} \widehat{\delta v}_x\,(j) + \frac{T_i^2}{2} \sum_{j=k-N}^{k-1} a_x(j) \\ & - T_i \sum_{j=k-2N}^{k-N-1} \widehat{\delta v}_x\,(j) - \frac{T_i^2}{2} \sum_{j=k-2N}^{k-N-1} a_x(j), \end{aligned} \tag{4.13}$$

The state estimation $\widehat{\delta x}(k)$ will be performed as soon the visual measurement is available, according to (4.4).

On the other hand, the differential position estimation provided by the vision system is assumed coincident with the measurement itself

$$\widetilde{\delta x}_V(k) = \widehat{\delta x}_V(k) = \delta x_V(k) + \omega_{x_V}(k), \tag{4.14}$$

where $\delta x_V(k)$ is the ground-truth position displacement.

4.1.3 Pareto optimization problem

The estimator (4.4) can be modeled as follows

$$\widehat{\delta x}(k) = \delta x(k) + \omega_x(k), \tag{4.15}$$

where $\omega_x(k)$ is the error position, which is estimated as described in equation (4.3). The estimator bias is denoted by

$$P_1 = \mathbb{E}\{\omega_x(k)\}, \tag{4.16}$$

and the estimation variance is defined by

$$P_2 = \mathbb{E}\{(\widehat{\delta x}(k) - \mathbb{E}\{\widehat{\delta x}(k)\})^2\}, \tag{4.17}$$

where the operator $\mathbb{E}\{\cdot\}$ is the expected value of a random variable.
Then a Pareto optimization problem can be posed as

$$\begin{aligned} &\min_{\beta_{x,k}} (1-\rho_{x,k})\, P_2 + \rho_{x,k} P_1^2 \\ &\text{s.t. } \beta_{x,k} \in (-1,1), \end{aligned} \tag{4.18}$$

With this choice, the bias and the variance of the estimation error will be minimized simultaneously. The Pareto weighting factor $\rho_{x,k}$ has to be chosen at each step so to trade-off the high variance and bias of sensors. The constraint on $\beta_{x,k}$ is required because the bias may become unstable with time when the statistical modeling of the bias is computed, as shown later. The solution of the optimization problem requires first the evaluation of the quantities P_1 and P_2, that will be discussed in the following sections.

4.2 Error Estimation Bias and Variance

In this section an analytical recursive expression for equations (4.16) and (4.17) is provided. The computation of the bias and of the variance will be differentiated according to the types of signals to be fused and the sensor timing condition, i.e. synchronous/asynchronous cases.

4.2.1 Synchronous measurements

By considering (4.4) and (4.15), the error $\omega_x(k)$ can be expressed as follow

$$\begin{aligned} \omega_x(k) &= (1-\beta_{x,k})\omega_V(k) + \beta_{x,k}\left(\widehat{\delta x}_a(k) - \delta x_a(k)\right) \\ &= (1-\beta_{x,k})\omega_V(k) + \beta_{x,k}\left(\omega_x(k-1) + T^2\omega_{a_x}(k-1)\right). \end{aligned} \tag{4.19}$$

Consequently, (4.16) can be rewritten as

$$\begin{aligned} P_1 =& \mathbb{E}\{\omega_x(k)\} = (1-\beta_{x,k})\mathbb{E}\{\omega_{V_x}(k)\} + \beta_{x,k}\mathbb{E}\{\omega_x(k-1))\} + \\ & \beta_{x,k}T^2\mathbb{E}\{\omega_{a_x}(k-1)\}, \end{aligned} \tag{4.20}$$

where

$$\begin{aligned}
&\mathbb{E}\{\omega_{a_x}(k)\} = a_{I,x}(k)c_\varphi c_\theta e^{\frac{-\sigma_\varphi^2}{2}} e^{\frac{-\sigma_\theta^2}{2}} + a_{I,y}(k) \\
&(c_\varphi s_\theta s_\psi e^{\frac{-\sigma_\varphi^2}{2}} e^{\frac{-\sigma_\theta^2}{2}} e^{\frac{-\sigma_\psi^2}{2}} - s_\varphi c_\psi e^{\frac{-\sigma_\varphi^2}{2}} e^{\frac{-\sigma_\psi^2}{2}}) + a_{I,z}(k) \\
&(s_\varphi s_\psi e^{\frac{-\sigma_\varphi^2}{2}} e^{\frac{-\sigma_\psi^2}{2}} + c_\varphi s_\theta c_\psi e^{\frac{-\sigma_\varphi^2}{2}} e^{\frac{-\sigma_\theta^2}{2}} e^{\frac{-\sigma_\psi^2}{2}}) - a_{I,x}(k)c_\varphi c_\theta \\
&- a_{I,y}(k)(c_\varphi s_\theta s_\psi - s_\varphi c_\psi) - a_{I,z}(k)(s_\varphi s_\psi + c_\varphi s_\theta c_\psi).
\end{aligned}$$

The term $\mathbb{E}\{\omega_{a_x}(k)\}$ has been derived by using the following result [8, 9]

$$\begin{aligned}
\mathbb{E}\{\widetilde{a}(k)\cos(\widetilde{\varphi}(k))\} &= a(k)\cos(\varphi(k))e^{\frac{-\sigma_\varphi^2}{2}} \\
\mathbb{E}\{\widetilde{a}(k)\sin(\widetilde{\varphi}(k))\} &= a(k)\sin(\varphi(k))e^{\frac{-\sigma_\varphi^2}{2}},
\end{aligned}$$

where $\tilde{\boldsymbol{\phi}}(k)$ has been considered Gaussian and statistically independent in its own components and with respect to the acceleration measurements $\boldsymbol{a}_I$. The proof for the sine term is provided in the Lemma A.2.1.

In the proposed formulation the bias $\mathbf{b_{a_I}} = [\ b_{a_{I,x}} \quad b_{a_{I,y}} \quad b_{a_{I,z}}\ ]^{\mathrm{T}}$ on the single acceleration measurement has been neglected for simplicity. In fact, since it is generally constant, it can be estimated and subtracted from the measurement itself, as done in [61]. An alternative approach can be to keep the bias in the measurement and consider it as a penalty in P_1 for the acceleration pose estimation, thus obtaining a formulation similar to the one presented above.

Notice that (4.20) is a discrete time recursive expression, where the value of velocity bias at time k is related to that one at time $k-1$ trough $\beta_{x,k}$ coefficient. This can be directly interpreted as a discrete time differential equation. To avoid a blow up of the bias, all the eigenvalues should be in the circle of radius 1. Hence, it is necessary that $|\beta_{x,k}| \in (0,1)$ and thus $\beta_{x,k} \in (-1,1)$ so as it has been required in (4.18).

By substituting (4.20) in (4.17), the quantity P_2 can be rewritten as

$$\begin{aligned}
P_2 =& \mathbb{E}\{(\widehat{\delta x}(k) - \mathbb{E}\{\widehat{\delta x}(k)\})^2\} \\
=& \mathbb{E}\left\{((1-\beta_{x,k})v_{V_x}(k) + \beta_{x,k}v_x(k-1) \ \ + \beta_{x,k}v_{a_x}(k-1))^2\right\},
\end{aligned} \tag{4.21}$$

where

$$\begin{aligned}
v_{V_x}(k) &\triangleq \omega_{V_x}(k) - \mathbb{E}\{\omega_{V_x}(k)\} \\
v_x(k) &\triangleq \omega_x(k) - \mathbb{E}\{\omega_x(k)\} \\
v_{a_x}(k) &\triangleq \omega_{a_x}(k) - \mathbb{E}\{\omega_{a_x}(k)\} \\
v_{v_x}(k) &\triangleq \omega_{v_x}(k) - \mathbb{E}\{\omega_{v_x}(k)\},
\end{aligned} \tag{4.22}$$

with $\sigma_{V_x}^2$, σ_x^2, $\sigma_{a_x}^2$ and $\sigma_{v_x}^2$ the corresponding variances. Since v_{V_x}, v_x, v_{a_x} and v_{v_x} are statistically independent and $\mathbb{E}\{v_{a_x}\} = \mathbb{E}\{v_x\} = 0$, (4.21) yields

$$P_2 = (1-\beta_{x,k})^2\sigma_{V_x}^2(k) + \beta_{x,k}^2\sigma_x^2(k-1) + \beta_{x,k}^2T^4\sigma_{a_x}^2(k-1), \tag{4.23}$$

where

$$\begin{aligned}\sigma^2_{a_x}(k) =& \mathbb{E}\{(\omega_{a_x}(k) - \mathbb{E}\{\omega_{a_x}(k)\})^2\} = \mathbb{E}\{\omega^2_{a_x}(k) + \\ & \mathbb{E}^2\{\omega_{a_x}(k)\} - 2\omega_{a_x}(k)\,\mathbb{E}\{\omega_{a_x}(k)\}\} \\ =& \mathbb{E}\{\omega^2_{a_x}(k)\} - \mathbb{E}^2\{\omega_{a_x}\}.\end{aligned}$$

The second order moment $\mathbb{E}\{\omega^2_{a_x}(k)\}$ could be characterized using the following properties [8, 9]:

$$\begin{aligned}\mathbb{E}\{\widetilde{a}^2(k)\cos^2(\widetilde{\varphi}(k))\} &= \sigma_a^2\left(\frac{1}{2} + \frac{1}{2}c_{2\varphi}e^{-2\sigma^2_\varphi}\right) \\ \mathbb{E}\{\widetilde{a}^2(k)\sin^2(\widetilde{\varphi}(k))\} &= \sigma_a^2\left(\frac{1}{2} - \frac{1}{2}c_{2\varphi}e^{-2\sigma^2_\varphi}\right),\end{aligned}$$

where σ_a^2 is the acceleration variance. The proof for the sine term is provided in the Lemma A.2.2.

4.2.2 Asynchronous measurements

By considering equation (4.4), (4.13), and (4.15) the quantity $\omega_x(k)$ can be expressed as follows

$$\omega_x(k) = (1 - \beta_{x,k})\omega_V(k) + \beta_{x,k}\omega_{x_a}(k), \tag{4.24}$$

where $\omega_{x_a}(k)$ represents the position differential error of the inertial measurements in (4.13). Hence, (4.16) can be written as

$$\begin{aligned}P_1 = \mathbb{E}\{\omega_x(k)\} &= (1 - \beta_{x,k})\mathbb{E}\{\omega_V(k)\} + \beta_{x,k}\mathbb{E}\{\omega_x(k-N)\} \\ &+ \beta_{x,k}T_i\left(\sum_{j=k-N}^{k-1}\mathbb{E}\{\omega_{v_x}(j)\} + \sum_{j=k-2N}^{k-N-1}\mathbb{E}\{\omega_{v_x}(j)\}\right) \\ &+ \beta_{x,k}\frac{T_i^2}{2}\left(\sum_{j=k-N}^{k-1}\mathbb{E}\{\omega_{a_x}(j)\} + \sum_{j=k-2N}^{k-N-1}\mathbb{E}\{\omega_{a_x}(j)\}\right),\end{aligned} \tag{4.25}$$

where the evaluation of $\mathbb{E}\{\omega_{v_x}(j)\}$ is presented in Appendix A.2, while $\mathbb{E}\{\omega_{a_x}(k)\}$ can be computed in a similar way as in the previous subsection.

Notice that (4.25) is similar to (4.20) then the same constraint $\beta_{x,k} \in (-1, 1)$ is required.

Finally, P_2 can be derived from (4.22) by substituting the expression of the error bias (4.25) in (4.17), that yields to the following expression for the error

variance

$$\begin{aligned} P_2 =& \mathbb{E}\{(\widehat{\delta x}(k) - \mathbb{E}\{\widehat{\delta x}(k)\})^2\} \\ =& \mathbb{E}\left\{((1-\beta_{x,k})v_V(k) + \beta_{x,k}v_x(k-N) + \right. \\ &+\beta_{x,k}\sum_{j=k-N}^{k-1}\left(T_i v_{v_x}(j) + \frac{T_i^2}{2}v_{a_x}(j)\right) + \\ &\left.-\beta_{x,k}\sum_{j=k-2N}^{k-N-1}\left(T_i v_{v_x}(j) + \frac{T_i^2}{2}v_{a_x}(j)\right)\right)^2\Bigg\}. \end{aligned} \tag{4.26}$$

Since v_{V_x}, v_x, v_{a_x} and v_{v_x} are statistically independent and $\mathbb{E}\{v_{a_x}\} = \mathbb{E}\{v_x\} = 0$, (4.26) yields

$$\begin{aligned} P_2 = \sigma_x^2(k) =& (1-\beta_{x,k})^2\sigma_V^2(k) + \beta_{x,k}^2\sigma_x^2(k-N) \\ &+ \beta_{x,k}^2\left(\sum_{j=k-N}^{k-1} T_i^2\sigma_{v_x}^2(j) + \sum_{j=k-N}^{k-1}\frac{T_i^4}{4}\sigma_{a_x}^2(j)\right) + \\ &- \beta_{x,k}^2\left(\sum_{j=k-2N}^{k-N-1} T_i^2\sigma_{v_x}^2(j) + \sum_{j=k-2N}^{k-N-1}\frac{T_i^4}{4}\sigma_{a_x}^2(j)\right). \end{aligned} \tag{4.27}$$

4.3 Solution of the Pareto Optimization Problem

The optimization problem (4.18) is convex and it can been proven just taking the derivative of the objective function respect to $\beta_{x,k}$. Thus the optimal value of $\beta_{x,k}$ for a fixed $\rho_{x,k}$ is

$$\beta_{x,k}^* = \max\left(-1, \min\left(\xi, 1\right)\right), \tag{4.28}$$

with

$$\xi = \frac{2(1-\rho_{x,k})\sigma_{V_x}^2(k+1) - 2\rho_{x,k}\gamma_{x,k}\mathbb{E}\{\omega_{V_x}(k+1)\}}{2(1-\rho_{x,k})\eta_{x,k} + 2\rho_{x,k}\gamma_{x,k}^2}.$$

The best value of the parameter $\rho_{x,k}$ is found by building a Pareto trade-off curve as in Fig. 4.2, and selecting the *knee-point* on this curve [18]. Thus, the optimal value $\rho_{x,k}^*$ is chosen such that P_1 and P_2 computed in $\beta_{x,k}^*(\rho_{x,k}^*)$ give $P_2 \simeq P_1^2$, that is given by the solution of the following optimization problem

$$\rho_{x,k}^* = \underset{\rho_{x,k}}{\arg\min}\left(P_2\left(\beta_{x,k}^*\left(\rho_{x,k}\right)\right) - P_1^2\left(\beta_{x,k}^*\left(\rho_{x,k}\right)\right)\right). \tag{4.29}$$

This problem being non-linear a numerical procedures based on the discrimination of $\rho_{x,k}$ can be employed [18].

It is important to highlight that the analytical recursive expressions of $\eta_{x,k}$ and $\gamma_{x,k}$ are different for the synchronous and the asynchronous sensor fusion cases. In

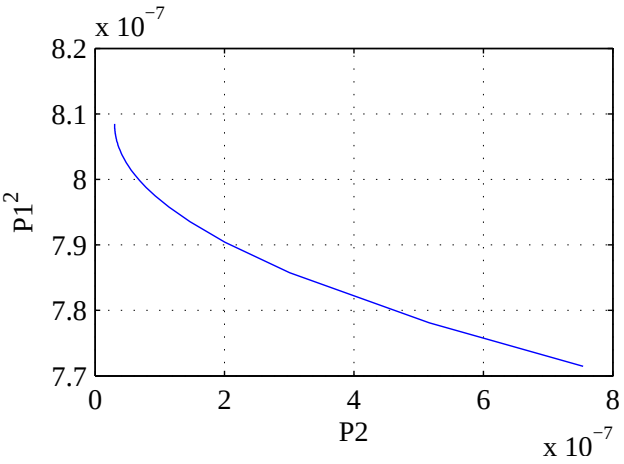


Figure 4.2: Pareto tradeoff curve. Each point on the curve is computed considering a different value of $\rho_{x,k}$.

the case of synchronous measurements, the expressions of these parameters depend on the estimation of the state bias and variance at the current step time, i.e.

$$\begin{aligned}\eta_{x,k} =&\sigma^2_{V_x}(k) + \sigma^2_x(k-1) + T^4\sigma^2_{a_x}(k-1),\\ \gamma_{x,k} =&-\mathbb{E}\{\omega_{V_x}(k)\} + \mathbb{E}\{\omega_x(k-1)\} + T^2\mathbb{E}\{\omega_{a_x}(k-1)\}.\end{aligned}$$

The solution for the asynchronous case depends on the state bias and variance at current step time, i.e.

$$\begin{aligned}\eta_{x,k} =&\sigma^2_{V_x}(k) + \sigma^2_x(k-1)+\\ &+T_i^2\left(\sum_{j=k-N}^{k-1}\sigma^2_{v_v}(j) - \sum_{j=k-2N}^{k-N-1}\sigma^2_{v_v}(j)\right)+\\ &+\frac{T_i^4}{4}\left(\sum_{j=k-N}^{k-1}\sigma^2_{a_x}(j) - \sum_{j=k-2N}^{k-N-1}\sigma^2_{a_x}(j)\right),\\ \gamma_{x,k} =&-\mathbb{E}\{\omega_{V_x}(k)\} + \mathbb{E}\{\omega_x(k-1)\}+\\ &+T_i^2\left(\sum_{j=k-N}^{k-1}\mathbb{E}\{\omega_{v_x}(j)\} + \sum_{j=k-2N}^{k-N-1}\mathbb{E}\{\omega_{v_x}(j)\}\right)\\ &+\frac{T_i^4}{4}\left(\sum_{j=k-N}^{k-1}\mathbb{E}\{\omega_{a_x}(j)\} - \sum_{j=k-2N}^{k-N-1}\mathbb{E}\{\omega_{a_x}(j)\}\right).\end{aligned}$$

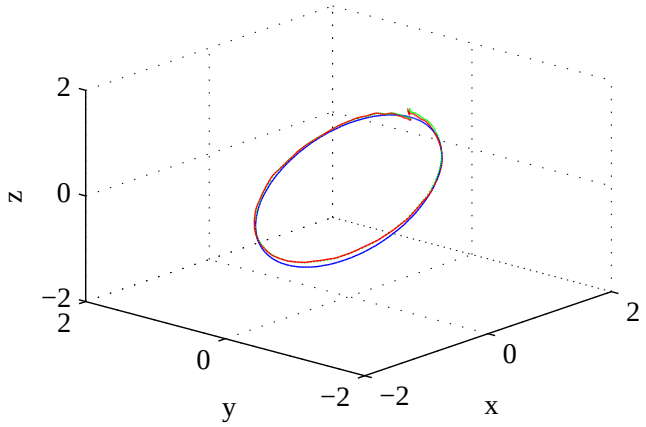


Figure 4.3: Trajectory paths in the synchronous case: ground true (blu), vision based estimation (green), and Pareto estimation (red), with $T = 0.1$ s, $\sigma^2_{a_x} = \sigma^2_{a_y} = \sigma^2_{a_z} = 0.2^2$ m^2/s^4, $\sigma_V(d) \in [1, 4]$ mm, $\mathbb{E}\{\omega_{V_x}\} = 0.3$ mm, $\sigma^2_\phi = 0.02^2, \sigma^2_\theta = 0.03^2, \sigma^2_\psi = 0.01^2$ rad/s^2. The path is performed in 30s considering a pitch rotation $\theta = \pi/6$.

Remark 1. *The solution of the optimization problem (4.18) is equivalent to an optimization respect to both $\beta_{x,k}$ and $\rho_{x,k}$ parameters. In fact, imposing first order optimality condition respect to $\rho_{x,k}$ giving $P_2 \simeq P_1^2$.*
Then, imposing the first order optimality condition respect to $\beta_{x,k}$

$$(1 - \rho_{x,k}) \frac{\partial P_2}{\partial \beta_{x,k}} + 2\rho_{x,k} P_1 \frac{\partial P_1}{\partial \beta_{x,k}} = 0$$

which should be solved numerically chosing $\beta_{x,k}$ depending on $\rho_{x,k}$ such the first enounced condition $P_2 \simeq P_1^2$ is verified. The term Pareto is derived from the classical game theory where the goal of two players is to choose a given strategy such to maximize(minimize) their utility, in the presented problem represented by P_2 and P_1^2 respectively.

4.4 Simulations

The proposed method has been tested on simulated trajectories both for the synchronous and the asynchronous case. The considered path is a $3D$ circle generated according to different time law profiles emulating different operating conditions (see Figs. 4.3 and 4.4).

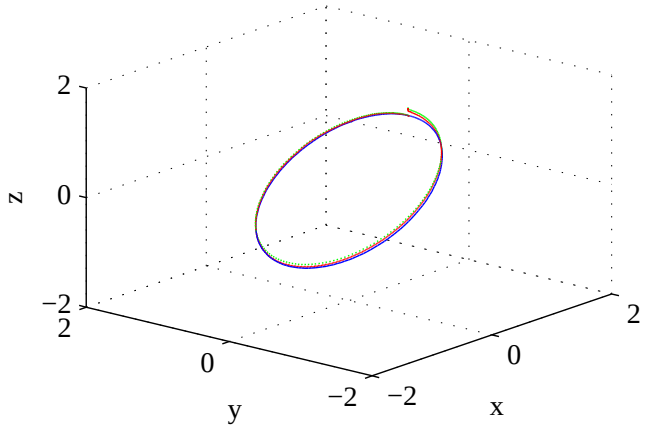


Figure 4.4: Trajectory paths in the asynchronous case: ground true (blue dashed line), vision based estimation (green point dashed line), and Pareto estimation (red continuous line), with $T_i = 0.01s$, $T_V = 0.1s$, $\sigma^2_{a_x} = \sigma^2_{a_y} = \sigma^2_{a_z} = 0.3^2$ m^2/s^4, $\sigma_V(d) \in [0.1, 0.4]$ mm, $\mathbb{E}\{\omega_{V_x}\} = 0.3$ mm $\sigma^2_\phi = 0.02^2, \sigma^2_\theta = 0.03^2, \sigma^2_\psi = 0.01^2$ rad/s^2. The path is performed in 30s considering a roll, pitch and yaw rotation $\phi = -\pi/4, \theta = \pi/8, \psi = -\pi/6$.

4.4.1 System Characterization

The variance for the vision measurement is chosen so as to model a variable distance d with respect to the observed target, i.e.

$$\sigma^2_V(k) = \sigma^2_{V\min} + \frac{\sigma^2_{V\max} - \sigma^2_{V\min}}{d_{\max} - d_{\min}}(d(k) - d_{\min}),$$

where $d_{\min}$ and $d_{\max}$ represent the minimum and the maximum distance, respectively.

The trajectory estimation error of the proposed optimization technique has been compared with the case when only vision data are employed. In both the synchronous and the asynchronous cases, the estimation error benefits of the proposed approach, as it is shown in Figs. 4.5 and 4.6. Tables 4.1 and 4.2 show the robustness of this method in different working conditions, by considering different time laws.

4.4.2 System Performances

The proposed method has been compared to a Stochastic Cloning Kalman filtering (SC-KF) [87] using possible IMU noise values. A Second order dynamic model has

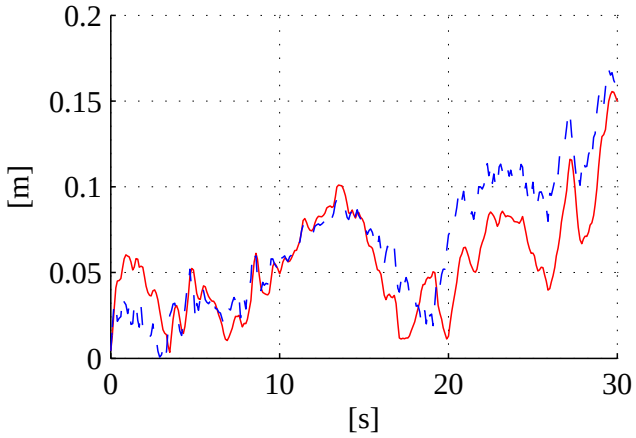


Figure 4.5: Synchronous case: time history of the norm of the trajectory estimation error with respect to the ground true by using only vision data (blue dashed line) and with the Pareto optimization (red continous line).

Table 4.1: Average error norm in the synchronous case

Case	Trapez. Velocity	Cubic Poly	5^{th}order Poly
Vision error [m]	0.0676	0.0671	0.0665
Estimation error [m]	0.0568	0.0586	0.0549

been used, where the state is increased by the old visual system position, such to consider differential visual position measurements.

Differently from the proposed approach, Kalman filtering techniques rely on the state and measurement covariance matrices, which are typically constant in classic Kalman-filter implementations. The proposed approach, instead, takes into account the variance and the bias on the system state at each instant of time, thus producing a significant benefit.

To compare the two different approaches, the same time varying law estimated variance, as employed in the proposed method, is used in the SC-KF implementation. Moreover, the bias is modeled as a constant parameter causing a state augmentation. With reference to the synchronous case, Table 4.1 show the comparison between the two different mentioned approaches. whereas, Table 4.3 and Fig. 4.7 show the comparison for the asynchronous case. In both cases the proposed problem formulation is able to reduce the error norm of about 30%. In the asynchronous case, it can be noticed an increasing of oscillations in the error norm, which is caused by the presence of significant noise on acceleration measurements typical on aerial platforms.

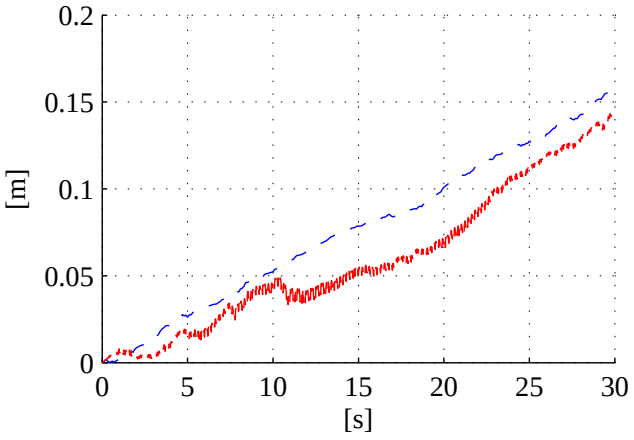


Figure 4.6: Asynchronous case: time history of the norm of the trajectory estimation error with respect to the ground true by using only vision data (blue dashed line) and with the Pareto optimization (red continuous line).

Table 4.2: Average error norm in the asynchronous case

Case	Trapez. Velocity	Cubic Poly	5^{th}order Poly
Vision error [m]	0.0770	0.0769	0.0760
Estimation error [m]	0.0574	0.0544	0.0497

Notice that, when the vision measurement is not available only acceleration is used for few seconds to perform system position identification. By starting the estimation from an optimal position value, as shown in (4.12), it is possible to prevent estimation divergence due to acceleration measurements. The bias $\mathbb{E}\{\omega_{a_x}(k)\}$ and its corresponding variance in case of too noisy acceleration, can be computed from filtered acceleration data obtained using a classical first order low pass filter.

The index (4.18) minimizes a combination of state bias and variance, while through a KF approach the covariance state matrix is minimized. Being a combination of bias and variance, it can be directly interpreted as a measure of the Mean Square Error (MSE) of the unknown scalar parameter ω_x, similarly to the biased estimation in [59]. This is even confirmed by the analysis of the average covariance values.

The computational complexity of the proposed solution is $\mathrm{O}(n_1 n_2)$ with $n_1 = t/T$, $n_2 = 1/T_\rho$, where t is the current time, $T = T_i$ and T_ρ is the step time used to search the ρ value such that $P_2 \simeq P_1^2$, while for KF approaches it is just $\mathrm{O}(n_1)$. The difference is confirmed by the average computational time, of both methods

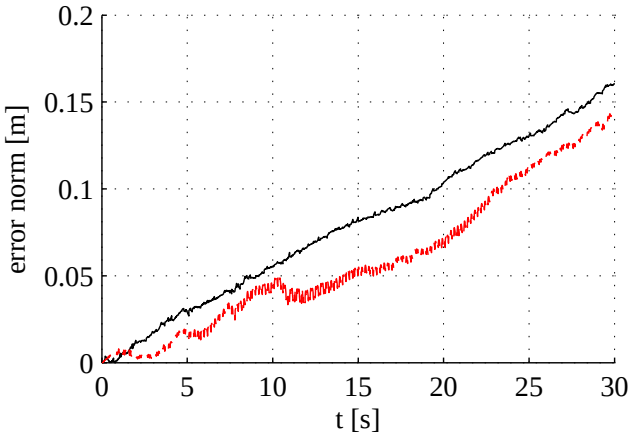


Figure 4.7: Asynchronous case: comparison between SC-KF (black point dashed line) and the proposed method (red continuous line).

Table 4.3: Average error norm

Case	Trapez. Velocity	Cubic Poly	5^{th}order Poly
Proposed method [m]	0.0574	0.0544	0.0497
Sto. Clo. KF [m]	0.0799	0.0798	0.0792

shown in Table 4.4 for two different hardware platforms.

Table 4.4: Computational time

Case	Intel Core $i2$	Intel Core $i7$
Proposed method [ms]	2.3	1.6
Sto. Clo. KF [ms]	0.4	0.18

Chapter 5

Conclusion and Future Research Directions

A brief digest of the methods presented in this thesis and the achieved results will be the object of the current chapter. Proposals for future research directions will be discussed as well.

5.1 Main results

Different vision based algorithms have been proposed in this work, showing how cameras, due to the characteristics mentioned in Section 1.2.1, can be used for reactive control, environment reconstruction, eventually combined with other sensors like IMU to increase robustness and motion estimation rate. The thesis has been obviously split in three different parts, which are namely:

- Feedback Reactive control based on visual information
- 3D Environment Reconstruction
- Vision and IMU sensor fusion.

For each of the previous items, new contributions have been presented and validated through simulations and experimental results.

5.2 Conclusion

In this thesis, the main purpose was to show how vision algorithms can be used for reactive control, environment reconstruction and combined with other sensors to increase robustness and motion estimation rate. In particular, a new vision-based obstacle avoidance technique for indoor navigation of Micro Aerial Vehicles

has been presented. The Depth Map of the surrounding environment has been constructed using only visual and inertial measurements. An existing closed-form solution for the absolute-scale velocity estimation based on visual correspondences and inertial measurements has been generalized and employed for the velocity estimation. This last has been used for the evaluation of the absolute-scaled Optical Flow, which allows the construction of the desired Depth Map. Relying on this map, a safe navigation control has been proposed, which able to avoid lateral obstacles, to self-limit the cruise velocity in view of the available free space, and to dynamically set the regions of interest for image features extraction. Simulations have been carried out to prove the effectiveness of the proposed solution. To conclude the first part, this work demonstrates a first step towards autonomous dynamic grasping and manipulation for micro aerial vehicles in unstructured environments. A quadrotor system equipped with a monocular camera is considered, formulating the dynamics of the underactuated system directly in the virtual image plane. The system has been demonstrated to be differential flat, with the image coordinates being the set of flat outputs. The trajectory generation method guarantees dynamic feasibility and enables incorporating visual constraints as linear constraints. A non-linear vision-based controller for trajectory tracking in the image space is presented. A proof of stability is given and a validation of the controller both in simulation and in experimentation on a quadrotor has been provided.

In the second part, a framework for real-time pose estimation for autonomous flight and cooperative mapping is proposed. By decomposing the problem into a monocular SLAM problem with sparse representation and the problem of associating robot poses and depths with features create a dense 3-D map, avoiding the computational bottleneck of 3-D RGB-D cooperative SLAM, distributing a large fraction of the computations and achieve increased robustness to noise in depth which is typical in outdoor or brightly lit environments. The localization performance is comparable to one of the most used RGB-D SLAM frameworks [39]. In particular, the approach allows pose estimation at frame rates and yields global position estimates even if the depth is not available at every frame. Further the presented approach approach yields a sparse map in addition to the dense map. The sparse map is particularly useful for computationally limited platform or when the scale factor, due to environmental constraints, cannot be estimated. Experimental results based on camera datasets [112] and aerial vehicles' data in Section 3.4 substantiate these claims. In the same field, a different environment representation has been proposed, in case of planning and high level. The system at work in two real challenging scenarios where the Aerial Service Vehicle is involved, respectively, in a physical and visual inspection task. Moreover, the point cloud data is generated by different sensors, showing the general applicability of the presented environment reconstruction approach.

In the last part, a new sensor fusion technique for motion estimation which

combines visual and inertial measurements via a Pareto optimization process has been introduced. The proposed method minimizes a combination of state bias and variance by balancing available data input in an optimal way. Only the measurements statistical characterization is required, without any prior knowledge of the motion model. The effectiveness of the proposed theoretical approach, in terms of accuracy, computational requirements, and robustness was tested with simulation case studies and compared to a Kalman-filter based approach. It is shown that the proposed method gives a benefit in terms of estimation accuracy with a limited increase of the computational complexity.

5.3 Proposals for the future

Likewise in section 5.1, where three main research topics can be identified, the directions for future researches can be also split in three parts.

Concerning visual reactive control, a next step would be to validate the optical flow approach with experimental results and generalize the control law to full three dimensions and considering the yaw of the robot by using image moments to detect the primary axis of the cylinder.

Clearly, the integration of an IMU into the 3D reconstruction framework, will increase the performance for aggressive flight maneuvers. In future work real-time control loops at 100 Hz. or higher for autonomous flight will be developed. Future work will also focus on additional real-world experiments and on the extension to the case of cooperative vehicles [80].

For the sensor fusion part, obviously a validation based on real experiments is the natural extension of the proposed method. Moreover another extension would be to consider to incorporate the possibility to estimate 6 DOF pose. Finally, an extended analysis on the optimization index can be useful to prove from a theoretical point of view, the benefit of this approach with respect to classical Kalman filtering techniques.

Appendix A

Appendix

A.1 Image based Visual Servoing

This section contains the additional material related to section 2.4.

A.1.1 Stability of Attitude Dynamics

For the attitude controller, the Lyapunov candidate is

$$\mathcal{V}_R = \frac{J_q}{2} e_\Omega \cdot e_\Omega + K_R \Psi\left(R, R_d\right) + c_2 e_R \cdot e_\Omega, \tag{A.1}$$

with c_2 being a positive scalar, such that,

$$\mathbf{z}_\theta^T M_\theta \mathbf{z}_\theta \leq \mathcal{V}_R \leq \mathbf{z}_\theta^T M_\Theta \mathbf{z}_\theta, \tag{A.2}$$

$$\dot{\mathcal{V}}_R \leq -\mathbf{z}_\theta^T W_\theta \mathbf{z}_\theta, \tag{A.3}$$

where $\mathbf{z}_\theta = \left[\left\|e_R\right\|, \left\|e_\Omega\right\|\right]^T$, and M_θ, M_Θ, and W_θ are positive definite.

A.1.2 Stability of Translational Dynamics in the Image Coordinates

We take an approach very similar to [65] to show that the controller is exponentially stable. First, define $K_p', K_d', B, \alpha \in \mathbb{R}$ as,

$$K_p' = m_q \left\|J\right\| \left\|J^{-1}\right\| K_p \tag{A.4}$$

$$K_d' = \left\|J\right\| \left(m_q \left\|J^{-1}\right\| K_d + \left\|\dot{J^{-1}}\right\|\right) \tag{A.5}$$

$$B = \left\|J\right\| \left(\left\|G_A\right\| + m_q \left\|J^{-1}\right\| \left\|\ddot{\mathbf{v}}^d\right\| + \left\|\dot{J^{-1}}\right\| \left\|\dot{\mathbf{v}}^d\right\|\right) \tag{A.6}$$

$$\alpha = \left\|e_R\right\| \tag{A.7}$$

and define $W_{v_1}, W_{v_2}, W_{v\theta}, W_v \in \mathbb{R}^{2\times 2}$ as

$$W_{v_1} = \begin{bmatrix} \frac{c_1 K_p}{m_q} & \frac{c_1 K_d}{2m_q} \\ \frac{c_1 K_d}{2m_q} & K_d - c_1 \end{bmatrix}, W_{v\theta} = \begin{bmatrix} \frac{c_1}{m_q} B & 0 \\ B & 0 \end{bmatrix} \tag{A.8}$$

$$W_{v_2} = \begin{bmatrix} \frac{c_1 \alpha K'_p}{m_q} & \frac{\alpha}{2}\left(\frac{c_1}{m_q} K'_d + K'_p\right) \\ \frac{\alpha}{2}\left(\frac{c_1}{m_q} K'_d + K'_p\right) & \alpha K'_d \end{bmatrix} \tag{A.9}$$

$$W_v = W_{v_1} - W_{v_2}. \tag{A.10}$$

Suppose we choose positive constants $c_1, K_p, K_d, K_R, K_\Omega$ such that,

$$K_p > \frac{c_1^2}{m_q} \tag{A.11}$$

$$\lambda_{min}(W_\theta) > \frac{4\left\|W_{v\theta}\right\|^2}{\lambda_{min}(W_v)} \tag{A.12}$$

Then, there exists positive constants $\gamma_1, \gamma_2, \gamma_3$, such that $\|J\| \leq \gamma_1$, $\left\|J^{-1}\right\| \leq \gamma_2$, $\left\|\dot{J^{-1}}\right\| \leq \gamma_3$, and if initial conditions and the desired trajectory satisfy

$$\alpha < \frac{1}{m_q \gamma_1 \gamma_2}, \tag{A.13}$$

$$dist(\mathbf{v}_d(t), V^c) < \left\|\mathbf{e}_v(0)\right\|, \tag{A.14}$$

where V^c is the complement of V, and $dist(\mathbf{v}_d(t), V^c) = \inf_{t\in[0,\infty), w\in V^c} \|\mathbf{v}_d(t) - w\|$ is the smallest distance between a trajectory and a set, then the zero equilibrium $(\mathbf{e}_v, \dot{\mathbf{e}}_v, e_R, e_\Omega) = (\mathbf{0}, \mathbf{0}, 0, 0)$ is locally exponentially stable.

Proof. Using (2.33), we can determine the image errors

$$\ddot{\mathbf{e}}_v = \ddot{\mathbf{v}} - \ddot{\mathbf{v}}_d = \frac{1}{m} J\left[f R\mathbf{e}_2 - \mathbf{G}_A\right] + \dot{J}J^{-1}\dot{\mathbf{v}} - \ddot{\mathbf{v}}_d \tag{A.15}$$

so that

$$m\ddot{\mathbf{e}}_v = f J R\mathbf{e}_2 - J\mathbf{G}_A + m\dot{J}J^{-1}\dot{\mathbf{v}} - m\ddot{\mathbf{v}}_d. \tag{A.16}$$

Defining

$$\mathbf{X} = J\frac{f}{\mathbf{e}_2^T R_c^T R\mathbf{e}_2}\left(\left(\mathbf{e}_2^T R_c^T R\mathbf{e}_2\right) R\mathbf{e}_2 - R_c\mathbf{e}_2\right), \tag{A.17}$$

the error dynamics become

$$m\ddot{\mathbf{e}}_v = J\left(\frac{f}{\mathbf{e}_2^T R_c^T R\mathbf{e}_2} R_c\mathbf{e}_2\right) + \mathbf{X} - J\mathbf{G}_A + m\dot{J}J^{-1}\dot{\mathbf{v}} - m\ddot{\mathbf{v}}_d. \tag{A.18}$$

Next, let

$$f = \mathbf{A}\cdot R\mathbf{e}_2 \tag{A.19}$$

and the commanded attitude be defined by

$$R_c\mathbf{e}_2 = \frac{\mathbf{A}}{\|\mathbf{A}\|}. \tag{A.20}$$

Then, from the previous two equations, we have

$$f = \|\mathbf{A}\|\, \mathbf{e}_2^T R_c^T R\mathbf{e}_2. \tag{A.21}$$

Substituting this into (A.18) and using $\mathbf{A}$, we have

$$\begin{aligned} m\ddot{\mathbf{e}}_v &= J\left(\frac{\|\mathbf{A}\|\, \mathbf{e}_2^T R_c^T R\mathbf{e}_2}{\mathbf{e}_2^T R_c^T R\mathbf{e}_2} R_c\mathbf{e}_2\right) + \mathbf{X} - J\mathbf{G}_A + m\dot{J}J^{-1}\dot{\mathbf{v}} - m\ddot{\mathbf{v}}_d && \text{(A.22)}\\ &= J\left(\|\mathbf{A}\|\, R_c\mathbf{e}_2\right) + \mathbf{X} - J\mathbf{G}_A + m\dot{J}J^{-1}\dot{\mathbf{v}} - m\ddot{\mathbf{v}}_d && \text{(A.23)}\\ &= J\mathbf{A} + \mathbf{X} - J\mathbf{G}_A + m\dot{J}J^{-1}\dot{\mathbf{v}} - m\ddot{\mathbf{v}}_d && \text{(A.24)}\\ &= -K_p\mathbf{e}_v - K_d\dot{\mathbf{e}}_v + \mathbf{X} && \text{(A.25)} \end{aligned}$$

which has the same form as (83) in [65]. We use the same Lyapunov candidate, but in our image coordinates,

$$\mathcal{V}_v = \frac{1}{2}K_p\left\|\mathbf{e}_v\right\|^2 + \frac{1}{2}m\left\|\dot{\mathbf{e}}_v\right\|^2 + c_1\mathbf{e}_v\cdot\dot{\mathbf{e}}_v. \tag{A.26}$$

Now, let $\mathbf{z}_v = \begin{bmatrix} \|\mathbf{e}_v\|, \|\dot{\mathbf{e}}_v\| \end{bmatrix}^T$, then it follows that the Lyapunov function $\mathcal{V}_v$ is bounded as

$$\mathbf{z}_v^T M_v \mathbf{z}_v \le \mathcal{V}_v \le \mathbf{z}_v^T M_V \mathbf{z}_v, \tag{A.27}$$

where $M_v, M_V \in \mathbb{R}^{2\times 2}$ are defined as,

$$M_v = \frac{1}{2}\begin{bmatrix} K_p & -c_1 \\ -c_1 & m \end{bmatrix},\ M_V = \frac{1}{2}\begin{bmatrix} K_p & c_1 \\ c_1 & m \end{bmatrix}. \tag{A.28}$$

Then,

$$\dot{\mathcal{V}}_v = K_p\left(\dot{\mathbf{e}}_v\cdot\mathbf{e}_v\right) + m\left(\ddot{\mathbf{e}}_v\cdot\dot{\mathbf{e}}_v\right) + c_1\left(\mathbf{e}_v\cdot\ddot{\mathbf{e}}_v + \dot{\mathbf{e}}_v\cdot\dot{\mathbf{e}}_v\right), \tag{A.29}$$

and incorporating (A.25),

$$\begin{aligned} \dot{\mathcal{V}}_v = &-\frac{c_1K_p}{m}\left\|\mathbf{e}_v\right\|^2 - \left(K_d - c_1\right)\left\|\dot{\mathbf{e}}_v\right\|^2 \\ &-c_1\tfrac{K_d}{m}\left(\mathbf{e}_v\cdot\dot{\mathbf{e}}_v\right) + \mathbf{X}\cdot\left(\tfrac{c_1}{m}\mathbf{e}_v + \dot{\mathbf{e}}_v\right). \end{aligned} \tag{A.30}$$

Now, we establish a bound on $\mathbf{X}$. From (A.17),

$$\begin{aligned} \mathbf{X} &= J\frac{f}{\mathbf{e}_2^T R_c^T R\mathbf{e}_2}\left(\left(\mathbf{e}_2^T R_c^T R\mathbf{e}_2\right)R\mathbf{e}_2 - R_c\mathbf{e}_2\right) && \text{(A.31)}\\ \|\mathbf{X}\| &\le \|J\|\left\|\frac{\|A\|\, R_c\mathbf{e}_2\cdot R\mathbf{e}_2}{R_c\mathbf{e}_2\cdot R\mathbf{e}_2}\right\|\|e_R\| && \text{(A.32)}\\ &\le \|J\|\,\|\mathbf{A}\|\,\|e_R\| && \text{(A.33)}\\ &\le \|J\|\left\|\mathbf{G}_A + mJ^{-1}\left[-K_p\mathbf{e}_v - K_d\dot{\mathbf{e}}_v + \ddot{\mathbf{v}}_d\right] + \dot{J^{-1}}\left[\dot{\mathbf{e}}_v + \dot{\mathbf{v}}_d\right]\right\|\|e_R\| && \text{(A.34)}\\ &\le \left(K_p'\left\|\mathbf{e}_v\right\| + K_d'\left\|\dot{\mathbf{e}}_v\right\| + B\right)\|e_R\| && \text{(A.35)} \end{aligned}$$

where K_p', K_d', B are as defined in (A.4)-(A.6), and from [65], $0 \leq \|e_R\| \leq 1$.

Next we will show that there exists positive constants $\gamma_1, \gamma_2, \gamma_3$ s.t., $\|J\| \leq \gamma_1, \left\|J^{-1}\right\| \leq \gamma_2$, and $\left\|\dot{J^{-1}}\right\| \leq \gamma_3$. Since Γ is smooth (we only require C^2 here), J is smooth on the closed set S. This implies J is bounded on S, i.e., $\exists \gamma_1 > 0$, s.t. $\|J\| < \gamma_1$. Next, since J is smooth and nonsingular on S, the inverse is well defined and is smooth on S, which implies J^{-1} is bounded on S, i.e., $\exists \gamma_2 > 0$, s.t. $\left\|J^{-1}\right\| < \gamma_2$. Next, observe that $\frac{d}{dt}J^{-1}(\mathbf{r}_q) = \frac{\partial}{\partial \mathbf{r}_q}J^{-1}(\mathbf{r}_q)\dot{\mathbf{r}}_q$ is a composition of smooth functions on S, implying that it is bounded on S, i.e., $\exists \gamma_3 > 0$, s.t. $\left\|\dot{J^{-1}}\right\| < \gamma_3$.

Then, similar to [107], we can express $\dot{\mathcal{V}}_v$ as

$$\dot{\mathcal{V}}_v = -\begin{bmatrix} \mathbf{e}_v^T & \dot{\mathbf{e}}_v^T \end{bmatrix} W_{v_1} \begin{bmatrix} \mathbf{e}_v \\ \dot{\mathbf{e}}_v \end{bmatrix} + \mathbf{X} \cdot \left(\frac{c_1}{m}\mathbf{e}_v + \dot{\mathbf{e}}_v\right) \tag{A.36}$$

$$\begin{aligned} \leq -\begin{bmatrix} \mathbf{e}_v^T & \dot{\mathbf{e}}_v^T \end{bmatrix} W_{v_1} \begin{bmatrix} \mathbf{e}_v \\ \dot{\mathbf{e}}_v \end{bmatrix} \\ + K_p' \|\mathbf{e}_v\| \|e_R\| \left(\tfrac{c_1}{m}\|\mathbf{e}_v\| + \|\dot{\mathbf{e}}_v\|\right) \\ + K_d' \|\dot{\mathbf{e}}_v\| \|e_R\| \left(\tfrac{c_1}{m}\|\mathbf{e}_v\| + \|\dot{\mathbf{e}}_v\|\right) \\ + B \|e_R\| \left(\tfrac{c_1}{m}\|\mathbf{e}_v\| + \|\dot{\mathbf{e}}_v\|\right). \end{aligned} \tag{A.37}$$

This can be written as,

$$\dot{\mathcal{V}}_v \leq -\mathbf{z}_v^T W_v \mathbf{z}_v + \mathbf{z}_v^T W_{v\theta} \mathbf{z}_\theta \tag{A.38}$$

where $W_{v\theta}, W_v$ are as defined in (A.8), (A.10). Since $W_v = (W_v)^T$ and $W_v \in \mathbb{R}^{2\times 2}$, it is sufficient to show that $\det(W_v) > 0$ and $W_v(1,1) > 0$ in order to claim that $W_v > 0$. Then, from the assumption on α in (A.13), we have $w_{11} > 0$. This is reasonable since α is a functional on the attitude error such that $\alpha \in [0, 1]$. Thus, the assumption in (A.13) is simply a bound on the attitude error. The determinant can be expressed as a quadratic function of K_d such that

$$\det(W_v) = \beta_0 + \beta_1 K_d + \beta_2 K_d^2 \tag{A.39}$$

and β_i is a function of c_1, K_p, γ_1, γ_2, γ_3, and m. The critical point of the quadratic occurs when

$$K_d = \frac{K_p m}{c_1} + \frac{K_p m + \alpha c_1 \gamma_1 \gamma_3}{c_1 \left(1 - \alpha \gamma_1 \gamma_2 m\right)} \tag{A.40}$$

and has a value of

$$\det(W_v) = \frac{K_p \left(1 - \alpha \gamma_1 \gamma_2 m\right)\left(K_p m - c_1^2\right)}{m}. \tag{A.41}$$

In both equations, $(1 - \alpha\gamma_1\gamma_2 m) > 0$ as a result of the assumption in (A.13). Thus (A.40) is positive, and by (A.11), (A.41) is positive and $W_v' > 0$. Now, we

consider the combined Lyapunov candidate for the translational and rotational error dynamics, $\mathcal{V} = \mathcal{V}_v + \mathcal{V}_R$. From (A.2) and (A.27), we have,

$$\mathbf{z}_v^T M_v \mathbf{z}_v + \mathbf{z}_\theta^T M_\theta \mathbf{z}_\theta \leq \mathcal{V} \leq \mathbf{z}_\theta^T M_\Theta \mathbf{z}_\theta + \mathbf{z}_v^T M_V \mathbf{z}_v. \tag{A.42}$$

Further, we see that

$$\begin{aligned} \dot{\mathcal{V}} \leq & -\mathbf{z}_v^T W_v \mathbf{z}_v + \mathbf{z}_v^T W_{v\theta} \mathbf{z}_\theta - \mathbf{z}_\theta^T W_\theta \mathbf{z}_\theta, & \text{(A.43)} \\ \leq & -\lambda_{min}(W_v) \left\|\mathbf{z}_v\right\|^2 + \left\|W_{v\theta}\right\| \left\|\mathbf{z}_v\right\| \left\|\mathbf{z}_\theta\right\| \\ & -\lambda_{min}(W_\theta) \left\|\mathbf{z}_\theta\right\|^2, & \text{(A.44)} \end{aligned}$$

and from (A.12), we have $\dot{\mathcal{V}}$ to be negative definite, and the zero equilibrium of the closed-loop system is locally exponentially stable. □

A.2 Sensor Fusion

By developing the third term of (4.25) we have

$$\begin{aligned} &\sum_{j=k-N}^{k-1} \mathbb{E}\{\omega_{v_x}(j)\} = \mathbb{E}\{\omega_{v_x}(k-N)\} \\ &+ \mathbb{E}\{\omega_{v_x}(k-N+1)\} + \cdots + \mathbb{E}\{\omega_{v_x}(k-1)\}, \end{aligned}$$

where every single term can be written as

$$\begin{aligned} &\mathbb{E}\{\omega_{v_x}(k-N+1)\} = \\ &\qquad \mathbb{E}\{\omega_{v_x}(k-N)\} + T_i \mathbb{E}\{\omega_{a_x}(k-N)\} \end{aligned}$$

$$\begin{aligned} &\mathbb{E}\{\omega_{v_x}(k-N+2)\} = \mathbb{E}\{\omega_{v_x}(k-N+1)\} \\ &+ T_i \mathbb{E}\{\omega_{a_x}(k-N+1)\} = \mathbb{E}\{\omega_{v_x}(k-N)\} \\ &+ T_i \mathbb{E}\{\omega_{a_x}(k-N)\} + T_i \mathbb{E}\{\omega_{a_x}(k-N+1)\} \end{aligned}$$

$$\begin{aligned} &\mathbb{E}\{\omega_{v_x}(k-1))\} = \mathbb{E}\{\omega_{v_x}(k-N)\} \\ &+ T_i \mathbb{E}\{\omega_{a_x}(k-N)\} + \cdots + T_i \mathbb{E}\{\omega_{a_x}(k-2)\}. \end{aligned}$$

Thus it gives

$$\begin{aligned} \sum_{j=k-N}^{k-1} T_i \mathbb{E}\{\omega_{v_x}(j)\} &= (N-1)\mathbb{E}\{\omega_{v_x}(k-N)\} \\ &+ T_i \sum_{j=k-N}^{k-1} (k-j)\mathbb{E}\{\omega_{a_x}(j)\}. \end{aligned}$$

This results shows how the recursive expression of the velocity bias is obtained. The same approach can be used to derive the velocity variance σ_{v_x}.

In the following the symbol σ_i for the variance associated to i will be employed.

Lemma A.2.1. *Let $\widetilde{\varphi}(k)$ be Gaussian and independent respect to the acceleration component $\widetilde{a}(k)$ then*

$$\mathbb{E}\{\widetilde{a}(k)\sin(\widetilde{\varphi}(k))\} = a(k)\sin(\varphi(k))e^{\frac{-\sigma_\varphi^2}{2}}.$$

Proof. Using sine properties, since $\widetilde{a}(k)$ and $\widetilde{\varphi}(k)$ are statistically independent we obtain that

$$\begin{aligned}&\mathbb{E}\{\widetilde{a}(k)\sin(\widetilde{\varphi}(k))\} = \mathbb{E}\{a(k)\}\mathbb{E}\{\sin(\widetilde{\varphi}(k))\} = \\ &a(k)\mathbb{E}\{\sin(\varphi(k))\}\mathbb{E}\{\cos(\omega_\varphi(k))\}+ \\ &a(k)\mathbb{E}\{\cos(\varphi(k))\}\mathbb{E}\{\sin(\omega_\varphi(k))\}.\end{aligned}$$

As shown in [8, 9]

$$\begin{aligned}\mathbb{E}\{\cos(\omega_\varphi(k))\} &= e^{\frac{-\sigma_\varphi^2}{2}} \\ \mathbb{E}\{\sin(\omega_\varphi(k))\} &= 0,\end{aligned}$$

that leads

$$\mathbb{E}\{\widetilde{a}(k)\sin(\widetilde{\varphi}(k))\} = a(k)\sin(\varphi(k))e^{\frac{-\sigma_\varphi^2}{2}}.$$

□

Lemma A.2.2. *Let $\widetilde{\varphi}(k)$ be Gaussian and independent with respect to the acceleration component $\widetilde{a}(k)$ then*

$$\mathbb{E}\{\widetilde{a}^2(k)\sin^2(\widetilde{\varphi}(k))\} = \sigma_a^2\left(\frac{1}{2} - \frac{1}{2}\cos(2\varphi)\right)e^{-2\sigma_\varphi^2}.$$

Proof. Since $\widetilde{a}(k)$ and $\widetilde{\varphi}(k)$ are statistically independent we obtain that

$$\mathbb{E}\{\widetilde{a}^2(k)\sin^2(\widetilde{\varphi}(k))\} = \mathbb{E}\{\widetilde{a}^2(k)\}\mathbb{E}\{\sin^2(\widetilde{\varphi}(k))\}.$$

Then using sine properties

$$\begin{aligned}&\mathbb{E}\{\sin^2(\widetilde{\varphi}(k))\} = \mathbb{E}\left\{\frac{1}{2} - \frac{1}{2}\cos(2\widetilde{\varphi}(k))\right\} = \\ &\mathbb{E}\left\{\frac{1}{2} - \frac{1}{2}\cos(2(\varphi(k) + \omega_\varphi(k))\right\} = \\ &\frac{1}{2} - \frac{1}{2}\mathbb{E}\left\{\cos(2(\varphi(k))\cos(2\omega_\varphi(k))+\right. \\ &\left.-\sin(2(\varphi(k))\sin(2\omega_\varphi(k))\right\}.\end{aligned}$$

As shown in [8, 9]

$$\mathbb{E}\{\cos(2\omega_\phi(k))\} = e^{-2\sigma_\varphi^2}.$$

The result can be extended to the sin term considering its series

$$\mathbb{E}\{\sin(2\omega_\varphi(k))\} = \mathbb{E}\left\{2\omega_\varphi(k) + \cdots + \right.$$
$$\left. + (-1)^n \frac{2\omega_\varphi(k)^{2n+1}}{(2n+1)!}\right\} = 0.$$

So the initial expression becomes

$$\mathbb{E}\{\sin^2(\widetilde{\varphi}(k))\} = \left(\frac{1}{2} - \frac{1}{2}\cos(2\varphi)\right) e^{-2\sigma_\varphi^2}.$$

Then the final result is that

$$\mathbb{E}\{\widetilde{a}^2(k)\sin^2(\widetilde{\varphi}(k))\} = \sigma_a^2 \left(\frac{1}{2} - \frac{1}{2}\cos(2\varphi)\right) e^{-2\sigma_\varphi^2}.$$

□

Bibliography

[1] *"Ascending Technologies, GmbH," http://www.asctec.de.*

[2] *EU Collaborative Project ICT-248669, "AIRobots",* www.airobots.eu.

[3] *"Gumstix, Inc.," http://www.gumstix.com.*

[4] *OpenNI, www.openni.org.*

[5] *Optitrack, "optical motion capture system and tracking software"* http://www.naturalpoint.com/optitrack/.

[6] *Robotics operating system, "ros",* www.ros.org.

[7] *"Vicon Motion Systems, Inc," http://www.vicon.com.*

[8] A. De Angelis and C. Fischione, *A distributed information fusion method for localization based on Pareto optimization*, International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS) (Barcelona), 2011, pp. 1–8.

[9] A. De Angelis, C. Fischione, and P. Händel, *A sensor fusion algorithm for mobile node localization*, IFAC World Congress (Milan), 2011, pp. 11258–11264.

[10] L. Armesto, J. Tornero, and M. Vincze, *On multi-rate fusion for non-linear sampled-data systems: Application to a 6D tracking system*, Robotics and Autonomous Systems **56** (2008), 706–715.

[11] C. Audras, A. Comport, M. Meilland, and P. Rives, *Real-time dense appearance-based slam for rgb-d*, Australasian Conference on Robotics and Automation (Melbourne), 2011.

[12] A.-J. Baerveldt and R. Klang, *A low-cost and low-weight attitude estimation system for an autonomous helicopter*, IEEE International Conference on Intelligent Engineering Systems (Budapest), 1997, pp. 391–395.

[13] K. Bass, B. Leith, J. Anderson, P. Bassett, J. Stevens, H. Pearson, and J. Turner, *Nature's Most Amazing Events*, [DVD]. BBC Worldwide Ltd. Programs, 2009.

[14] M. Blöesch, S. Weiss, D. Scaramuzza, and R. Siegwart, *Vision based MAV navigation in unknown and unstructured environments*, ICRA 2010 (Anchorage), 2010, pp. 21–28.

[15] S. Bouabdallah, P. Murrieri, and R. Siegwart, *Design and control of an indoor micro quadrotor*, IEEE International Conference on Robotics and Automation (New Orleans), April 2004, pp. 4393–4398.

[16] J Bouguet, *Pyramidal implementation of the affine lucas kanade feature trackerdescription of the algorithm*, pages.slc.edu (2001).

[17] F. Bourgeois, L. Kneip, S. Weiss, and R. Siegwart, *Delay and dropout tolerant state estimation for mavs*, 12th Int. Symposium on Experimental Robotics (New Delhi and Agra), 2010.

[18] S. Boyd and L. Vandenberghe, *Convex Optimization*, vol. 25, 2004.

[19] G. Byrnes, T. Libby, N. T.-L Lim, and A. J. Spence, *Gliding saves time but not energy in Malayan colugos.*, The Journal of experimental biology **214** (2011), no. 16, 2690–2696.

[20] J. Cacace, A. Finzi, V. Lippiello, G. Loianno, and D. Sanzone, *Aerial service vehicles for industrial inspection: Task decomposition and plan execution*, Recent Trends in Applied Artificial Intelligence (M. Ali et al., ed.), vol. 7906, Springer-Verlag.

[21] ———, *Aerial service vehicles for industrial inspection: Task decomposition and plan execution*, 26th International Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems (Amsterdam), 2013.

[22] ———, *Aerial service vehicles for industrial inspection: Task decomposition and plan execution*, 23th International Conference on Automated Planning and Scheduling, Workshop on Planning and Robotics (Rome), 2013.

[23] ———, *Integrated planning and execution for an aerial service vehicle*, 23th International Conference on Automated Planning and Scheduling, Workshop on Planning and Robotics (Rome), 2013.

[24] ———, *Aerial service vehicles for industrial inspection: Task decomposition and plan execution*, Appliedd Intelligence, Springer (2014).

[25] P. Caitlin, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, *Influence of Aerodynamics and Proximity Effects in Quadrotor Flight*, Proceedings of the International Symposium on Experimental Robotics (Montreal), 2012.

[26] B. Call, R. Beard, C. Taylor, and B. Barber, *Obstacle avoidance for unmanned air vehicles using image feature tracking*, Proceedings of AIAA Conference on Guidance, Navigation, and Control (Keystone, CO), 2006, pp. 1–9.

[27] B. R. Call, *Obstacle avoidance for unmanned air vehicles*, Master's thesis, Brigham Young University, 2006.

[28] R. Castle, G. Klein, and D.W. Murray, *Video-rate localization in multiple maps for wearable augmented reality*, (2008), 15–22.

[29] F. Chaumette and S. Hutchinson, *Visual servo control. I. Basic approaches*, IEEE Robotics & Automation Magazine **13** (2006), no. 4, 82–90.

[30] ———, *Visual servo control. II. Advanced approaches [Tutorial]*, IEEE Robotics & Automation Magazine **14** (2007), no. 1, 109–118.

[31] T. Cheviron, T. Hamel, R. Mahony, and G. Baldwin, *Robust Nonlinear Fusion of Inertial and Visual Data for position, velocity and attitude estimation of UAV*, IEEE International Conference on Robotics and Automation (Rome), 2007, pp. 2010–2016.

[32] M. Cognetti, P. Stegagno, A. Franchi, G. Oriolo, and H.H. Bulthoff, *3-D mutual localization with anonymous bearing measurements*, IEEE International Conference on Robotics and Automation, 2012, pp. 791–798.

[33] A. I. Comport, E. Marchand, M. Pressigout, and F. Chaumette, *Real-time markerless tracking for augmented reality: the virtual visual servoing framework*, IEEE transactions on visualization and computer graphics **12**, no. 4, 615–28.

[34] N. J. Cowan, J. D. Weingarten, and D. E. Koditschek, *Visual servoing via navigation functions*, IEEE Transactions on Robotics and Automation **18** (2002), no. 4, 521–533.

[35] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, *MonoSLAM: real-time single camera SLAM.*, IEEE transactions on pattern analysis and machine intelligence **29** (2007), 1052–1067.

[36] P. Doherty, G. Granlund, K. Kuchcinski, S. Erik, K. Nordberg, E. Skarman, and J. Wiklund, *The WITAS unmanned aerial vehicle project*, 14th European Conference on Artificial Intelligence (Berlin), 2000, pp. 747–755.

[37] P. Doherty, J. Kvarnström, and H. Fredrik, *A temporal logic-based planning and execution monitoring framework for unmanned aircraft systems*, AAMAS (Budapest), 2009, pp. 332–377.

[38] F. Donnarumma, V. Lippiello, and M. Saveriano, *Fast incremental clustering and representation of a 3D point cloud sequence with planar regions*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Vilamoura), 2012, pp. 3475–3480.

[39] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, *Realtime 3d visual slam with a hand-held rgb-d camera*, Workshop on 3D Perception in Robotics at the European Robotics Forum (Vasterås), 2011.

[40] B. Espiau, F. Chaumette, and P. Rives, *A new approach to visual servoing in robotics*, IEEE Transactions on Robotics and Automation **8** (1992), no. 3, 313–326.

[41] A. Finzi and A. Orlandini, *Human-robot interaction through mixed-initiative planning for rescue and search rovers*, AI*IA-05 (Milan), 2005, pp. 483–494.

[42] C. Forster, S. Lynen, L. Kneip, and D. Scaramuzza, *Collaborative Monocular SLAM with Multiple Micro Aerial Vehicles*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Tokyo), 2013.

[43] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, *A Probabilistic Approach to Collaborative Multi-Robot Localization*, Autonomous Robots **8** (2000), 325–344.

[44] F. Fraundorfer and D. Scaramuzza, *Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications*, IEEE Robotics & Automation Magazine **19** (2012), 78–90.

[45] B. Freedman, A. Shpunt, M. Machline, , and Y. Arieli, *Depth mapping using projected patterns*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Taipei), 2010.

[46] J. Gancet, G. Hattenberger, R. Alami, and S. Lacroix, *Task planning and control for a multi-UAV system: architecture and algorithms*, IEEE/RSJ Inernational Conference on Intelligent Robots and Systems (Edmonton), 2005, pp. 1017–1022.

[47] A. Geiger, J. Ziegler, and C. Stiller, *Stereoscan: Dense 3D reconstruction in real-time*, IEEE Intelligent Vehicles Symposium (Karlsruhe), 2011, pp. 963–968.

[48] W. E. Green, P. Y. Oh, K. Sevcik, and G. Barrows, *Autonomous landing for indoor flying robots using optic flow*, ASME International Mechanical Engineering Congress and Exposition (Washington, DC), 2003, pp. 1347–1352.

[49] F. Gustafsson, *Statistical sensor fusion*, 2010.

[50] T. Hamel and R. Mahony, *Attitude estimation on SO[3] based on direct inertial measurements*, IEEE International Conference on Robotics and Automation (Orlando), 2006, pp. 2170–2175.

[51] T. Hamel and R. Mahony, *Image based visual servo control for a class of aerial robotic systems*, Automatica **43** (2007), no. 11, 1975–1983.

[52] T. Hamel and T. Mahony, *Visual Servoing of an Under-Actuated Dynamic Rigid-Body System: An Image-Based Approach*, IEEE Transactions on Robotics & Automation **18** (2002), no. 2, 187–198.

[53] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, *Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments*, International Symposium on Experimental Robotics (ISER) (New Dehli), 2010.

[54] A. Howard, *Multi-robot mapping using manifold representations*, IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 **4** (2004).

[55] S. Hrabar, *Vision-based 3D navigation for an autonomous helicopter*, (2006).

[56] F. H. Hsiao and S.-T. Pan, *Robust kalman filter synthesis for uncertain multiple time-delay stochastic systems*, ASME Journal of Dynamic Systems, Measurement, and Control **15** (1996), 803–808.

[57] S. Hutchinson, G. D. Hager, and P. I. Corke, *A tutorial on visual servo control*, IEEE Transactions on Robotics and Automation **12** (1996), no. 5, 651–670.

[58] H. Jabbari, G. Oriolo, and H. Bolandi, *Dynamic IBVS control of an underactuated UAV*, IEEE International Conference on Robotics and Biomimetics (ROBIO), Ieee, December 2012, pp. 1158–1163.

[59] S. Kay and Y. C. Eldar, *Rethinking biased estimation [Lecture Notes]*, IEEE Signal Processing Magazine **25** (2008).

[60] F. Kendoul, I. Fantoni, and K. Nonami, *Optic flow-based vision system for autonomous 3D localization and control of small aerial vehicles*, Robotics and Autonomous Systems **57** (2009), 591–602.

[61] L. Kneip, A. Martinelli, S. Weiss, D. Scaramuzza, and R. Siegwart, *Closed-form solution for absolute scale velocity determination combining inertial measurements and a single feature correspondence*, IEEE International Conference on Robotics and Automation (Shangai), 2011, pp. 4546–4553.

[62] L. Kneip, D. Scaramuzza, and R. Siegwart, *On the initialization of statistical optimum filters with application to motion estimation*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Taipei), 2010, pp. 1500–1506.

[63] T. D. Larsen, N. A. Andersen, O. Ravn, and N. K. Poulsen, *Incorporation of time delayed measurements in a discrete-time Kalman filter*, 37th IEEE Conference on Decision and Control (Tampa), vol. 4, 1998, pp. 3972–3977.

[64] D. Lee, T. Ryan, and H. Jin. Kim, *Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing*, IEEE International Conference on Robotics and Automation (St. Paul), May 2012, pp. 971–976.

[65] T. Lee, M. Leok, and N.H. McClamroch, *Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)*, Asian Journal of Control (2011).

[66] Q. Lindsey, D. Mellinger, and V. Kumar, *Construction of Cubic Structures with Quadrotor Teams*, Mechanical Engineering (2011).

[67] V. Lippiello, G. Loianno, and B. Siciliano, *Mav indoor navigation based on a closed-form solution for absolute scale velocity estimation using optical flow and inertial data*, IEEE Confenrence on Decision and Control and European Control Conference (Orlando), 2011, pp. 3566–3571.

[68] ———, *Mav obstacles avoidance based on the absolute depth map estimation through the optical flow and inertial data*, 10th International IFAC Symposium on Robot Control, SYROCO (Dubrovnik), 2012.

[69] V. Lippiello and R. Mebarki, *Closed-form solution for absolute scale velocity estimation using visual and inertial data with a sliding least-squares estimation*, 21st Mediterranean Conference on Control and Automation (Crete), 2013, pp. 1261–1266.

[70] V. Lippiello and B. Siciliano, *Wall inspection control of a VTOL unmanned aerial vehicle based on a stereo optical flow*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Vilamoura), 2012, pp. 4296–4302.

[71] V. Lippiello, B. Siciliano, and L. Villani, *Adaptive extended kalman filtering for visual motion estimation of 3d objects*, Control Engineering Practice **15** (2007), 123–134.

[72] G. Loianno, V. Lippiello, C. Fischione, and B. Siciliano, *Visual and inertial multi-rate data fusion for motion estimation via pareto-optimization*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Tokyo), 2013.

[73] G. Loianno, V. Lippiello, and B. Siciliano, *Fast locaztion and mapping using a rgb-d sensor*, IEEE International Conference on Advanced Robotics (Montevideo), 2013, pp. 1–6.

[74] B. D. Lucas and T. Kanade, *An Iterative Image Registration Technique with an Application to Stereo Vision*, International Joint Conference on Artifical Intelligence, vol. 130, 1981, pp. 674–679.

[75] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, Interdisciplinary Applied Mathematics, Springer, New York, 2004.

[76] S. E. Macfarlane and E. A. Croft, *Jerk-bounded manipulator trajectory planning: design for real-time applications*, IEEE Transactions on Robotics **19** (2003), 42–52.

[77] S. O. H. Madgwick, A. J. L.Harrison, and R. Vaidyanathan, *Estimation of IMU and MARG orientation using a gradient descent algorithm*, IEEE International Conference on Rehabilitation Robotics (Zurich), 2011, pp. 1–7.

[78] E. Mair, G. Hager, D. Burschka, M. Suppa, and G. Hirzinger, *Adaptive and Generic Corner Detection Based on the Accelerated Segment Test*, European Conference on Computer Vision (2010), 183–196.

[79] L. Marconi, L. Basile, G. Caprari, R. Carloni, P. Chiacchio, C. Huerzeler, V. Lippiello, R. Naldi, N. Janosch, B. Siciliano, S. Stramigioli, and E. Zwicker, *Aerial service robotics: The AIRobots perspective*, 2nd International Conference on Applied Robotics for the Power Industry (Zurich), 2012, pp. 64–69.

[80] L. Marconi, C. Melchiorri, M. Beetz, D. Pangercic, R. Siegwart, S. Leutenegger, R. Carloni, S. Stramigioli, H. Bruyninckx, P. Doherty, A. Kleiner, V. Lippiello, A. Finzi, B. Siciliano, A. Sala, and N. Tomatis, *The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments*, IEEE Int. Workshop on Safety, Security and Rescue Robotics (SSRR) (College Station), 2012, pp. 1–4.

[81] L. Marconi, R. Naldi, A. Torre, J. Nikolic, C. Huerzeler, G. Caprari, E. Zwicker, B. Siciliano, V. Lippiello, R. Carloni, and S. Stramigioli, *Aerial*

service robots: An overview of the airobots activity, 2nd International Conference on Applied Robotics for the Power Industry (Zurich), 2012, pp. 76–77.

[82] C. McCarthy, N. Barnes, and M. Srinivasan, *Real Time Biologically-Inspired Depth Maps from Spherical Flow*, IEEE International Conference on Robotics and Automation (Rome), 2007, pp. 4887–4892.

[83] D. Mellinger and V. Kumar, *Minimum snap trajectory generation and control for quadrotors*, IEEE International Conference on Robotics and Automation (Shangai), IEEE, May 2011, pp. 2520–2525.

[84] D. Mellinger, M. Shomin, and V. Kumar, *Control of Quadrotors for Robust Perching and Landing*, International Powered Lift Conference (Philadelphia), 2010.

[85] D. Mellinger, M. Shomin, N. Michael, and V. Kumar, *Cooperative grasping and transport using multiple quadrotors*, International Symposium on Distributed Autonomous Robotic Systems (Lausanne), 2010.

[86] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, *The GRASP multiple micro-UAV testbed*, Robotics & Automation Magazine, IEEE **17** (2010), no. 3, 56–65.

[87] A.I. Mourikis and S.I. Roumeliotis, *On the treatment of relative-pose measurements for mobile robot localization*, IEEE International Conference on Robotics and Automation (Orlando), 2006, pp. 2277–2284.

[88] Anastasios I Mourikis, Stergios I Roumeliotis, and Joel W Burdick, *SC-KF Mobile Robot Localization: A Stochastic Cloning Kalman Filter for Processing Relative-State Measurements*, IEEE Transactions on Robotics **23** (2007), no. 4, 717–730.

[89] R.M. Murray, M. Rathinam, and W. Sluis, *Differential flatness of mechanical control systems: A catalog of prototype systems*, ASME International Congress and Exposition, Citeseer, 1995.

[90] R. Naldi, M. Marconi, and L. Gentili, *Modelling and control of a flying robot interacting with the environment*, Journal of IFAC **4** (2011), no. 12, 2571–2583.

[91] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, P. Kohli A. Davison, J. Shotton, S. Hodges, and A. Fitzgibbon, *Kinectfusion: Real-time dense surface mapping and tracking*, International Symposium on Mixed and Augmented Reality (ISMAR) (Basel), 2011, pp. 127 –136.

[92] J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Hurzeler, and R. Siegwart, *A uav system for inspection of industrial facilities*, IEEE Aerospace Conference (AeroConf) (Big Sky), 2013, pp. 1–8.

[93] P. Oguz-Ekim, J. Gomes, J. Xavier, and P. Oliveira, *A convex relaxation for approximate maximum-likelihood 2D source localization from range measurements*, IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP) (Vancouver), 2010, pp. 2698–2701.

[94] R. Rocha, J. Dias, and A. Carvalho, *Cooperative multi-robot systems: A study of vision-based 3-D mapping using information theory*, Robotics and Autonomous Systems **53** (2005), 282–311.

[95] E. Rosten and T. Drummond, *Fusing points and lines for high performance tracking*, IEEE International Conference on Computer Vision, vol. 2, 2005.

[96] F. Ruffier and N. Franceschini, *Aerial robot piloted in steep relief by optic flow sensors*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Nice), 2008, pp. 1266–1273.

[97] D. Scaramuzza and F. Fraundorfer, *Visual Odometry : Part I: The First 30 Years and Fundamentals*, IEEE Robotics & Automation Magazine **18** (2011), 80–92.

[98] D. Scaramuzza, A. Martinelli, and R. Siegwart, *A Toolbox for Easily Calibrating Omnidirectional Cameras*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Beijing), 2006, pp. 5695–5701.

[99] H.A.P. Selvatici and A.H.R. Costa, *Obstacle avoidance using time-to-contact information*, IEEE International Conference on Robotics and Automation (Rome), 2007, pp. 21–28.

[100] J. Serres, D. Dray, F. Ruffier, and N. Franceschini, *A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance*, Autonomous Robots **25** (2008), 103–122.

[101] S. Shen, N. Michael, and V. Kumar, *Autonomous multi-floor indoor navigation with a computationally constrained micro aerial vehicle*, IEEE International Conference on Robotics and Automation (Shangai), 2011, pp. 2968–2969.

[102] ———, *Autonomous indoor 3D exploration with a micro-aerial vehicle*, IEEE International Conference on Robotics and Automation (St. Paul), 2012, pp. 9–15.

[103] D. Shiokata, A. Namiki, and M. Ishikawa, *Robot dribbling using a high-speed multifingered hand and a high-speed vision system*, IEEE/RSJ International Conference on Intelligent Robots and Systems (Alberta), IEEE, 2005, pp. 2097–2102.

[104] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, Springer, 2009.

[105] J. Sola, A. Monin, M. Devy, and T. Vidal-Calleja, *Fusing Monocular Information in Multicamera SLAM*, IEEE Transactions on Robotics **24** (2008).

[106] K. Sreenath and V. Kumar, *Dynamics, Control and Planning for Cooperative Manipulation of Payloads Suspended by Cables from Multiple Quadrotor Robots*, Proceedings of Robotics: Science and Systems (Berlin, Germany), 2013.

[107] K. Sreenath, T. Lee, and V. Kumar, *Geometric Control and Differential Flatness of a Quadrotor UAV with a Cable-Suspended Load*, IEEE Conference on Decision and Control (CDC), to appear (Florence), 2013, pp. 2269–2274.

[108] K. Sreenath, N. Michael, and V. Kumar, *Trajectory Generation and Control of a Quadrotor with a Cable-Suspended Load A Differentially-Flat Hybrid System*, International Conference on Robotics and Automation (karlsruhe), 2013, pp. 4888–4895.

[109] M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett, *Honeybee navigation en route to the goal: visual flight control and odometry*, The Journal of experimental biology **199** (1996), 237–44.

[110] A. Stentz, *Optimal and efficient path planning for unknown and dynamic environments*, Internationl Journal of Robotics and Automation **10** (1995), no. 3, 89–100.

[111] H. Strasdat, J. Montiel, and A. J. Davison, *Real–time monocular slam: Why filter?*, IEEE International Conference on Robotics and Automation (Anchorage), 2010, pp. 2657–2664.

[112] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, *A benchmark for the evaluation of RGB-D SLAM systems*, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (Vilamoura), 2012, pp. 573–580.

[113] L. Taeyoung, M. Leoky, and N.H. McClamroch, *Geometric tracking control of a quadrotor uav on se(3)*, 49th IEEE Conference on Decision and Control (CDC) (Atlanta).

[114] L.F. Tammero and M. H. Dickinson, *The influence of visual landscape on the free flight behavior of the fruit fly Drosophila melanogaster.*, The Journal of experimental biology **205** (2002), 327–343.

[115] C.J. Taylor and J.P. Ostrowski, *Robust vision-based pose control*, IEEE International Conference on Robotics and Automation. Symposia Proceedings, Millennium Conference (San Francisco), vol. 3, IEEE, 2000, pp. 2734–2740.

[116] The Slow Mo Guys, *"Red Kites in Slow Motion", http://youtu.be/AYOx-iCMZhk*.

[117] J. Thomas, G. Loianno, J. Polin, K. Sreenath, and V. Kumar, *Toward autonomous avian-inspired dynamic grasping and perching*, Bioinspiration & Biomimetics (2014).

[118] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, *Toward image based visual servoing for aerial grasping and perching*, IEEE International Conference on Robotics and Automation (Hong Kong), 2014.

[119] J. Thomas, J. Polin, K. Sreenath, and V. Kumar, *Avian-Inspired Grasping for Quadrotor Micro UAVs*, IDETC/CIE 2013 (Portland), 2013.

[120] N. Trawny, S. I. Roumeliotis, and G. B. Giannakis, *Cooperative multi-robot localization under communication constraints*, IEEE International Conference on Robotics and Automation, 2009.

[121] R. Voigt, J. Nikolic, C. Hurzeler, S. Weiss, L. Kneip, and R. Siegwart, *Robust embedded egomotion estimation*, IEEE/RSJ International Conference on Intelligent Robots and Systems (San Francisco), 2011, pp. 2694–2699.

[122] S. Weiss, D. Scaramuzza, and R. Siegwart, *Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments*, Journal of Field Robotics **28** (2011), no. 6, 854–874.

[123] B. Williams, G. Klein, and I. Reid, *Real-Time SLAM Relocalisation*, IEEE 11th International Conference on Computer Vision (Rio De Janeiro), 2007.

[124] M. Zamponi, *Optical flow based navigation*, Master's thesis, ETH Zurich, 2009.

[125] T. Zhang, *Design and performance analysis of a direct adaptive controller for nonlinear systems*, Automatica **35** (1999), no. 11, 1809–1817.

[126] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart, *MAV navigation through indoor corridors using optical flow*, IEEE International Conference on Robotics and Automation (Anchorage), 2010, pp. 3361–3368.

[127] D. Zou and P. Tan, *CoSLAM: collaborative visual SLAM in dynamic environments.*, IEEE transactions on pattern analysis and machine intelligence **35** (2013), 354–66.

[128] J. C. Zufferey, A. Klaptocz, A. Beyeler, J. D. Nicoud, and D. Floreano, *A 10-gram vision-based flying robot*, Advanced Robotics **21** (2007), 1671–1684.