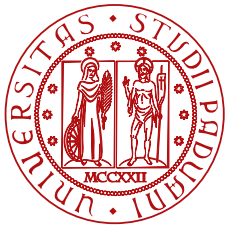


Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Master Thesis in Control System Engineering

Study and development of a reliable fiducials-based localization system for multicopter UAVs flying indoor

Master Candidate

Simone Montecchio

Student ID 2045758

Supervisor

Prof. Angelo Cenedese

University of Padova

Co-supervisor(s)

Prof.ssa Giulia Michieletto

Dr. Massimiliano Bertoni

University of Padova

Academic Year
2022/2023

To my family and my friends.
To everyone who has always believed in me
and supported me along the way

Abstract

The recent evolution of technology in automation, agriculture, IoT, and aerospace fields has created a growing demand for mobile robots capable of autonomous operation and movement to accomplish various tasks. Aerial platforms are expected to play a central role in the future due to their versatility and swift intervention capabilities. However, the effective utilization of these platforms faces a significant challenge due to localization, which is a vital aspect for their interaction with the surrounding environment. While GNSS localization systems have established themselves as reliable solutions for open-space scenarios, the same approach is not viable for indoor settings, where localization remains an open problem as it is witnessed by the lack of extensive literature on the topic.

In this thesis, we address this challenge by proposing a dependable solution for small multi-rotor UAVs using a Visual Inertial Odometry localization system. Our KF-based localization system reconstructs the pose by fusing data from onboard sensors. The primary source of information stems from the recognition of AprilTags fiducial markers, strategically placed in known positions to form a "map".

Building upon prior research and thesis work conducted at our university, we extend and enhance this system. We begin with a concise introduction, followed by a justification of our chosen strategies based on the current state of the art. We provide an overview of the key theoretical, mathematical, and technical aspects that support our work. These concepts are fundamental to the design of innovative strategies that address challenges such as data fusion from different AprilTag recognition and the elimination of misleading measurements. To validate our algorithms and their implementation, we conduct experimental tests using two distinct platforms by using localization accuracy and computational complexity as performance indices to demonstrate the practical viability of our proposed system.

By tackling the critical issue of indoor localization for aerial platforms, this thesis tries to give some contribution to the advancement of robotics technology, opening avenues for enhanced autonomy and efficiency across various domains.

Contents

List of Figures

List of Tables

List of Acronyms

UAV Unmanned Aerial Vehicle

VTOL Vertical Take-Off and Landing

IMU Inertial Measurement Unit

EKF Extended Kalman Filter

VIO Visual Inertial Odometry

GNSS Global Navigation Satellite System

GPS Global Positioning System

SLAM Simultaneous Localization and Mapping**VO!** (**VO!**) Visual Odometry

C.O.M Center of Mass

PID Proportional-Integral-Derivative

LPF Low-Pass Filter

ROS Robot Operating System

FIR Finite Impulse Response

1

Introduction

Throughout the past few decades, robotics has witnessed remarkable growth in interest and advancement, captivating the imaginations of scientists, engineers, and the general public. With industrial robots' controlled precision and unmanned aerial vehicles' autonomous mobility, robotics has expanded its horizons, pushing the boundaries of what was once thought possible. While industrial robotics has succeeded in manufacturing, mobile robotics offers a wide range of potential applications that extend far beyond factory settings. Despite being mainly used in the industrial sector to improve productivity, robots cannot fully replace humans in the future, while could instead "replace their hands". Robotic systems can deal with repetitive, demeaning, dangerous tasks or even perform duties impossible for humans. In terms of workers' safety and well-being, this can have a significant impact.

Industrial robotics and mobile robotics differ in their respective operation domains. While industrial robots perform repetitive tasks with impressive accuracy, speed, and efficiency in factories, mobile robotics has the potential to revolutionize numerous fields, despite its still early stage of development. Powered by sensors, small but powerful onboard computers, and equipped with the ability to navigate complex environments, mobile robots are expected to become a common technology in the near future. Further progress could also be achieved by exploring how these systems can be integrated with emerging artificial intelligence technology and interact with the humans.

Several factors may contribute to the aforementioned growth in the usage of mobile robots. Above all, there has been a significant increase in the demand automated systems and machines, primarily driven by the need for enhanced productivity, efficiency, and safety. Mobile robots present the ability to adapt and perform tasks in ever-changing and challenging conditions, making them valuable in various sectors such as logistics, healthcare, agriculture, and search and rescue operations. Additionally, the progress made in sensor technology, miniaturization, and computing power has allowed the development of compact and intelligent platforms that can function autonomously, effectively tackling the challenges posed by unpredictable environments.

1.1 Localization of aerial mobile robots

The future of robotics holds great significance for small *Unmanned Aerial Vehicle (UAV)s*, often more commonly referred to as drones. These compact and agile flying robots have demonstrated immense potential in various domains and will certainly play a pivotal role in the future.

There exist several types of these vehicles, built for different purposes and to work in the most disparate conditions. The first classification is based on the aerodynamic principle the aircraft exploits. It is fundamental to distinguish between fixed-wings and spinning-wings vehicles (see figure 1.1). The first typology is provided of extended wings able to lift the aircraft in the air thanks to its moving speed, similarly to airplanes. The second type is lifted by the thrust generated by one or more propellers. If the vehicle presents a main propeller responsible for the thrust and the others are used just to compensate for drag forces acting on the platform, then it is categorized as a helicopter. Otherwise, if three or more identical rotors are mounted respecting proper geometrical requirements, the aircraft is a multi-copter. More recently, some fixed-spinning hybrid systems have been also proposed and studied, referred to as *Vertical Take-Off and Landing (VTOL)* to bring together the main advantages of the two configurations. However, the most used and widespread UAVs remain for now the multi-copters, thanks to their simple structural design, versatility and relatively low costs, making them suitable for a large number of applications.

Figure 1.1: Small UAVs classification based on the propulsion system.

Several standards for multi-copter exist, differing for the number of rotors/propellers and their geometric arrangement (some common configurations are reported in figure 1.2).

UAVs are already in use for various tasks in the most disparate sectors in outdoor scenarios (see for instance [4], [12], [15]), such as (figure 1.3):

- aerial shooting and recordings;
- surveillance;
- light cargo delivery;
- Internet services;
- precision agriculture.

However, multi-copters could be particularly useful also in restricted space environments like indoor settings, where the utilization of small UAVs presents compelling opportunities for a wide range of applications. Consider the possibilities of autonomous inspection and monitoring tasks within complex infrastructures such as warehouses, factories, or even disaster-stricken buildings, where human access is limited or hazardous. Small UAVs can navigate confined spaces, capture high-resolution imagery, and relay critical information to aid in decision-making processes. Furthermore, their potential ability to perform tasks with precision and efficiency opens avenues for applications in areas such as inventory management, environmental sensing, security, surveillance, and even

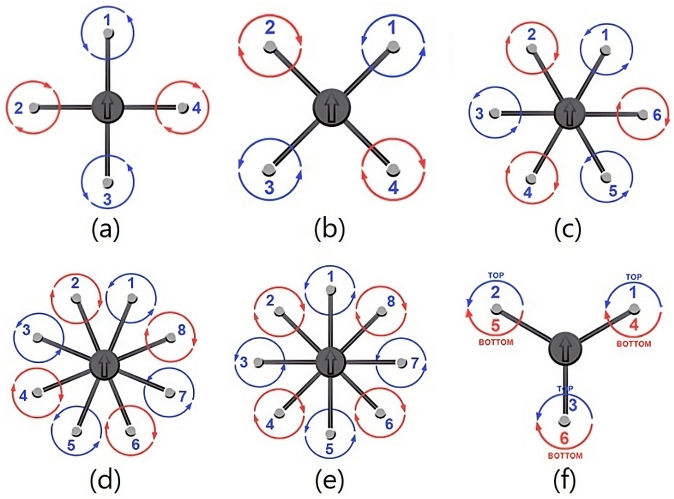


Figure 1.2: Typical commercial multi-copter configurations: (a) I-quad, (b) X-quad, (c) X-hexa, (d) X-octa, (e) I-octa, (f) Y6-hexa.

Figure 1.3: Modern scenarios involving the use of UAVs.

entertainment and immersive experiences. By harnessing the potential of small UAVs in restricted space environments enhance productivity, safety, and the overall human-machine interaction in diverse sectors. For these reasons, multi-rotor UAVs can be considered as the most representative instance of aerial robots able to operate indoors, becoming the main actors for this thesis work.

In general, robotics can be regarded as the scientific study of the interplay between perception and action in autonomous systems. Robots serve as platforms to apply fundamental principles from control system theory, notably emphasizing the significance of closed-loop systems. In the case of mobile robots, particularly UAVs, awareness of their own position and the location of their target is crucial for task fulfilment. These autonomous systems are typically confined within controlled and secure workspaces. Such closed environments often impose narrow spaces and limited maneuvering capabilities, necessitating precise navigation. This becomes even more critical when these robots are designed to collaborate with human beings or operate in sparsely occupied environments. Moreover, for tasks requiring high precision, the robot must possess accurate knowledge of its intended position and the relative pose with respect to its target. In this context, localization emerges as a key aspect that addresses these challenges. Localization is defined as the process of determining the precise position and orientation of a robot relative to its surroundings, representing a fundamental aspect of robotics. A robust self-localization and mapping system is indispensable, as it allows the robot to operate effectively and efficiently in its environment. However, indoor environments pose unique challenges due to the absence of GPS signals, the presence of obstacles, and the requirement for precise and reliable localization in confined spaces.

Typically, all the aerial vehicles are equipped with *Inertial Measurement Unit (IMU)* and possibly barometers. These sensors can be used to approximately estimate the position and the orientation of the aircraft with respect to a fixed reference system. However, since this calculation is performed by integration, even very small errors in the measurements can produce a drift effect on the estimate, resulting in a completely wrong localization of the platform. This problem can be overcome by adding another source of information to face the drift effect introduced by the IMU. Modern aerial robots implement a state estimation system based on an *Extended Kalman Filter (EKF)*, i.e. an enhanced Kalman Filter

adapted to deal also with non-linear dynamic systems. This filter can merge data coming from different sensors (data fusion) providing a robust estimate of the pose. Normally, this second source of measurement consists of a Global Positioning System (GPS) signal. Since this signal is known to quickly degrade in quality and precision when used indoors, a different method must be adopted in similar situations.

1.2 State of the art

To enable autonomous movement and operation, a multi-rotor UAV must continually measure or estimate its position and orientation relative to a fixed reference frame within its workspace, thereby achieving real-time localization. Extensive literature has addressed the localization problem, proposing various approaches. However, the majority of studies have predominantly focused on outdoor scenarios, where *Global Navigation Satellite System (GNSS)* solutions have become the prevailing choice. In contrast, the challenge of localizing robots operating in indoor environments, where GNSS signals are not feasible due to temporary signal loss or degradation, remains an insufficiently explored domain. Presently available technologies struggle to offer the high performance and reliability required for precision-demanding indoor tasks, thereby leaving the problem of localization within small, enclosed spaces as an open challenge that must be overcome for the maturation of multi-rotor UAV technology.

Numerous alternative sources of pose information have been proposed in the literature as substitutes for GNSS. Early works considered acoustic [23] and radio frequency [35] signals, as well as localization methods based on visible light [30]. However, the standalone application of these localization methods demonstrated notably low accuracy, limiting their practical utility to a narrow range of applications. More recent approaches have turned to image/vision systems, which have gained significant popularity in the literature ([36], [37]), owing to their increased accuracy and robustness. Vision systems can be mounted onboard or off-board. Off-board solutions typically entail the use of cost-effective motion capture camera systems, which offer the highest achievable level of accuracy in determining the pose of a mobile platform within their field of view. Despite their performance advantages, these localization systems are generally impractical for deployment outside laboratory settings because they require an im-

portant infrastructure and periodic calibrations, aside from their valuable cost. Consequently, their predominant usage remains within academic and research contexts, serving as ground truth for comparative analyses against other localization methods, as exemplified in this thesis work.

For these reasons, the focus is primarily directed towards onboard visual systems. In this regard, visual-based systems have been employed in conjunction with other sensor measurements, such as lidars, to implement the *Simultaneous Localization and Mapping (SLAM)* algorithm, which has proven particularly effective for indoor navigation tasks [7]. Another common approach involves the utilization of fiducial markers [18] strategically designed and placed in advance within the robot's workspace at pre-determined positions. This approach was initially proposed for precise autonomous outdoor landing, requiring higher accuracy than that provided by GNSS systems, utilizing custom fiducial markers, as demonstrated in [16]. However, more frequently, these fiducial markers belong to well-established standardized families, such as for the approach described in [33], where it is part of a comprehensive visual-servo UAV landing control strategy. Fiducial markers have also been cleverly employed in [22] for ground robot tracking by a UAV and in [38] for tracking another UAV, with the marker directly attached to the target. Moreover, fiducial markers have been employed in [8] to address the *Autonomous Valet Parking (AVP)* problem, an ongoing challenge in autonomous driving. Although the adoption of standardized marker families in UAV localization remains relatively uncommon, the recent literature predominantly discusses AprilTags [27], a widely debated type of fiducial marker. AprilTags offer a robust digital coding system, along with increased resistance to occlusion, warping, and lens distortions. In a recent work [2], AprilTags are employed to tackle indoor navigation for multi-rotor UAVs, with a specific focus on developing a new closed-loop control system directly utilizing visual odometry readings as feedback.

In [13], the utilization of AprilTags to establish a localization system for an autonomous ground robot navigating within a confined environment has been explored. All fiducial markers are strategically positioned on the ceiling, ensuring continuous visibility for a camera mounted atop the robot. Leveraging AprilTags detection in conjunction with SLAM algorithms, the ground robot achieves highly accurate navigation throughout the environment, exhibiting precision in

every maneuver. However, when applying the same approach to UAVs, new challenges arise due to the potential variations in flight altitudes. If an UAV platform is not constrained to a fixed height, the recognition of small markers becomes increasingly challenging as the distance from the camera increases.

To address the issue of different flight altitudes, several strategies have been proposed. In [34], the problem is tackled by incorporating multiple smaller markers within a larger marker or employing distinct fiducial markers with different sizes [21], [3]. In [39], a ground map generated from a simple and repetitive pattern of equally sized tags is proposed, requiring the UAV to perform a complete flight at a constant altitude. Position data is processed using a Kalman Filter to enhance precision and mitigate noise affecting measurements. Similarly, [26] adopts a framework where a scattered set of fiducial markers (AprilTags from a standardized family) is strategically placed in predetermined positions within the area, replacing the need for a ground map. Building upon this work, [20] improves the localization performance by employing a Visual Inertial Odometry (VIO) approach, that relies on a (EKF) to fuse IMU measurements with marker recognition data.

More recent and advanced solutions involve this type of data fusion from various sources of localization. Given that both cameras and IMUs are typically mounted on UAVs due to their cost-effectiveness and versatility, several studies have pursued the fusion of data from these sources. This approach, commonly referred to as *VIO*, represents an ideal candidate for indoor localization across a wide range of autonomous platforms. The combination of these two distinct sources of information allows for mitigating their respective limitations. The errors introduced by IMU measurements can be rectified by leveraging data from visual sensors, while operative limits to the UAV flight altitude can be overcome by relying on two complementary sources of data. Drawing inspiration from some of these last works, Bertoni et al. [5] adopted a similar yet enhanced approach for two custom multi-rotor aerial robots. Their system enables the robots to follow prescribed trajectories, incorporating various altitudes while relying on a dense AprilTags map consisting of markers of different sizes. On the software side, localization algorithms are implemented within a *ROS2* environment, ensuring a high level of abstraction and code separation across different nodes.

1.3 Objectives and contents summary

The final objective of this thesis is to develop and validate a reliable **VO!** localization system for multi-copters able to ensure good performances inside a small free-space indoor environment.

The localization system considered and implemented in this thesis is VIO pose estimator based on the recognition of a-priori known fiducial markers placed inside the workspace to form a "map" that guides the UAVs inside the environment. Among all the types of fiducial markers, AprilTags are chosen due to their simplicity and proven precision [18]. Two different models of multi-copters have been formerly developed at the *C-Square (Computer and Control Engineering laboratory)* inside the *Dipartimento di Tecnica e Gestione dei Sistemi Industriali (DTG)*, headed by the University of Padova. These drones, used for experimental tests, are provided by a Flight Controller implementing the EKF filter to merge data from IMU and the *AprilTag recognition system*.

This work can be intended as the natural pursuance of the former work conducted by Pescante [29], which was in turn strongly based on the first work made by Segattini [32]. In [32] it is exhaustively discussed the design and development of a multi-copter (called *HR01*) adopted for all the following research on this topic, along with a quad-copter (referred to as *QR01*). In the same thesis, a first simple VIO system is proposed, which relies on an EKF to integrate the visual data coming from the recognition of the bigger-size tag on the field with the measurements coming from inertial odometry sensors. In [29], the localization system is extended and improved, making it able to collect data from more than one tag per time, and introducing other algorithms to clean and filter the input data. A simple moving average filter is also added to the localization pipeline to reduce the noise at high frequency. These improvements made the resulting state estimation more robust to noise and errors due to bad recognition of the tags, as the experimental tests performed at the final part of the research confirmed. The test methodology adopted by Pescante [29] is inspired by that designed and followed by Bertoni et al. in [5].

In this thesis, the same problem is again addressed, investigating new strategies or methods that could improve the localization system itself. The objec-

tive is to design and implement new algorithms that can improve in practice the results achieved previously. Whenever this appears to be not possible, the willingness is to show (even possibly prove) why the already adopted strategies represent the best choice, at least relying on the current literature.
The above-mentioned improvements can be intended both in terms of the quality of the pose estimation and the efficiency of the algorithms. To this intent, completely new algorithms have been introduced and other ones upgraded. In particular, the problem of effectively merging the information of the pose coming from the recognition of different markers (data fusion) and the problem of excluding less-reliable estimates from the data set (outliers detection and removing) are studied in deep, by comparing different alternative solutions.
Regarding the pose estimation, the goodness is evaluated considering accuracy and robustness, comparing them with those achieved in the earlier works. To this purpose, the methodology for the tests and the experimental setup adopted in [5] are here again replicated or openly taken as a reference. For what concern the execution speed, this aspect was never addressed before, therefore it will be presented here, also by briefly adding performance evaluation of the previous methods to establish a comparison.

The following chapters discuss the theoretical foundations, methodologies, development and improvement of such localization systems culminating in practical insights and recommendations for future developments.

Chapter 2 starts by illustrating the multi-copter mathematical model and its pose representation, fixing a reference notation used along the rest of the thesis. Then, it follows an overview of various other useful technical details on the specific hardware, including the VIO system.

Chapter 3 contains the main contribution of this thesis, with the analysis of several new strategies and algorithm candidates to improve the former work. A preliminary statistical analysis is very briefly discussed, just the promising methods are then tested in the experimental setup.

Chapter 4 describes the experimental setup, including the flight trajectory for the two drones and the performance indices adopted. The results of all the various tests are thereafter reported and discussed, with a particular focus on the

improvements achieved.

Chapter 5 finally summarizes all the evidence coming from the tests and briefly discusses the achieved goals. The thesis ends up with some ideas for future works on this topic.

2

Theoretical and technical background

2.1 Modelling concepts

In order to design an effective localization system, a fundamental prerequisite is to gain a comprehensive understanding of pose representations (i.e. both position and orientation) of a generic UAV platform, followed by the development of a streamlined mathematical model to describe its dynamics and kinematics. Equally important is revisiting the process of sensor fusion that ensures a cohesive integration of data.

2.1.1 UAV pose representations

In order to describe the vehicle's pose accurately within the 3D space, the introduction of two distinct reference frames becomes imperative. The first is the Body Frame $\mathcal{F}_B$, with its origin $\mathbf{0}_b$ coinciding precisely with the Center of Mass (C.O.M) of the vehicle. The second is the World Frame $\mathcal{F}_W$, where the origin $\mathbf{0}_b$ is a fixed and known position in the working environment. While the platform's position is expressed through a 3-dimensional vector identifying the the position of $\mathbf{0}_b$ in $\mathcal{F}_W$, the orientation of a generic UAV, corresponding to the relative orientation between $\mathcal{F}_B$ and $\mathcal{F}_W$ can be mainly represented in three different ways, each possessing its advantages and limitations.

Rotation Matrices

Rotation matrices are widely employed representations for a rigid body's pose across diverse fields, spanning robotics, computer vision, and mechanics. Following this convention, the vehicle's pose is denoted by a pair $(\mathbf{p}_\mathbf{B}^\mathbf{W}, \mathbf{R}_\mathbf{B}^\mathbf{W})$, where $\mathbf{p}_\mathbf{B}^\mathbf{W} \in \mathbb{R}^3$ and $\mathbf{R}_\mathbf{B}^\mathbf{W} \in \mathbb{SO}(3)$. Alternatively, the pose can be described entirely by a transformation matrix $\mathbf{T}_\mathbf{B}^\mathbf{W} \in \mathbb{SE}(3) = \mathbb{R}^3 \times \mathbb{SO}(3)$. Opting for the algebraic group $\mathbb{SO}(3)$ offers a robust representation, devoid of singularity and ambiguity concerns. However, due to its redundant nature and computationally intensive demands, it may not be the most suitable choice for mobile agent orientations.

Rotational angles

Rotational angles offer an alternative approach for representing the vehicle's pose. Under this convention, the pose is characterized by the pair $(\mathbf{p}_\mathbf{B}^\mathbf{W}, \alpha_\mathbf{B})$, with $\alpha_\mathbf{B} \in (\mathbb{S}^1)^3 \subset \mathbb{R}^3$ expressed in radians. This representation is minimalistic and intuitive. The three angles can be interpreted as Euler Angles, representing the magnitudes of three elemental rotations around the body frame axes (typically in sequences ZYX or ZYZ), culminating in the desired overall rotation. Alternatively, they can be intended as Aeronautical Angles (*pitch*, *roll*, *yaw*), directly describing the vehicle's rotational angles around fixed axes centered in $\mathbf{0}_b$. Nevertheless, rotational angles may present certain challenges. Euler Angles heavily rely on the chosen rotation sequence and can encounter singularities, leading to unfortunate configurations where the angles are not uniquely determined—a situation known as *gimbal lock*. Even when opting for aeronautical angles, ambiguity issues may arise due to sign conventions, making rotational angles less suitable for advanced applications.

Unit quaternions

Quaternions are hyper-complex numbers denoted as $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$ and following the so-called Hamilton rule [9], . When restricting our focus to unit norm quaternions, this set becomes isomorphic to the $\mathbb{S}^3$ sphere and effectively represents rotations within three-dimensional space. Consequently, the pose is uniquely identified by the pair $(\mathbf{p}_B^W, \mathbf{q}_B^W)$, where $\mathbf{q}_B^W \in \mathbb{S}^3$. Despite lacking an intuitive interpretation, the quaternion representation offers significant advantages. Notably, it is highly convenient due to its absence of singularities, in

contrast to other representations. Moreover, quaternion operations entail lower computational complexity compared to those involving rotation matrices, given that quaternions are four-element entities as opposed to nine-element objects.

Given the aforementioned considerations, the adoption of quaternions to represent the platform's rotation emerges as an appropriate choice in this work. On the other hand, rotation matrices will find utility in describing certain equations and for specific operations, even at the implementation level. However, as will be explored later, transitioning from $\mathbb{SO}(3)$ to $\mathbb{S}^3$ can lead to problems in certain contexts due to the Double Coverage Property of quaternions. Specifically, for any $\mathbf{R} \in \mathbb{SO}(3)$, there exist two quaternions $\mathbf{q}$ and $-\mathbf{q} \in \mathbb{S}^3$ that both represent the same rotation (see [14]). This characteristic of quaternions can introduce ambiguity and needs careful consideration to avoid potential issues.

2.1.2 Standard multi-rotor modelling

Different models for the multi-rotor UAV have been presented in the literature. Here, the cinematic and dynamical model of a standard n-rotor, viewed as a rigid body equipped with n propellers spinning around their own axes, is described using the Newton-Euler approach.

From a kinematic perspective, the pair $(\mathbf{v}_W, \boldsymbol{\omega}_W)$ represents the *Twist* of the platform, comprising the linear velocity $\mathbf{v}_W = \dot{\mathbf{p}}_W$ of $\mathbf{0}_b$ expressed with respect to $\mathcal{F}_W$, and the angular velocity $\boldsymbol{\omega}_W$ of $\mathcal{F}_b$ with respect to $\mathcal{F}_W$. The relation $\hat{\mathbf{R}}_B^W = \mathbf{R}_B^W[\omega]_x$ governs the rotational aspect.

From a dynamics viewpoint, the total forces and torques applied to the vehicle can be obtained by considering the action of each propeller individually. The i-th propeller, $i = 1 \dots n$, rotates about its own spinning axis $\hat{\mathbf{u}}_{z_i} \in \mathbb{R}^3$ passing through the propeller's centre $\mathbf{0}_i$ at a speed ω_i controlled by the motors. Propellers are labelled as either *clockwise (CW)* or *counter-clockwise (CCW)* as shown in figure 2.1. By convention, the sign of the angular velocity is

$$\begin{cases} -\omega_i \hat{\mathbf{u}}_{z_i} & \text{for CW propellers} \\ -\omega_i \hat{\mathbf{u}}_{z_i} & \text{for CCW propellers} \end{cases}$$

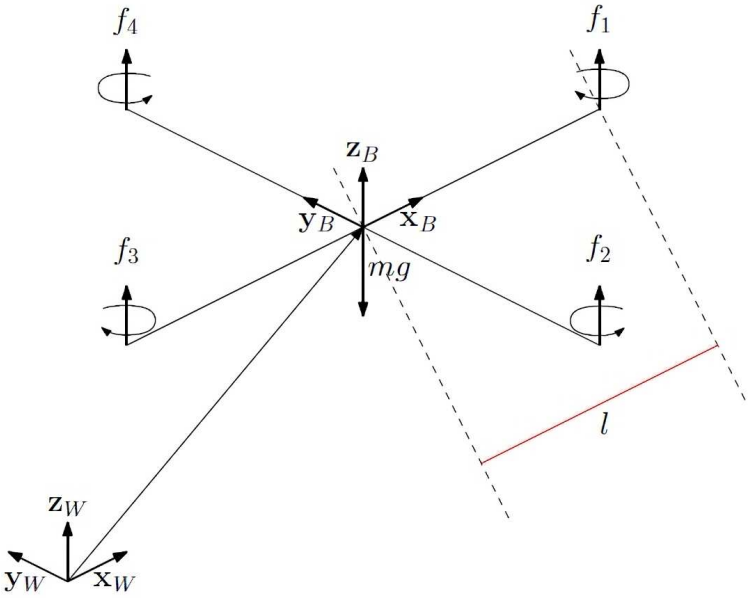


Figure 2.1: Reference frame convention adopted.

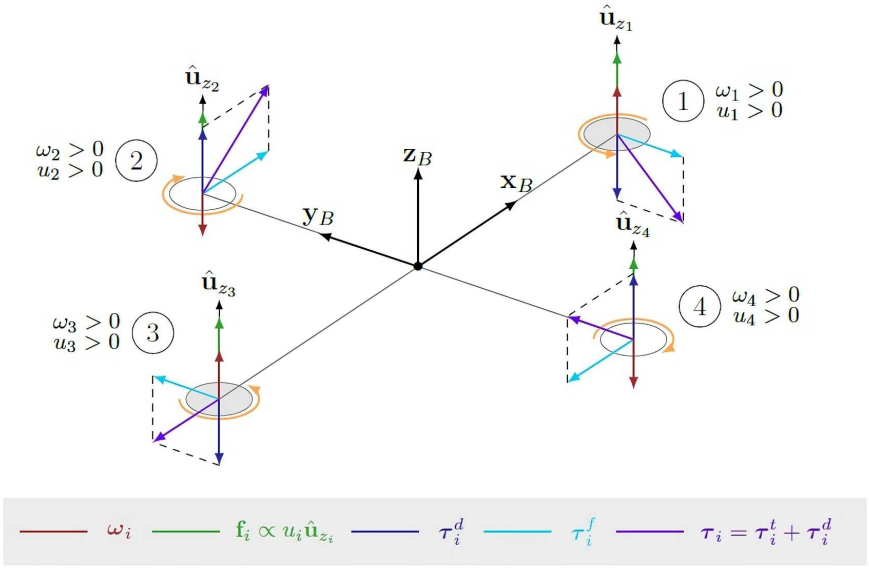


Figure 2.2: Forces and torques acting on the UAV platform.

and the control variable is defined as $u_i = \omega_i|\omega_i| \in \mathbb{R}$. With reference to figure 2.2, each actuator generates

- a thrust force $\mathbf{f}_i = c_{f_i} u_i \hat{\mathbf{u}}_{z_i} \in \mathbb{R}^3$ expressed in $\mathcal{F}_b$, where $c_{f_i} \in \mathbb{R}$ is a constant parameter determined by the mechanical characteristics of the propeller;
- a drag moment $\tau_i^d = c_{\tau_i} u_i \hat{\mathbf{u}}_{z_i} \in \mathbb{R}^3$ expressed in $\mathcal{F}_b$ with direction opposite to $\vec{\omega}_i$ where $c_{\tau_i} \in \mathbb{R}$ is a constant parameter;
- a thrust moment $\tau_i^t = \mathbf{p}_i \times \mathbf{f}_i \in \mathbb{R}^3$.

By aggregating all forces and torques generated by the propellers, the total force $\mathbf{f}_c$ and the total moment $\boldsymbol{\tau}_i$ applied at $\mathbf{0}_b$ and expressed in $\mathcal{F}_b$ are given by

$$\mathbf{f}_c = \sum_{i=1}^{n} \mathbf{f}_i = \sum_{i=1}^{n} c_{f_i} \hat{\mathbf{u}}_{z_i} u_i \quad \text{in } \mathcal{F}_W \tag{2.1}$$

$$\tau_c = \sum_{i=1}^{n} (\tau_i^t + \tau_i^d) = \sum_{i=1}^{n} (c_{f_i} \mathbf{p}_i \times \hat{\mathbf{u}}_{z_i} + c_{\tau_i} \mathbf{u}_{z_i}) u_i \quad \text{in } \mathcal{F}_W \tag{2.2}$$

A shortened notation can be obtained introducing the control input vector $\mathbf{u} = [u_1 \dots u_n]^T \in \mathbb{R}^n$ and two matrices: the control force input matrix $\mathbf{F}$ and the control moment input matrix $\mathbf{M}$, such that $\mathbf{f}_c = \mathbf{F}\mathbf{u}$ and $\tau_c = \mathbf{M}\mathbf{u}$. Neglecting all the second order effects the dynamics equations are

$$m\ddot{\mathbf{p}}_W = -mg\mathbf{e}_3 + \mathbf{R}_B^W \mathbf{f}_c = -mg\mathbf{e}_3 + \mathbf{R}_B^W \mathbf{F}\mathbf{u} \tag{2.3}$$

$$\mathbf{J}\dot{\omega} = -\omega \times \mathbf{J}\omega + \tau_c = -\omega \times \mathbf{J}\omega + \mathbf{M}\mathbf{u} \tag{2.4}$$

where $g \in \mathbb{R}$ is the gravitational acceleration, $m \in \mathbb{R}$ is the total mass of the multi-rotor and $\mathbf{J} \in \mathbb{R}^{3\times3}$ its matrix of inertia. Finally, willing to move to the quaternion representation for the pose, by employing quaternion algebra, the final equations of the model are obtained as follows:

$$\dot{\mathbf{p}}_W = \mathbf{v} \tag{2.5}$$

$$\dot{\mathbf{q}}_B^W = \frac{1}{2} \mathbf{q}_b^W \circ \begin{pmatrix} 0 \\ \omega \end{pmatrix} \tag{2.6}$$

$$m\ddot{\mathbf{p}}_W = -mg\mathbf{e}_3 + \mathbf{R}(\mathbf{q})\mathbf{F}\mathbf{u} \tag{2.7}$$

$$\mathbf{J}\dot{\omega} = -\omega \times \mathbf{J}\omega + \mathbf{M}\mathbf{u} \tag{2.8}$$

where $\circ$ denotes the quaternion (or Hamilton's) product.

2.1.3 Extended Kalman filter and Sensors data fusion

The Kalman Filter plays a fundamental role in robots' localization and navigation. This versatile filtering technique is extensively utilized, even with non-linear systems, through its extended version (EKF), which enables data fusion from various sources of measurements.

In general terms, the Kalman filter is employed to achieve an optimal estimate of a variable that can only be indirectly measured and/or is subject to noisy disturbances. Introduced by Kalman in [19], it swiftly became a standard tool for movement tracking and navigation tasks. The filter combines predictions derived from the system model's knowledge with the noisy measurements obtained from available sensors. Under certain mathematical requirements concerning the system model, this approach ensures an optimal estimate of the unknown variable, as in this case, the vehicle pose.
The main equations of the Kalman filter stem two phases known as the *prediction phase* and the *update (or correction) phase*. During the prediction phase, the estimates from the previous time instances are projected to the next time instant using the provided system model, generating a new estimate of the pose and a new covariance matrix of the error. In contrast, the subsequent update phase involves the integration of the estimate obtained in the prediction step with the measurements collected from the sensors, resulting in a new "corrected" estimate of the pose and the error covariance matrix. This iterative mechanism empowers the estimator to adapt and correct itself continually, converging asymptotically to an optimal estimate of the vehicle's pose, which constitutes part of the system's state.

The standard Kalman filter is specifically designed to handle linear systems affected by white Gaussian noise. The EKF represents its natural evolution, intended to address non-linear systems with measurements affected by more general types of disturbances. While the core algorithm of the EKF follows the same principles as the standard version, there is one crucial difference: the matrices describing the state transition and noise of the model are linearized prior to the prediction phase, and in the subsequent update step, certain functions are linearized using the Jacobian matrix.
The EKF's versatility also enables *sensors fusion*, allowing for the combination

of measurements from different sensors, even operating at different sampling frequencies. When multiple sensors provide measurements of (or some components of) the state, this strategy involves evaluating the Kalman Gain contribution independently for each sensor, utilizing distinct measurement matrices and covariance matrices. Subsequently, the final Kalman gain is determined by combining the contributions from all the sensors, ensuring consistent updates of estimates and covariances.
In the context of this localization system, this approach is employed to merge the IMU measurements with the pose localization obtained from visual odometry. This integration takes place inside the Pixhawk firmware, as elucidated in [*].

2.2 Visual Inertial Odometry

The heart of this thesis project revolves around the Visual Inertial Odometry (VIO) localization system. This section elucidates the complexities of the localization solution, encompassing the process of converting sensor data into precise position and orientation measurements.

2.2.1 IMU sensors and Inertial Odometry

An Inertial Measurement Unit (IMU) is a device equipped with multiple inertial sensors that enable the tracking of a system's dynamics, capable of moving with a specific number of degrees of freedom. IMUs are categorized based on the number of sensors they incorporate, with options ranging from three-axis, six-axis, to even nine-axis devices.

The current IMU utilized on the two reference platforms is a six-axis IMU, integrated with the *PixHawk flight controller*. This device comprises a three-axis gyroscope and a three-axis accelerometer, providing measurements of angular velocity and linear acceleration, respectively. These measurements are always referenced to the body frame $\mathcal{F}_b$. Additionally, the IMU includes a barometer, serving the purpose of obtaining a rough altitude measurement. However, for this localization system, the barometer has been deemed non-essential and consequently left unused.

The measurements acquired from accelerometers and gyroscopes can be described by the following equations:

$$\hat{\omega}_b(t) = \omega_b(t) + \mathbf{b}_\omega + \mathbf{w}_\omega(t) \tag{2.9}$$

$$\hat{a}_b(t) = (\mathbf{R}_B^W)^T(\ddot{\mathbf{p}}_b(t) - g\mathbf{e}_3) + \mathbf{b}_a + \mathbf{w}_a(t) \tag{2.10}$$

where $\hat{\omega}_b(t)$ and $\hat{a}_b(t)$ represent angular velocity and linear acceleration, respectively, expressed with respect to $\mathcal{F}_b$ and measured at time t. The terms $\omega_b(t)$ and $\ddot{\mathbf{p}}_b(t)$ stands for the true values of angular velocity and linear acceleration, while $\mathbf{b}_\omega$ and $\mathbf{b}_a$ represent static bias. The vectors $\mathbf{w}_\omega(t)$ and $\mathbf{w}_a(t)$ are time-varying three-dimensional noise vectors drawn from the distribution $\mathcal{N}(0, \Sigma)$.

As mentioned earlier, counting on IMU measurements only for localization would not be acceptable, primarily due to the presence of drifting phenomena. Therefore, an additional source of localization is required to correct the estimates through Kalman fusion.

2.2.2 Visual Odometry and Fiducial Markers

The measurements recorded by the inertial sensors needs to be fused with the data sourced from visual sensors. The system architecture in use comprises a sole onboard camera (*Raspicam2* for QR01, *Intel Realsense D435* for HR01) that is directed downward towards the floor. Intrinsically, capturing the three-dimensional movement of an object in space solely from the two-dimensional information given by the camera is unfeasible. This is the motivation behind *Monocular Visual Odometry (VO) systems* such as this one, necessitating the identification of *fiducial markers* i.e. pre-positioned elements within the surrounding environment.

Let's introduce an additional reference frame $\mathcal{F}_C$, centered at $\mathbf{0}_C$, located in the on-board camera's Center of Mass. The vision algorithms that underpin the fiducial marker detector initially find, for each recognized marker T_i, with $i = 1 \dots n$ in the image at each time instance t, a relative pose $(\hat{\mathbf{p}}_{T_i}^C, \hat{\mathbf{R}}_{T_i}^C)$, where $\hat{\mathbf{p}}_{T_i}^C \in \mathbb{R}^3$, $\hat{\mathbf{R}}_{T_i}^C \in \mathbb{SO}(3)$ delineate the transformation from the frame $\mathcal{F}_{T_i}$ centered in the top center of the marker to the camera frame $\mathcal{F}_C$. This is achieved by employing the perspective view of the detected markers on the image plane (p-plane), along with the information regarding the known positions of the markers in $\mathcal{F}_W$. This

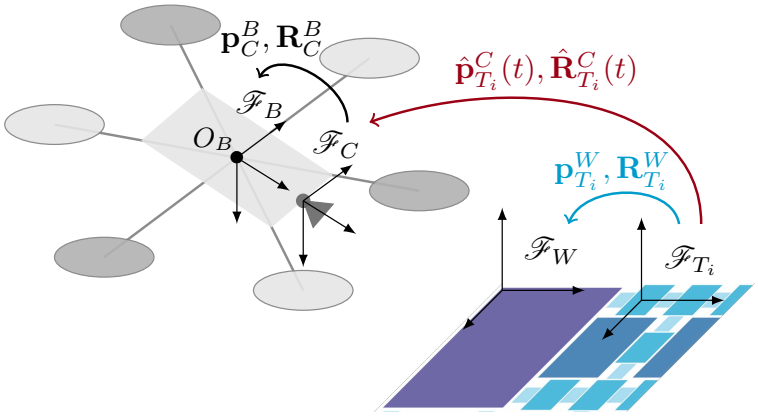


Figure 2.3: Reference frames and transformations involved in the VIO localization process.

computational challenge is acknowledged in the literature as the Perspective-n-Point (PnP) problem (see [28]). Subsequently, the fiducial marker detector is able to supply the camera's pose $(\hat{\mathbf{p}}_C^W, \hat{\mathbf{R}}_C^W)$ through the following equations:

$$\hat{\mathbf{R}}_C^W(t) = \mathbf{R}_{T_i}^W \cdot \hat{\mathbf{R}}_C^{T_i}(t) \tag{2.11}$$

$$\hat{\mathbf{p}}_C^W(t) = \mathbf{R}_{T_i}^W \cdot \hat{\mathbf{p}}_C^{T_i}(t) + \mathbf{p}_{T_i}^W \tag{2.12}$$

Wherein, $\hat{\mathbf{R}}_C^{T_i}(t) = (\hat{\mathbf{R}}_{T_i}^C(t))^{-1}$ and $\hat{\mathbf{p}}_C^{T_i}(t) = -(\hat{\mathbf{R}}_{T_i}^C(t))^{-1} \cdot \hat{\mathbf{p}}_{T_i}^C(t)$ are computed by inverting the affine transformation from marker to camera.

Furthermore, since the relative position $\mathbf{p}_C^B$ and orientation $\mathbf{R}_C^B$ between the body frame and the camera frame are static and known, it becomes feasible to reconstruct the complete pose as follows:

$$\hat{\mathbf{R}}_B^W(t) = \hat{\mathbf{R}}_C^W(t) \cdot (\mathbf{R}_C^B)^{-1} \tag{2.13}$$

$$\hat{\mathbf{p}}_B^W(t) = \hat{\mathbf{R}}_C^W(t) \cdot \mathbf{p}_C^B + \hat{\mathbf{p}}_C^W \tag{2.14}$$

All the considered reference frames and their corresponding relative transformations are illustrated in Figure 2.3.

Fiducial markers that comprise an identifier encoded into their shape or pattern, irrespective of their specific form, are commonly referred to as Tags. Among the multitude of fiducial marker types proposed in academic literature, the *AprilTags* [27] have been adopted due to their well-established precision, dependabil-

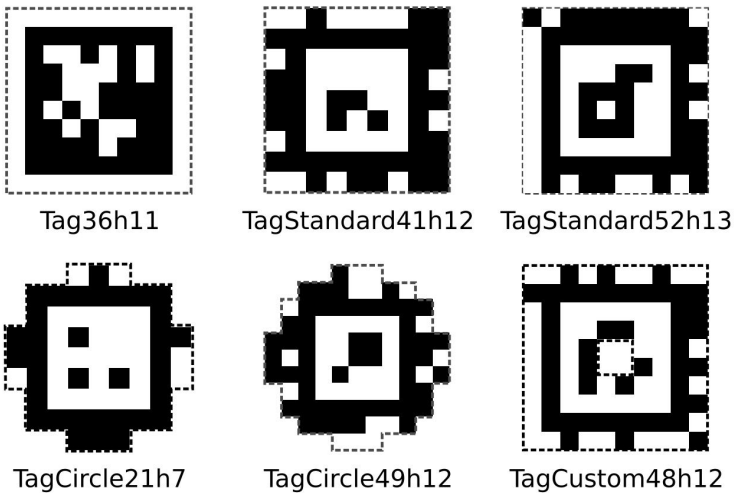


Figure 2.4: Different existing AprilTags families and their patterns.

ity, and rapid recognition capabilities across diverse scenarios (as evidenced in [17]), in contrast to other widely-known fiducial marker solutions like *ARtag* and *ArUco*. AprilTags comprise different families, differentiated by various parameters, including the number of bits allocated for encoding and their distribution. The AprilTags library features an algorithm for generating patterns for a set of tags. These patterns encompass black-and-white graphical codes set within square matrices that are enclosed by a black border (as depicted in Figure 2.4).

The process underlying pose estimation using AprilTags encompasses two principal stages:

Detection The algorithm analyze the image captured by the camera sensor to detect elements conforming to the AprilTag model. This is typically accomplished through techniques like image segmentation and/or thresholding.

Identification and decoding Upon detecting a potential AprilTag (or multiple), the algorithm tries to to extract information from the image, including the ID encoded within it.

After that, if the ID of the recognized tags were registered, the known positions of the tags in the workplace are retrieved and used for the pose estimation process (2.11-2.14).

2.2.3 AprilTags map

The localization methodology detailed in this thesis closely follows the approach outlined in [5]. Accordingly, the same framework for generating an AprilTag map is employed. This AprilTag map encompasses a substantial arrangement of planar tags, each varying in size. For simplicity, the map's markers are positioned on the floor, although in certain scenarios, a ceiling-mounted configuration could be more advantageous. The selected map follows a dense pattern, wherein the tags are positioned in close proximity, meticulously adhering to a specific layout that eliminates gaps. This strategy is substantiated by the intent to initially assess the **VO!** system's performance within an optimal environment abundant in visual cues, thereby facilitating comprehensive spatial navigation. The adoption of a sparse map is deferred to potential future stages of research endeavors.

The map features a selection of AprilTags available in four distinct sizes, denoted as `S`, `M`, `L`, and `XL`. This choice stems from the UAV's operational requirement to navigate across various altitudes, where larger markers may become imperceptible at low altitudes, and smaller ones may prove challenging to detect from higher elevations. Each tag within the map is registered, assigning a unique identifier encoded within its pattern, thereby establishing a correlation between the tag's identity and its corresponding positional information. For practical utility, the IDs are systematically allocated such that they escalate in value as the marker's size decreases. This sequential arrangement is illustrated in Table 2.1, which provides a comprehensive depiction of the ID ranges alongside their associated dimensions. A compact tag pattern has been devised to incorporate varying marker sizes in a precise configuration, aimed at minimizing blank spaces, as visualized in Figure 2.5. The final map layout results from the roto-translation and mirroring operations of this optimized tags pattern. This pattern is replicated consistently throughout the map in two directions. The map generated with this procedure can encompass an extensive quantity of tags, thereby necessitating the utilization of the *TagStandard41h12* family, which supports up to 2115 distinct tags.

ID Intervals	Side length [cm]	Label
$0 \longrightarrow 99$	46	XL
$100 \longrightarrow 399$	23	L
$400 \longrightarrow 999$	11.5	M
$1000 \longrightarrow \infty$	5.75	S

Table 2.1: Tags numbers, sizes and IDs scheme.

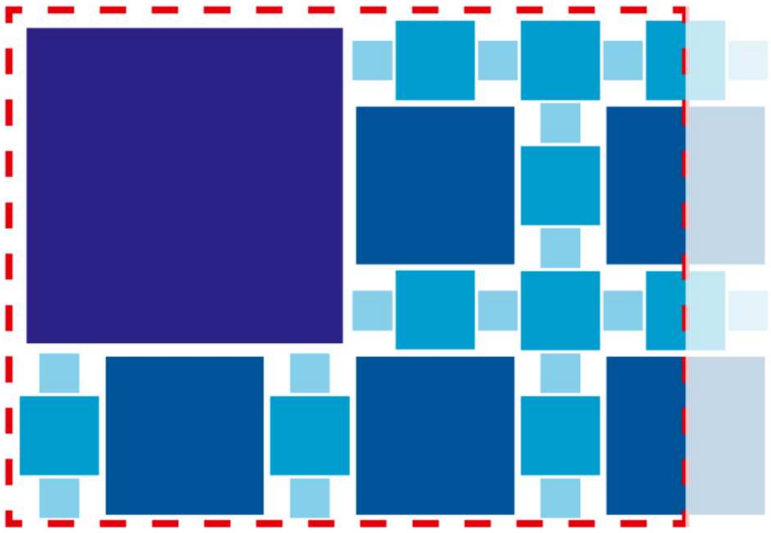


Figure 2.5: AprilTags map basic pattern design.

2.3 Hardware and software aspects

A standard multirotor system is characterized by four essential components, delineated as follows:

- An actuation system responsible for generating the requisite forces and torques that enable the vehicle to execute flight maneuvers as per commanded directives.
- A sensing and perception system comprising critical visual and inertial sensors, essential for capturing and quantifying the UAV's pose.
- A flight controller tasked with utilizing a control algorithm to furnish the actuators with appropriate input commands, facilitating adherence to the designated flight trajectory.
- A companion computer entrusted with labor-intensive data processing and, if necessary, furnishing trajectory setpoints.

Both the benchmark platforms used in this thesis, namely the custom quadrotor QR01 and the hexarotor HR01, adhere to a similar hardware architecture mirroring the aforementioned organization. Specifically:

- The flight controller is realized relying on the *PixHawk* platform, with version 4 for QR01 and version 6c for HR01.
- The companion computer is represented by a *Raspberry PI 4*, equipped with *Ubuntu OS* and *ROS2*.
- The sensor unit is constituted by an IMU (*ICM-20689*), integrated within the Pixhawk device.

A comprehensive representation of the adopted architecture is visually given in Figure 2.6.

2.3.1 PixHawk flight controller and PX4 Autopilot

The Pixhawk platform stands as a famous commercial hardware solution for UAVs, boasting of widespread utilization. Over the years, Pixhawk has been developed and released in various iterations. The more recent versions encompass an extended sensor board, housing the IMU in addition to the main flight controller board. Fueled by the *PX4 Autopilot* firmware, the flight control board takes charge of comprehensive vehicle behaviour management. In the case of the quadrotor platform (QR01), Pixhawk version 4 is integrated – a notable and popular release introduced in 2018. Conversely, the hexarotor platform (HR01) hosts a more recent version, Pixhawk 6c, representing a significant update not solely in hardware specifications, but also in terms of firmware enhancements.

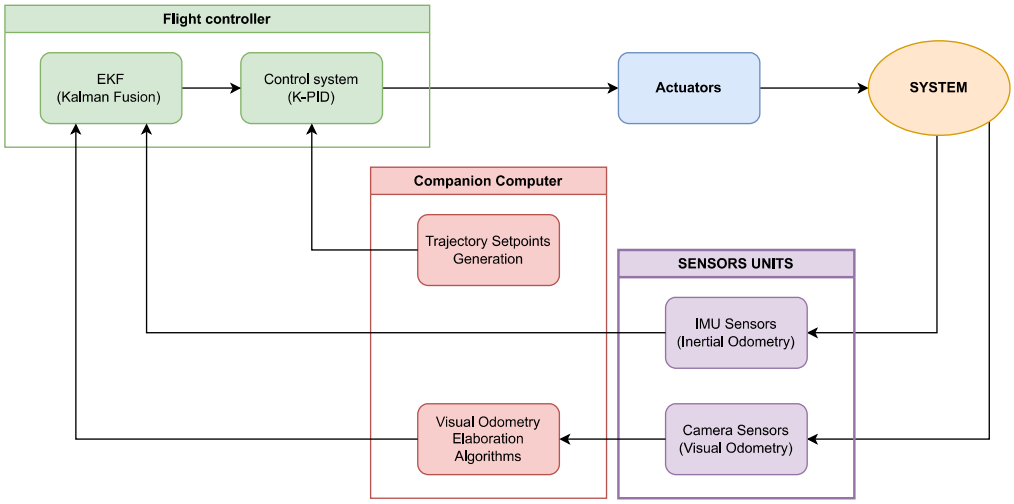


Figure 2.6: Basic scheme of the whole localization and control system.

Pixhawk is officially underpinned by PX4 Autopilot, a well-regarded open-source autopilot firmware embraced by both academic and industrial circles. The term "autopilot" denotes a system endowed with the capacity to autonomously control and navigate the trajectory of a vehicle or other robotic platforms, eliminating the need for human intervention. Accompanying the PX4 framework is the ground station software named *QGroundControl*. This software provides an intuitive graphical user interface (GUI) endowed with a range of functionalities, enabling trajectory planning and real-time tracking of UAV movement (albeit primarily tailored for outdoor applications). The architecture of the PX4 firmware is structured into four core layers:

1. The **middleware level**, housing essential drivers that facilitate interaction between hardware components (e.g., actuators, Pixhawk sensors, and IMU) and the internal operating system *NuttX*. Notably, *NuttX* is a Real-Time Operating System (RTOS) designed for embedded microcontrollers, boasting diverse applications.

2. The **communication level**, encompassing various communication protocols such as:

   - *MavLink*, a lightweight and secure protocol employed for the exchange of commands and information between a remote ground station powered by QGroundControl.
   - *Micro Object Request Broker (uORB)*, an intrinsic messaging protocol

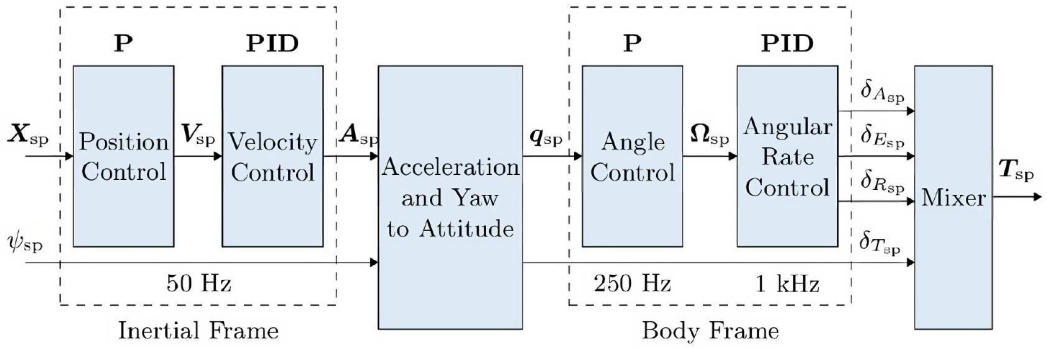


Figure 2.7: PX4 Flight controller block scheme, adopting a cascade strategy.

integrated within the PX4 firmware, facilitating streamlined and efficient communication between system modules.

3. A comprehensive **application layer** packed with a variety of customizable features.

The application layer comprises the nexus between the flight controller and the EKF. The flight control algorithm is characterized by a cascade architecture, the depiction of which can be found in Figure 2.7. This control algorithm encompasses, spanning from the inner loop, the following components:

- A simple Proportional (P) controller responsible for linear position, supplemented by an output limiter.
- A Proportional-Integral-Derivative (PID) controller governing linear velocity, comprising an Low-Pass Filter (LPF) within its derivative action.
- Another straightforward Proportional (P) controller for angular position (attitude).
- A K-PID controller, responsible for regulating angular speed. This constitutes a standard Proportional-Integral-Derivative controller with an additional tunable gain K, applied to all three facets of the control loop (proportional, derivative, integral). Additionally, a limiter is incorporated to prevent wind-up phenomena, and a LPF is invoked to modulate the impact of the derivative action.

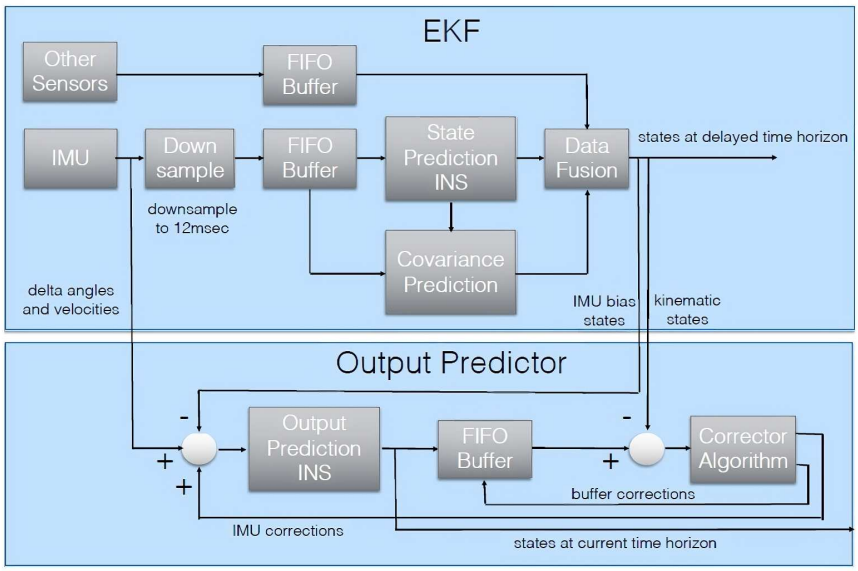


Figure 2.8: PX4 Extended Kalman Filter block scheme, comprising an additional output predictor.

Feedback for the system state is conveyed as a 3-dimensional vector for position and a 4-dimensional unit quaternion for rotation. This composite pose stands as the ultimate outcome of the Kalman fusion process executed by the EKF algorithm. The EKF is also part of the PX4 application layer and offers complete configurability. The block scheme for the EKF within the PX4 autopilot framework is depicted in Figure 2.8. Notably, the use of diverse sensors operating at varying sampling frequencies could potentially introduce delays and induce system instability. To address such problems, the architecture incorporates an auxiliary output predictor, thereby ensuring estimations free from delay issues.

2.3.2 Raspberry PI and Robot Operating System 2 (ROS2)

The primary component of the localization system is the companion computer, a Raspberry PI (version 4) hosting an Ubuntu operating system capable of complete support to the ROS2 library. Robot Operating System (ROS), a widely adopted open-source framework, offers a suite of libraries specifically created to support the development of intricate robotic applications. Despite its misleading nomenclature, ROS does not represent a conventional operating system, but rather a middleware. The fundamental objective of ROS is to facilitate the creation of robotic solutions by adopting a modular and highly scalable architec-

ture. This architecture relieves developers from the burden of managing low-level communications, enabling them to concentrate on high-level application development. Several characteristics underpinning its popularity, particularly within the academic research domain, encompass its status as a real-time peer-to-peer system, its support for multiple programming languages (including C++ and Python), and its open-source nature, making it freely accessible.

The core concepts of ROS2 can be concisely summarized in the following points:

Nodes ROS2 is grounded in a peer-to-peer communication framework where the individual peers are designated as "Nodes". Each node functions as an independent process programmed to execute a specific task within the robotic system. Nodes communicate and exchange information and directives through messages.

Messages Messages are "blueprints" defining the data types that nodes can share. These messages are categorized into different topics, thereby enabling each node to choose which types of messages to send and receive.

Topics Nodes can publish on designated topics and/or subscribe to other topics. This mechanism organizes messages more effectively, with topics serving as asynchronous communication channels. This use of topics preserves ROS2's intrinsic properties of modularity and scalability, permitting nodes to be added or removed from specific topics without necessitating a complete system reconfiguration.

Bags All the messages exchanged within a ROS2 session can be logged and stored within "bags". These bags function as log files encompassing all the data transmitted by nodes over the topics specified during recording. Bags can be "replayed", signifying that all the recorded messages are republished in a new session. This capability proves useful for recreating virtually identical scenarios to those captured during recording.

ROS2 represents an enhanced iteration of ROS aimed at overcoming some of its principal limitations and rendering it more suitable for complex robotic scenarios, including scenarios that encompass distributed systems structured within extensive and diverse networks. All the key algorithms and features developed for this thesis, are actualized through ROS2 nodes. These nodes interact

directly with each other, while also being capable of interfacing and exchanging information with the PX4 Autopilot's application layer. In specific, the quadrotor QR01 employs the ROS2 Foxy version, whereas the hexarotor HR01 employs the latest stable ROS2 version known as Humble[1].

2.3.3 PX4 – ROS2 COMMUNICATION BRIDGE

The use of PX4 Autopilot in conjunction with a ROS2 environment offers distinct advantages, as it allows for the expansion of PX4's conventional functionalities through the integration of new capabilities via ROS2 nodes. The pivotal factor is establishing a two-way communication channel between the two systems, each utilizing distinct internal messaging exchange protocols. To overcome this challenge, PX4 Autopilot features a dedicated *PX4-ROS2 bridge*. This bridge serves as a translation layer that facilitates bidirectional conversion between UORB and ROS2 messages.

The earlier version of PX4 (up to v1.13) employs a bridge with a *microRTPS architecture* (depicted in figure 2.9). This architecture encompasses a microRTPS client embedded in the PX4, located within the Pixhawk board, and a microRTPS agent running as a process within ROS2. This communication architecture is used in the QR01 quadcopter setup. It involves an older version of PX4 functioning on a PixHawk 4, communicating with *ROS2 Foxy*. In contrast, the more recent PX4 versions (beyond v1.13) adopt a new architecture (illustrated in figure 2.10), based on the *uXRCE-DDS middleware*. This newer architecture offers swifter and more reliable communication between PX4 and ROS2. Analogous to the previous architecture, it comprises a uXRCE-DDS client within PX4 and a uXRCE-DDS agent within ROS2. This contemporary bridge is employed in the HR01 hexacopter setup, which incorporates a modern PX4 version on a PixHawk 6c, communicating with ROS2 Humble.

For the bridge to function, two ROS2 packages are required and need to be constructed and installed within the companion computer:

- `Px4_msgs`, which entails the definition of PX4 messages in the ROS environment.

[1]for more details on ROS2 versions, see: https://docs.ros.org/en/rolling/Releases.html.

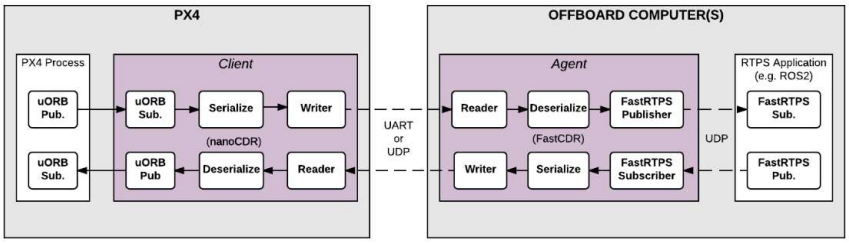


Figure 2.9: Old PX4 microRTPS bridge for communicating with ROS2.

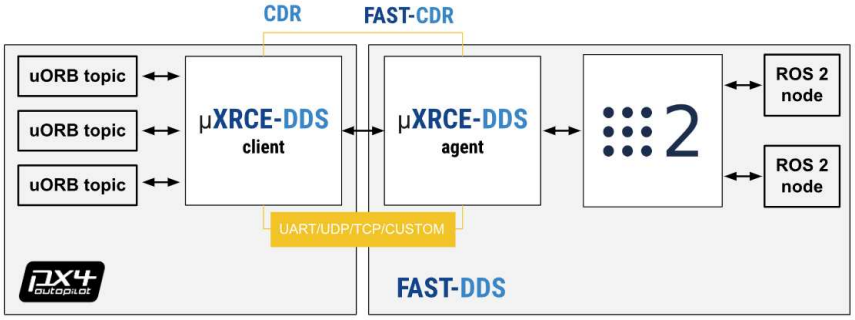


Figure 2.10: New PX4 uXRCE-DDS bridge for communicating with ROS2.

- `Px4_ros_com`, which houses the source code of the agent and requires compilation for proper functionality.

3

Tags processing algorithms for localization: analysis and implementation

3.1 Outliers Removal

The VIO system architecture elucidated in Section 2.2 demonstrates the capability to concurrently recognize multiple AprilTags. In the study conducted by *Bertoni et al.* [5], the ultimate pose computation concentrated solely on the most significant marker (the one with the smallest ID) detected within the scene. However, the decision to employ an extensive map of various-sized markers results in the simultaneous derivation of multiple pose estimations. Each of these estimations corresponds to the outcome of the identification process of an individual marker captured by the camera. In order to optimize the utilization of information furnished by the camera sensor, the final pose estimation should result from the fusion of data derived from diverse tags. The approach undertaken in this thesis, initially introduced in *Pescante et al.* [29], is deeply explored herein.

Nonetheless, direct averaging of poses originating from distinct tags can be ineffectual or potentially detrimental. This is predominantly due to the variable extent of uncertainty affecting these measurements. This aspect has already

been investigated in existing literature (see, for example, [1] and [24]), revealing that, in general, the error tends to grow as the distance between the tag and the camera increases. The tag-camera distance is intrinsically influenced by the flight altitude, exerting a greater impact on smaller-sized tags as opposed to larger ones. Distance, however, is not the sole factor influencing the precision of pose estimation from a single marker. Factors such as the camera's viewing angle, coupled with unforeseeable phenomena like environmental occlusions, reflections, and motion blurring, can rapidly degrade the accuracy of the estimation. In certain instances, certain AprilTags might even be confused with one another, thus resulting in entirely erroneous measurements.

Let's denote by $\mathcal{T}_T(t)$ the set of all the UAV's pose estimates obtained at a certain time instant t and by $\mathcal{I}_T(t)$ the set of all the marker IDs associated with the fiducial markers seen at that time t. All the pose measurements are initially referred to the camera frame $\mathcal{F}_C$, thus having for each i-th detected tag a pair $(\hat{\mathbf{p}}^C_{T_i}(t), \hat{\mathbf{R}}^C_{T_i}(t)) \in \mathcal{T}_T(t)$. Since the transformation from $\mathcal{F}_C$ to $\mathcal{F}_B$ is known and fixed, let's assume for simplicity to progress directly to the set

$$\mathcal{T}_B(t) = \{(\hat{\mathbf{p}}^W_{i,B}(t), \hat{\mathbf{R}}^W_{i,B}(t)) \in \mathbb{SE}(3), i \in \mathcal{I}_T(t)\} \tag{3.1}$$

referred to the body frame $\mathcal{F}_B$ [1].

In statistics, data points deviating significantly from the central cluster of a data set are typically referred to as outliers. To enhance the reliability of the subsequent pose estimation averaging process, the initial step entails the removal of outliers from the original data set. The final objective of this outlier removal procedure is to identify the subsets $\bar{\mathcal{I}}_T(t) \subset \mathcal{I}_T(t)$ and $\bar{\mathcal{T}}_T(t) \subset \mathcal{T}_T(t)$ containing solely those estimates deemed trustworthy, along with their corresponding IDs. In an ideal scenario, these subsets are attained by optimizing the detection of outliers while simultaneously minimizing the inadvertent exclusion of legitimate data points. Given that positions generally exhibit heightened susceptibility to estimation uncertainty during the marker recognition phase (refer to *López-Cern et al.* [24]), the entire process of outlier removal focuses exclusively

[1] Subsequently, each time from the notation for simplicity are omitted reference frame indications, it is assumed to be referring to $\mathcal{F}_B$, unless specifically stated otherwise.

on the positions set

$$\mathcal{P}_B(t) = \{\hat{\mathbf{p}}^W_{i,B}(t) \in \mathbb{R}^3, i \in \mathcal{I}_T(t)\} \tag{3.2}$$

This section proceeds to elaborate on three distinct well-established statistical approaches.

Interquartile Ranges Method

The Interquartile Ranges (IQR) method holds substantial popularity within statistical practices and offers the advantage of adaptability, especially for data samples that do not closely resemble a Gaussian distribution. This approach is traditionally employed with scalar variables; however, its application must be adjusted in the context of three-dimensional data points. One plausible adaptation involves the sequential consideration of each component of the vector $\hat{\mathbf{p}}^W_{i,B}(t) \in \mathbb{R}^3$ and therefore to work on the sets

$$\mathcal{P}_{k,B}(t) = \{\hat{p}_{i,k}(t) \in \mathbb{R} \mid \hat{\mathbf{p}}_i(t) = [\hat{p}_{i,1} \quad \hat{p}_{i,1} \quad \hat{p}_{i,1}]^T \in \mathcal{P}_B(t)\} \tag{3.3}$$

with $k \in \{1, 2, 3\}$.

Denoting by $|\cdot|$ the cardinality of a set, let's assume $|\hat{p}_{i,k}(t)| \geq 3, \forall k$ (otherwise the outliers removal process itself would be pointless). Assuming to sort the set $\hat{p}_{i,k}(t)$ in ascending order, the three quartiles (refer to figure 3.1) $\{q_{1,k}, q_{2,k}, q_{3,k}\}$ are evaluated such that:

$q_{1,k}$ signifies the value encountered just beyond the 25% of the ordered data.

$q_{2,k}$ represents the median value of $\mathcal{P}k, B(t)$.

$q_{3,k}$ corresponds to the value immediately above the 75% of the sorted data.

Subsequently, the *interquartile range (IQR)* is determined as $p^{IQR}_k = q_{3,k} - q_{1,k}$. The collection of estimates classified as reliable for the k-th component is

$$\bar{\mathcal{P}}_{k,B}(t) = \{\hat{p}_{i,k}(t) \in \mathcal{P}_{k,B}(t) \mid q_{1,k} - 1.5 \cdot p^{IQR}_k < \hat{p}_{i,k}(t) < q_{3,k} + 1.5 \cdot p^{IQR}_k\} \tag{3.4}$$

Further, the subset of IDs corresponding to estimates contained within $\bar{\mathcal{P}}_{k,B}(t)$ is designated as $\bar{\mathcal{I}}_{k,T}(t)$. Finally, the refined set of poses is determined by the

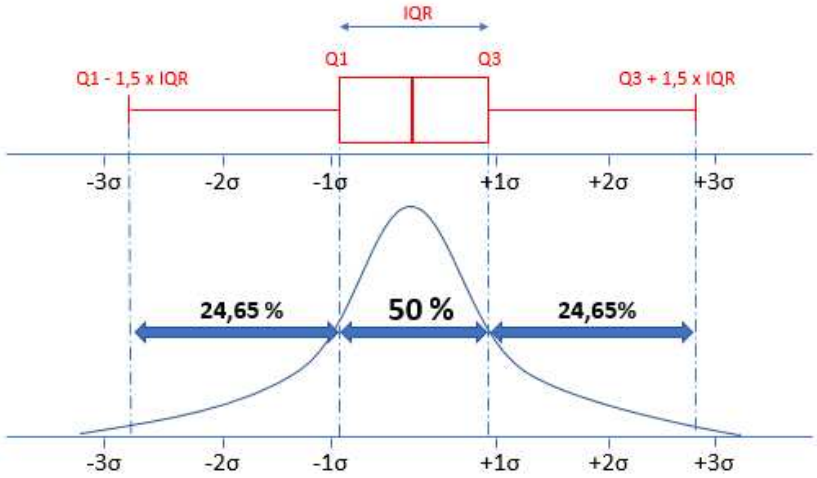


Figure 3.1: Quartiles and interquartile ranges for a Gaussian distribution.

intersection of the subsets yielded by each component, i.e. by

$$\bar{\mathcal{P}}_B(t) = \bar{\mathcal{P}}_{1,B}(t) \cap \bar{\mathcal{P}}_{2,B}(t) \cap \bar{\mathcal{P}}_{3,B}(t) \tag{3.5}$$

$$\bar{\mathcal{I}}_T(t) = \bar{\mathcal{I}}_{1,T}(t) \cap \bar{\mathcal{I}}_{2,T}(t) \cap \bar{\mathcal{I}}_{3,T}(t) \tag{3.6}$$

For its final implementation, this method will be extended into a weighted variant, introducing weights for each i-th pose estimate. This extension can be readily achieved by introducing the concept of *weighted quartiles*, denoted as $q^w_{1,k}, q^w_{2,k}, q^w_{3,k}$. These weighted quartiles are computed based on the sequence of cumulative weights, denoted as $\mathcal{W}_i = \sum_{k=1}^{i} \mathbf{w}_k$, where $\mathbf{w}_k \in \mathbb{R}^n$ represents the vector containing individual weights associated with the corresponding tag IDs in $\mathcal{I}_T(t)$. These weights will also play a role during the data fusion phase, being assigned to each pose estimate in accordance with specific heuristic guidelines. These guidelines incorporate a prior assessments of the reliability of each marker, considering factors such as its size or distance from the camera.

Distance from the Mean

The approach for outliers removal based on the distance from the mean is a widely adopted and straightforward method in statistics. With this approach, the mean of the data samples is computed, and then the distance of each individual element from the mean is assessed. If this distance exceeds a predetermined threshold, the element is identified as an outlier and subsequently removed from

the data set. Typically, the threshold is set at $\delta \cdot \sigma$, where σ represents the standard deviation of the measurements and $\delta \in [2, 3]$ is chosen based on the characteristics of the data distribution. In this context, considering that the data points correspond to positions $\hat{\mathbf{p}}_i(t)$ the calculation of the mean can be formulated as follows:

$$\boldsymbol{\mu}(t) = \frac{1}{n} \sum_{i=1}^{n} \hat{\mathbf{p}}_i(t) \tag{3.7}$$

Subsequently, the resulting set of cleaned positions is given by

$$\bar{\mathcal{P}}_B(t) = \{\hat{\mathbf{p}}_i(t) \in \mathcal{P}_B(t) \ \mid \| \hat{\mathbf{p}}_i - \mu(t) \| < \delta \cdot \sigma(t)\} \tag{3.8}$$

Distance from the Median

The technique based on the distance from the median employs the median rather than the mean, a more robust measure in statistical contexts. The initial step involves computing the median of the data set, denoted as $\boldsymbol{\mathcal{M}}(t) = \text{median}(\mathcal{P}_B(t))$. Subsequently, the focus shifts to determining the *Median Absolute Deviation (MAD)* of the data set:

$$MAD(t) = \text{median}\big(|\mathcal{P}_B(t) - \boldsymbol{\mathcal{M}}(t)| \big) \tag{3.9}$$

Sometimes, this MAD can be rescaled ($SMAD$), where it is multiplied by a constant $c \in \mathbb{R}$ that varies based on the data distribution. This rescaling is introduced to better adapt to the data set's characteristics, which may align more effectively with specific distribution types. For data sets influenced by white Gaussian noise, mimicking a Gaussian distribution, $c \approx 1.4826$, leading to $SMAD(t) = c \cdot MAD(t)$. Typically, the threshold for outlier detection is set as $\delta \cdot SMAD$, with $\delta \in [2, 3]$. Consequently, the refined set of pose estimations will encompass only those poses that lie within a radius characterized by $\delta \cdot SMAD$ from the median:

$$\bar{\mathcal{P}}_B(t) = \{\hat{\mathbf{p}}_i(t) \in \mathcal{P}_B(t) \ \mid \ \| \hat{\mathbf{p}}_i - \boldsymbol{\mathcal{M}}(t) \| < \delta \cdot SMAD(t)\} \tag{3.10}$$

Preliminary numerical assessment

The three aforementioned outliers removal methods were initially subjected to some simple numerical tests on MATLAB to assess their suitability for the application before considering their final implementation in ROS2. These tests were conducted using predefined sets of samples derived from previous experi-

mental flights performed in [5]. These data sets of pose estimates were extracted from several *ROS2 bags* recorded during past HR01 flights. The algorithms were conveniently implemented as *MATLAB* scripts and subjected to testing against the predefined examples.

During this preliminary and purely qualitative assessment, all three methods demonstrated the capability to achieve acceptable outlier detection, albeit with some distinctions. The techniques founded on interquartile ranges exhibited more consistent results when dealing with larger data sets, particularly when the data distribution deviated from the Gaussian ideal. Conversely, the methods relying on distance from the mean and distance from the median exhibited enhanced effectiveness in scenarios where the algorithms operated with very limited data sets ($n < 5$), necessitating a more assertive response. An illustration of algorithm application is depicted in Figure 3.2.

3.2 Data fusion by averaging transformations

After the removal of outliers from the initial data set, the subsequent task involves calculating the mean of transformations. In this section the problem of averaging transformations representing the pose is faced using various approaches inspired by the existing literature.

To formalize this, the goal of the data fusion phase is to compute a final pose estimate, denoted as $\left(\hat{\mathbf{p}}_B^W(t), \hat{\mathbf{R}}_B^W(t)\right)$ (or equivalently $\left(\hat{\mathbf{p}}_B^W(t), \hat{\mathbf{q}}_B^W(t)\right)$) by appropriately combining the information provided in $\bar{\mathcal{T}_B}(t)$. Achieving this objective requires addressing both positions and orientations separately. Let's define

$$\bar{\mathcal{P}}_B(t) = \{\hat{\mathbf{p}}_{i,B}^W(t) \in \mathbb{R}^3, i \in \bar{\mathcal{I}}_T(t)\} \tag{3.11}$$

$$\bar{\mathcal{R}}_B(t) = \{\hat{\mathbf{R}}_{i,B}^W(t) \in \mathbb{SO}(3), i \in \bar{\mathcal{I}}_T(t)\} \tag{3.12}$$

having both the same cardinality $\bar{n} \in \mathbb{R}$. While calculating the mean of positions may be considered a straightforward task, determining the mean of a set of rotations poses a complex problem to solve, regardless of the chosen representation (although certain representations may appear more convenient than others, particularly from a computational standpoint).

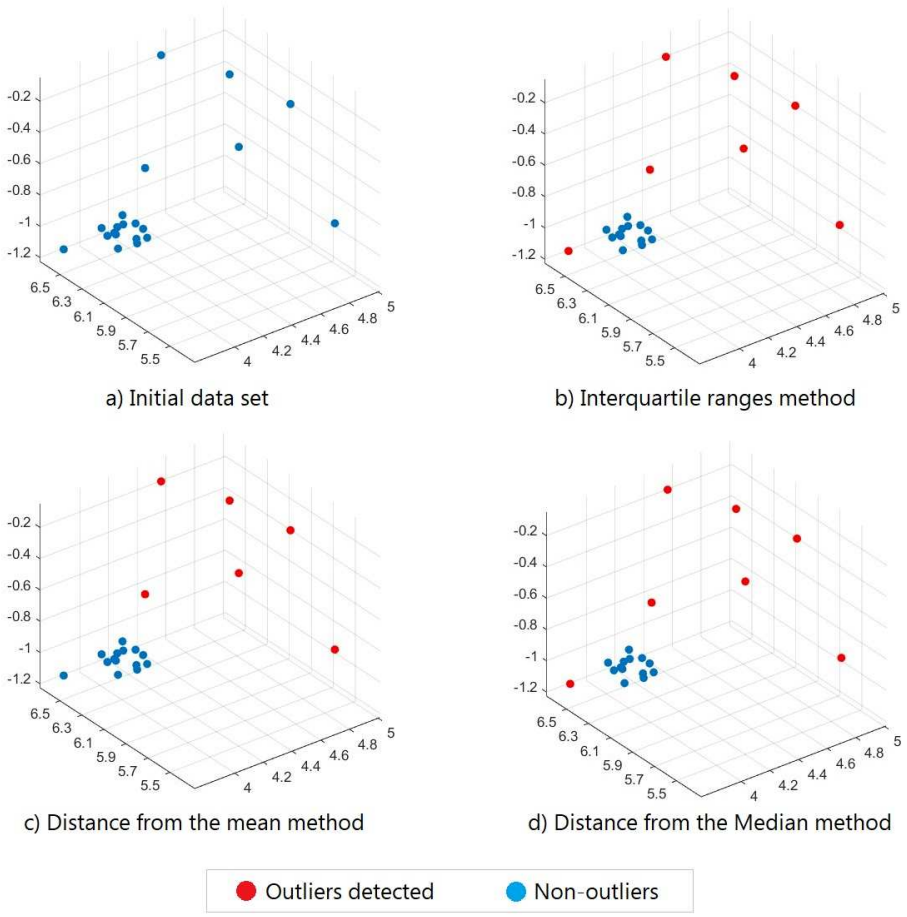


Figure 3.2: Visual example of an application of the three outliers removal methods.

3.2.1 Positions averaging

Averaging positions $\hat{\mathbf{p}}_{i,B}^{W}(t) \in \bar{\mathcal{P}}_B(t)$ is a straightforward operation achieved by employing the conventional Euclidean mean. The estimated UAV position is calculated as follows:

$$\hat{\mathbf{p}}_{B}^{W}(t) = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \hat{\mathbf{p}}_{i,B}^{W}(t) \tag{3.13}$$

In the final implementation, this computation naturally extends to the weighted case, considering associated weights for each position estimate.

3.2.2 Rotations averaging

Within the current body of literature, various methods for averaging rotations have been proposed, each employing a distinct rotation metric (refer to [14]) and aiming to minimize a unique cost function.

Method based on the Chordal L_2-Mean

Consider a set of rotation matrices $\mathcal{R} = \{\mathbf{R}_1, \ldots, \mathbf{R}_n\}$. The *Chordal L_2-Mean* also known as the Projected Arithmetic Mean, is derived from the following minimization problem:

$$\mathbf{R}^* = \underset{\mathbf{R_i} \in \mathcal{R}}{\operatorname{argmin}} \sum_{i=1}^{n} d_c(\mathbf{R}_i, \mathbf{R})^2 \in \mathbb{SO}(3) \tag{3.14}$$

Here, it's defined using the chordal metric for rotations, denoted as $d_c(\mathbf{R}_i, \mathbf{R}) = \| \mathbf{R_i} - \mathbf{R} \|_F$, where $\| \cdot \|_F$ represents the *Frobenius Norm*.

As proved in [11], if all the rotation matrices $\mathbf{R}_i$ are within a convex set $\mathcal{B}$ with a radius less than $\pi/4$, then the unique global Chordal L_2-Mean defined in 3.14 also falls within $\mathcal{B}$. Furthermore, its relative cost function is strictly convex within a certain ball $B \supset \mathcal{B}$. Under these convexity conditions, the existence of a global minimum is ensured. In this particular case, also a closed-form algorithm for solving the problem can be found. A theorem provided in [11] ensures that if all the rotations $\mathbf{R}_i$ lie inside a convex set $\mathcal{B}$ of radius less than $\pi/4$ then the unique global Chordal L_2-Mean defined in 3.14 lies also in $\mathcal{B}$ and its relative cost function is strictly convex on some ball $B \supset \mathcal{B}$. In this case, these convexity

conditions are enough not only to ensure the existence of a global minimum but also to find a closed-form algorithm to solve the problem.

The solution to this problem, outlined by *Markley et al.* in [25], involves utilizing quaternion representations for rotations. Let $\mathcal{Q} = (\mathbf{q}_1 \dots \mathbf{q}_n)$ be the collection of n quaternions one-to-one associated with the rotations in $\mathcal{R}$. In this context, when transitioning from $\mathbf{R}_i$ to $\mathbf{q}_i$, the choice of the sign between $+\mathbf{q}_i$ and $-\mathbf{q}_i$ does not influence the outcome.

The initial step is to construct the symmetric matrix $\mathbf{A} = \sum_{i=1}^{\bar{n}} \mathbf{q}_i \mathbf{q}_i^T$, $A \in \mathbb{R}^{4\times4}$. Through eigenvalue decomposition of $\mathbf{A}$, let Λ_A represent the spectrum of $\mathbf{A}$. The solution to the optimization problem in 3.14 is represented by the eigenvector $\mathbf{v}^*$ associated with the maximum eigenvalue λ^*, i.e.

$$\hat{\mathbf{q}}_B^W(t) = \mathbf{v}^*, \quad \mathbf{A}\mathbf{v}^* = \lambda^*\mathbf{v}^* \quad \text{s.t.} \quad \lambda^* = \max \Lambda_A. \tag{3.15}$$

Method based on the quaternion L_2-mean

This method, initially introduced and discussed in [11], is grounded in the utilization of *quaternion metrics* denoted as $d_q(\mathbf{q}_1, \mathbf{q}_2) = \min \left\{ \| \mathbf{q}_1 - \mathbf{q}_2 \|, \| \mathbf{q}_1 - \mathbf{q}_2 \| \right\} \in \mathbb{R}$. The quaternion L_2-mean is given by

$$\mathbf{q}^* = \underset{\mathbf{q}\in\mathbb{S}^3}{\operatorname{argmin}} \sum_{i=1}^{\bar{n}} d_q(\mathbf{q}_i, \mathbf{q}) \tag{3.16}$$

The minimization problem in 3.16 offers a closed-form solution under specified conditions concerning the set $\mathcal{R}$ of rotations to be averaged. Analogous to the Chordal L_2-Mean, if all rotations reside within a convex set $\mathcal{B}$ with a radius of less than $\frac{\pi}{2}$, then the unique global quaternion L_2-Mean also falls within $\mathcal{B}$. Moreover, the cost function in 3.16 exhibits strict convexity within a certain ball $B \supset \mathcal{B}$.

Let's denote by $d_R(\mathbf{R}_1, \mathbf{R}_2) = \| \log(\mathbf{R}_1, \mathbf{R}_2^T) \|$ the *Riemannian* (or *geodesic*) distance between two rotations $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{SO}(3)$ (an equivalent metrics is defined also for two quaternions $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{S}^3$). Now, given $\mathbf{S} \in \mathbb{SO}(3)$ such that $d_\angle(\mathbf{R}_i, \mathbf{S}) < \pi/4 \quad \forall\, \mathbf{R}_i \in \mathcal{R}$ and let $\mathbf{s} \in \mathbb{S}^3$ represent its quaternion form. For each rotation $\mathbf{R_i}$ one can assume to select the corresponding quaternion $\mathbf{q}_i$ with its sign such that $\| \mathbf{q}_i - \mathbf{s} \|_2 < \| -\mathbf{q}_i - \mathbf{s} \|_2$. Ultimately, the quaternion L_2-Mean can be computed

as

$$\hat{\mathbf{p}}_B^W(t) = \mathbf{q}^* = \frac{\tilde{\mathbf{q}}}{\| \tilde{\mathbf{q}} \|}, \quad \tilde{\mathbf{q}} = \sum_{i=1}^{\bar{n}} \mathbf{q}_i \tag{3.17}$$

This method is particularly advantageous due to its low computational complexity; the closed-form solution can be obtained simply by summing all the quaternions within the sets and subsequently normalizing them. However, special attention must be paid to the signs of the quaternion representations of the rotations. As previously mentioned in Section 2.1.1, this is a consequence of the Double Coverage Property. The signs must be chosen consistently to minimize $\| \mathbf{q}_i - \mathbf{s} \|_2 \ \ \forall \mathbf{q}_i \in \mathcal{Q}$, ensuring that quaternions lie in the same hemisphere of $\mathbb{S}^3$.

Method based on the geodesic L_2-Mean (Karcher Mean)

The Geodesic L_2-Mean, also referred to as the *Karcher mean*, is the rotation that resolves the minimization problem:

$$\mathbf{R}^* = \underset{\mathbf{R} \in \mathbb{SO}(3)}{\operatorname{argmin}} \sum_{i=1}^{\bar{n}} d_\angle(\mathbf{R}, \mathbf{R_i})^2, \quad \mathbf{R}_i \in \mathcal{R} \tag{3.18}$$

While it can benefit from particularly favorable convexity conditions, ensuring the existence of the global minimum for all $\mathbf{R}_i \in \mathcal{B}, \mathcal{B} \supset \mathbb{SO}(3)$, there is no closed-form solution for this problem. In [11], a simple convergent algorithm is proposed, which is a modified *Riemannian Gradient Descent* with a fixed unitary step-size. This method computes the average on $\mathbf{so}(3)$, i.e. the tangent space of $\mathbb{SO}(3)$ at the identity $\mathbf{R} = \mathbf{I}$, at each step. Then it projects the result back onto $\mathbb{SO}(3)$ using the exponential map[2]. The algorithm is the following.

This algorithm guarantees convergence under the condition that the ball $\mathcal{B}$ has a radius $\delta < \pi/2$, terminating at a geodesic distance $d_\angle < \epsilon \cdot \tan(\delta)/\delta$ from the mean.

Method based on the Geodesic L_1-Mean

This last strategy once again employs the geodesic distance to define the cost function but focuses on the L_1-Mean instead. The minimization problem is for-

[2]See [14] for more details on the tangent space of rotations $\mathbf{so}(3)$ and the exponential map, formally known as *Rodrigues's rotation formula*, which links the $\mathbb{SO}(3)$ manifold to its Lie Algebra.

Algorithm 1 Riemannian Gradient Descent for iterative geodesic L_2-Mean computation.

$\mathbf{R} \leftarrow \mathbf{R}_1$
Choose a tolerance $\epsilon > 0$
while ∞ **do**
  Compute $\mathbf{r} = \frac{1}{\bar{n}} \sum_{i=1}^{\bar{n}} \log(\mathbf{R^T R_i})$
  if $\| \mathbf{r} \| < \epsilon$ **then**
    return R
  end if
  $\mathbf{R} \leftarrow \mathbf{R}e^{\mathbf{r}}$
end while

mulated as follows:

$$\mathbf{R}^* = \underset{\mathbf{R} \in \mathbb{SO}(3)}{\operatorname{argmin}} \sum_{i=1}^{\bar{n}} d_{\angle}(\mathbf{R}, \mathbf{R_i}), \quad \mathbf{R}_i \in \mathcal{R} \tag{3.19}$$

This approach is intriguing because the L_1-Mean is traditionally considered more robust than the corresponding L_2-Mean, as discussed in [6] and similar sources. However, this method faces challenges related to its weak convexity and differentiability conditions, and once again, there is no closed-form solution. Some solutions using Riemannian Gradient Descent, similar to the one provided for the L_2-Mean, have been proposed in the literature. However, in this case, the approach would lead to a much more complex algorithm due to the need to perform a line search to compute the step length in the descending gradient direction. A practical strategy that simplifies the computation is the so-called *Weiszfeld Algorithm*, which offers a closed-form step length while still ensuring convergence in most circumstances. This traditional algorithm has been adapted to the geodesic L_1-Mean case by *Hartley et al.* [10]. The final algorithm is the following:

Preliminary numerical assessment

The algorithms outlined above were initially implemented in a MATLAB environment, and their suitability for the problem was verified, following a procedure similar to that employed for the OR algorithms. In this case, an example data set, denoted as Q_{gen}, was entirely artificial and randomly generated based on the model:

$$\mathbf{q}_i = \mathbf{q}_G + \mathbf{q}_{err}, \quad \mathbf{q}_i \in Q_{gen} \tag{3.20}$$

Algorithm 2 Weiszfeld algorithm for iterative geodesic L_2-Mean computation.

Initialize the geodesic median with the Karcher Mean $\mathbf{S} \leftarrow \mathbf{R}^*_{L_2}$
Choose a tolerance $\epsilon > 0$
while ∞ **do**
  Compute $\mathbf{v}_i = \log_S(\mathbf{R}_i)\ \forall i$ {logarithm map centered at S}
  $\delta \leftarrow \frac{\sum_{i=1}^{n} \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|}}{\sum_{i=1}^{n} \frac{1}{\|\mathbf{v}_i\|}}$ {Weiszfeld's step}
  if $\| \delta \| < \epsilon$ **then**
    return $\mathbf{S}$
  end if
  $\mathbf{S} \leftarrow e^{\delta}\mathbf{S}$
end while

Here, $\mathbf{q}_{err}$ is the corresponding quaternion representation of $\alpha_{err} \sim \mathcal{N}(0, \Sigma)$, a Gaussian random vector characterizing a random rotation in terms of Euler's angles. The covariance matrix Σ was thoughtfully selected to emulate realistic noise behavior.

Table 3.1 presents the mean and variance of the error obtained from running all algorithms for $k = 1, \ldots, 100$ iterations. The error is computed as $e_k = d_{\angle}(\mathbf{q}_k - \mathbf{q}_G)$ and is based on a set of $i = 1, \ldots, 20$ different rotations for each iteration, generated according to 3.20. The first three methods exhibit the best performance, while the approach relying on the geodesic L_1-Mean performed significantly worse. In some instances, it struggled to converge to a finite solution, revealing to be not suitable for a real-world application context. From a computational perspective, as anticipated, the first two methods, which offer closed-form solutions, demonstrated significantly faster execution times compared to the last two methods, which can only iteratively solve the problem. Consequently, it has been decided to exclude the latter two in favor of the former, particularly considering the limited computational capabilities of the Raspberry PI board on which ROS2 will be deployed. Hence, the two methods based on geodesic distance will not be integrated into the robotic system and will not be further considered in the subsequent chapters.

	Chordal L_2-Mean	Quaternion L_2-Mean	Geodesic L_2-Mean	Geodesic L_1-Mean
Angular distance error (radians)	0.018336 ± 8.4447e-05	0.018331 ± 8.4419e-05	0.018330 ± 8.441e-05	0.020642 ± 1.2e-04
Execution time (s)	0.000953 ± 7.8508e-06	0.002049 ± 6.5535e-06	0.18259 ± 0.00554	0.15319 ± 0.00341

Table 3.1: Results of the first rotation averaging algorithms validation in MATLAB.

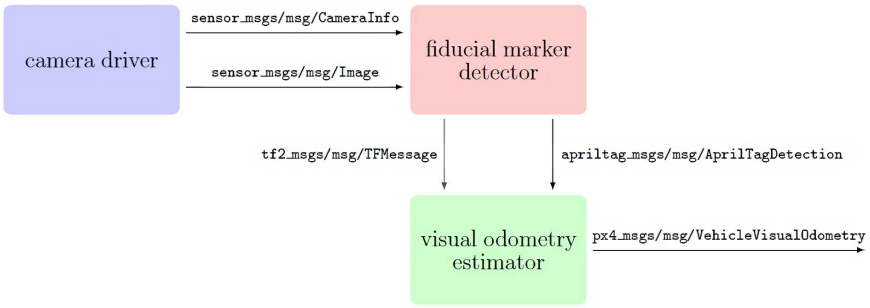


Figure 3.3: Principal ROS2 nodes constituting the VO system architecture and their relative topics.

3.3 VO ARCHITECTURE IN ROS2

In this section, the node architecture within ROS2 that implements the entire Visual Odometry (VO) localization procedure is presented. This architecture comprises three core nodes: a node responsible for serving as a camera driver, denoted as `camera_driver`; a fiducial marker detector node called `apriltag_ros`; and the visual odometry estimator, named after `apriltag_to_visual_odometry` and hereinafter abbreviated as *A2VO*. Notably, all of these nodes have been implemented in the C++ programming language to optimize performance. The primary focus of this thesis lies in the development and enhancement of the *A2VO* node. For a visual representation of the interactions among these nodes and the message exchange mechanisms, refer to Figure 3.3.

CAMERA DRIVER NODE

The camera driver is responsible for publishing sensor data from the captured images to specific topics. Specifically, the topics on which it publishes are:

- `sensor_msgs/msg/CameraInfo` – provides metadata related to the sensor, encompassing calibration parameters.
- `sensor_msgs/msg/Image` – contains the unprocessed image data captured by the sensor.

The choice of camera driver may vary depending on the type or model of the sensor connected to the board. Additionally, it's crucial to appropriately configure the camera driver, with key configuration parameters being the image resolution and the sampling frequency, often referred to as frames per second (fps). Opting for a higher resolution can enhance the recognition of AprilTags within the image, but it might demand greater computational power to process this information without compromising the sampling frequency. Given the limited computational capabilities of the companion computer, typically a Raspberry Pi in this context, it becomes imperative to strike a suitable balance between the required fps and image resolution.

Fiducial Marker Detector node

This node is responsible for implementing the algorithms related to the detection of April tags, following the procedure outlined in Section 2.2.2. It subscribes to the topics `sensor_msgs/msg/CameraInfo` and `sensor_msgs/msg/Image` to receive messages from the camera driver. It then processes the image data to detect and recognize April tags. For each detected tag, it computes pose estimates $(\hat{\mathbf{p}}^{C}_{T_i}(t), \hat{\mathbf{R}}^{C}_{T_i}(t))$ with respect to the camera frame $\mathcal{F}_C$. Subsequently, the node publishes on the following topics:

- `tf2_msgs/msg/TFMessage` - This message contains an array of type `geometry_msgs/msg/TransformStamped.msg`, where each element describes the affine transformation between the i-th identified marker $\mathcal{F}_{\mathcal{T}_i}$ and the camera frame $\mathcal{F}_C$.
- `apriltag_msgs/msg/apriltagDetection` – This message contains various 2D and 3D information pertaining to a single identified marker.

Actually, this convention has been subsequently modified by Pescante [29]. Instead of continuing to publish on the `/tf` topic, which is traditionally used by various sensors and robotics components and may lead to conflicts in future development, the node now publishes on a dedicated topic named `/tf_vio`. To run `apriltag_ros` and configure its behavior, a dedicated launch file is provided. This launch file allows users to set various parameters, including:

- "`family`": This parameter sets the family of AprilTags in use.
- "`tag_ids`": It contains an array specifying the subset of tag IDs chosen.
- "`tag_frames`": This parameter is an array containing the names of the frames to be associated one-by-one with the IDs listed in `tag_ids` (the arrays must have the same length).

- "`tag_sizes`": This parameter is used to specify the side dimension of the tag to be associated one-by-one with the IDs listed in `tag_ids` (the arrays must have the same length).

3.4 A2VO NODE

The node responsible for visual odometry estimation, named `apriltag_to_visual_odometry`, subscribes to the `tf2_msgs/msg/TFMessage` and `apriltag_msgs/msg/apriltagDetection` messages provided by `apriltag_ros`. These messages contain data regarding the relative transformations between recognized tags and the camera, represented by the set $\mathcal{T}_T(t)$. Initially, it calculates the camera's pose in relation to the world using Equation 2.11. Actually, in this specific laboratory setup, this operation involves just an offset compensation of $\mathbf{p}_{T_i}^W$, since the relative orientation of reference frames $\mathcal{F}_W$ and $\mathcal{F}_{T_i}$ is coinciding. In other words, $\mathbf{R}_{T_i}^W = I$, where I represents the identity rotation. After conducting all the necessary computations, which can encompass the transformation provided by a single tag (as done in [5]) or all the transformations, the final pose is determined using Equation 2.14. This estimate is subsequently published on a specific topic, such as `/fmu/vehicle_visual_odometry/in` (or `/fmu/in/vehicle_visual_odometry` as explained later). It is then transmitted to PX4 via the PX4-ROS2 bridge to be integrated into the Kalman fusion process. This node has gone through three different versions. The two older ones are briefly described in the following sections.

3.4.1 PAST VERSIONS

The initial version, referred to as *A2VO-v1*, was authored by SEGATTINI [32]. In this version, the pose estimation process relies solely on information provided by a single tag. Specifically, it selects the tag with the smallest ID, corresponding to the first AprilTag of the largest size encountered. The rationale for this choice is detailed in Section 2.2.3. *A2VO-v1* subscribes to the following topics:

- "`/tf`" - It retrieves pose information from `apriltag_ros` using a Transform Listener object.
- "`/fmu/timesync/out`" - This topic supplies the elapsed time since PX4's boot, facilitating synchronization between ROS2 and the PX4 Autopilot.

- "`fmu/vehicle_odometry/out`" - It receives the most recent vehicle pose estimate as output from the Extended Kalman Filter (EKF).

The node publishes data on the following topics:

- "`/tf`" -This is used to transmit pose estimate information to a simulation environment, such as *Rviz2*.
- "`/apriltag_for_estimation`" - Messages on this topic contain a string listing all the marker IDs observed since the start of the flight.
- "*Fmu/vehicle_visual_odometry/in*" - This is the most important output of the node, representing the Visual Odometry (VO) pose estimation sent to the EKF implemented on PX4.

The central function within the node is named `on_timer()`, and it is executed periodically at a frequency of 50Hz. In this function, the primary task is to locate the transformation related to the largest-sized tag. This search is performed within a transformations buffer, containing only the most recent transformations received. The *Transform Listener* class is employed for this purpose. However, it's essential to inspect all the buffers from the head to the tail since the transformations are not sorted in any specific order. Once the marker is located, its relative transformation is used to compute the final pose estimate, which is subsequently published, following the equations outlined in 2.14. A flowchart detailing this function is provided in Figure 3.4a. *A2VO-v1* underwent testing and validation, as described in [5]. However, the localization precision left room for improvement, prompting the need for enhancements.

Pescante [29] subsequently developed a new version of the node, referred to here as *A2VO-v2*, with the primary objective of enhancing localization performance. Several improvements were introduced in comparison to the initial version.

Latency reduction: To reduce latency, the arrival of a `/tf` message transmitted by `apriltag_ros` has been synchronized with the subsequent processing and publication of a visual odometry message. This adjustment involved discontinuing the use of the transformations buffer and the Transform Listener class entirely. Instead, a callback function called `tf_callback()` is invoked and

executed each time a `/tf` message is received. In this function, the transformation associated with the tag having the smallest ID (representing the largest tag seen) is straightforward to locate, as all transformations arrive in an array of type `geometry_msgs/msg/TransformStamped.msg` sorted in ascending order based on ID. The algorithm from *A2VO-v1*, hereafter referred to as **JB** (Just Bigger), remains viable but boasts a simpler and faster implementation that does not necessitate any loops. As a consequence of this change, the `on_timer()` function becomes unnecessary and has consequently been removed. To prevent any complications stemming from the general use of a `/tf` topic, *A2VO-v2* adopts the `/tf_vio` topic, as elaborated upon in Section 3.3.

Better data exploitation: A pivotal shift in approach involves harnessing the entirety of the information provided by all detected tags, rather than confining the analysis to just the largest marker. This strategic modification constitutes the initial implementation of the data fusion process, outlined in Section 3.2. Within *A2VO-v2*, a function named `translation_average()` has been integrated to compute the position average, treating it as the barycenter of data points, as depicted in Figure 3.4c. Complementary to this, the Chordal L_2-Mean is employed for rotations averaging. The key departure from Equation 3.15 lies in the introduction of weights linked to each individual tag, based on its marker size. Larger markers receive higher weighting. This same weight-based strategy is extended to the averaging of rotations, conducted using quaternion representation. Figure 3.4d illustrates the flowchart of the `quaternion_average()` function, which implements the same methodology expounded in Section 3.2.2, with the addition of the weight factors.

Outlier Identification and Removal: The introduction of a data fusion technique in the previous version of the node highlighted the need for data cleansing to mitigate the influence of outliers. In *A2VO-v2*, three distinct outlier removal functions have been integrated for this purpose.

- `JustBiggerTwo()` - This method concentrates solely on the two tags with the smallest IDs, combining their information. It has been deprecated in the last version of the node.
- `CartesianDistance()` - Employing an iterative approach, this method selectively removes data points situated far from the pose estimate generated downstream of the EKF algorithm and relative to the previous time instant. This estimate is available by A2VO from the

`fmu/vehicle_odometry/out` message. This method has been deprecated in the last version of the node.

- `Interquartile_filter()` - The third method introduces the initial version of the interquartile range approach. It further incorporates the adoption of an identical set of weights for each marker in the procedure, enabling the calculation of weighted quartiles. The flowchart outlining this function is illustrated in Figure 3.4b.

Although the first two techniques exhibited limited effectiveness, the interquartile ranges method demonstrated robustness and a high detection rate, prompting its inclusion in this thesis work.

Noise reduction at high frequency: The visual odometry estimate produced by *A2VO-v1* was susceptible to pronounced noise and high-frequency fluctuations. To address this issue, a Finite Impulse Response (FIR) filter was introduced to act as a low-pass filter. Such a filter can be described as a moving average filter with an impulsive response:

$$y(k) = \frac{1}{N} \sum_{i=0}^{N-1} u(k-i) \tag{3.21}$$

The impulsive response of this filter ends up after N sampling periods, corresponding to the filter's order. Equation 3.21 can be extended to accommodate weights w_i associated with $u(k-1), \ldots, u(k-N)$, yielding the weighted moving average filter:

$$y(k) = \frac{\sum_{i=0}^{N-1} w_i \cdot u(k-1)}{\sum_{i=0}^{N-1} w_i} \tag{3.22}$$

In [29], the weights remained unadjusted, treating all state entries equally (i.e. all the weights have been set to 1). A more in-depth exploration of the FIR weights to fine-tune this parametrization is conducted in the experimental portion of this work (refer to Section 4.3.1). The function responsible for implementing the FIR filter is straightforward, utilizing the same averaging functions employed for data fusion to compute the mean in both position and orientation of the last four VO estimates. These estimates are appropriately stored in a C++ Vector Object, functioning as a queue for efficient handling.

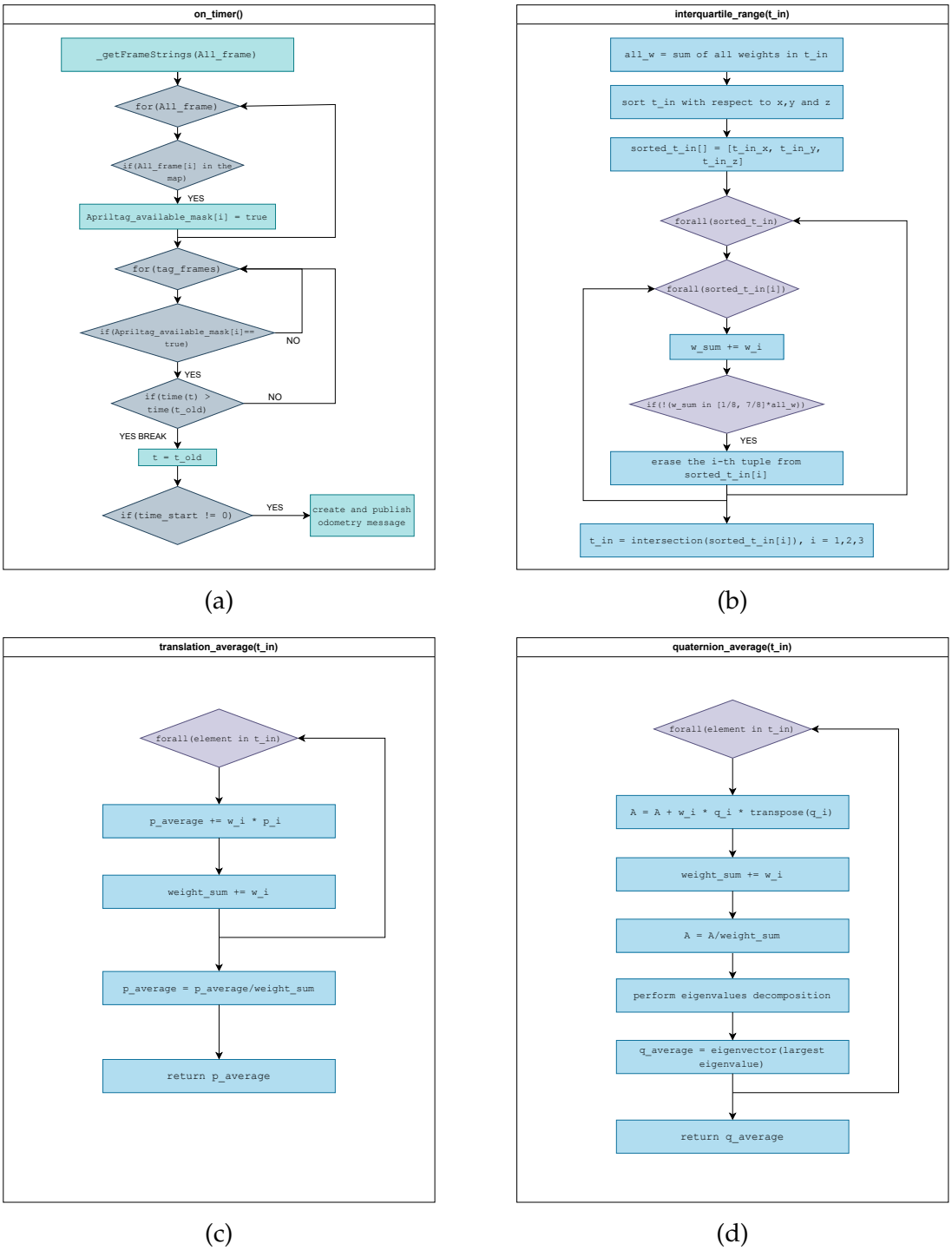


Figure 3.4: Logical scheme of the A2VO-v1 and A2VO-v2 main functions.

3.4.2 A2VO New Version

The new version of the node, named *A2VO-v3*, maintains the core structure of the code from *A2VO-v2*, but it introduces significant expansions and reorganizations within the `tf_callback()` function. Several functions embodying the novel techniques outlined in Sections 3.1 and 3.2 have been integrated.

Regarding the functions inherited from *A2VO-v2*, some remain unchanged, while others have undergone complete replacements or modifications aimed at enhancing performance and code clarity. However, an exception to this is the function implementing the FIR filter, which has remained largely unaltered from *A2V0-v2* due to its limited room for improvement. Notably, this version leverages the *Eigen3* C++ library to a greater extent. *Eigen3* is a widely adopted library for algebra and matrix computations, offering an array of templates and functions tailored for solving various algebraic problems such as linear systems, eigenvalue decomposition, and matrix operations. The primary advantage of this library lies in its remarkable performance and efficiency when compared to alternative implementations relying on standard C++ classes or less sophisticated mathematical libraries.

Some functions that were originally crafted in *A2VO-v2*, utilizing loops and other conventional programming control structures alongside frequent use of the C++ Vector class and its functions, have undergone partial or complete rewriting to enhance performance. Given that certain functions within the standard C++ Vector library entail high computational complexity and are less suited for this task, reliance on vector class functions has been substantially curtailed. Instead, the majority of loops have been substituted with algebraic operations carried out through Eigen3 Vectors and Matrix objects. This strategy offers heightened computational efficiency, significantly reducing the computational cost of algorithms, while aligning the implementation more closely with its mathematical formulation detailed in Sections 3.1 and 3.2. In practice, Matrix and vector operations executed using specialized libraries like Eigen3 can markedly outperform standard loops and control structures due to several critical factors, including:

- Optimized exploitation of memory cache locality.

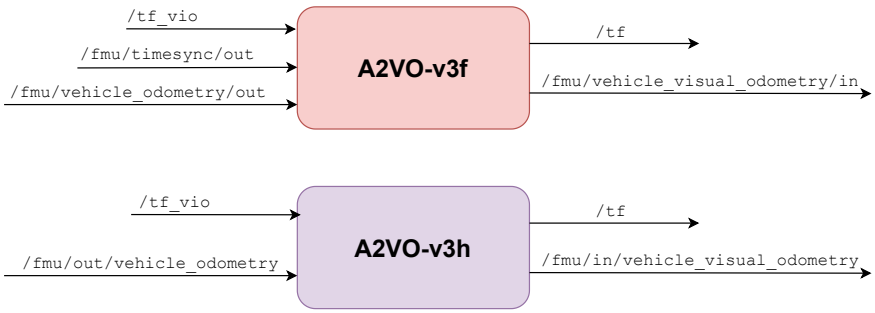


Figure 3.5: Graphical representation of A2VO-v3 and its relative topics used.

- Vectorization of calculations through *SIMD* (*Single Instruction, Multiple Data*) operations.
- Parallelism across distinct CPU cores, and in some cases, harnessing GPU acceleration where feasible.
- Minimized memory allocation for variables and temporary data.

Regarding the topics used and messages exchanged with the other node, the configuration has remained largely unchanged compared to *A2VO-v2*. However, as *A2VO-v3* needs to be compatible with both the older hardware/software architecture used by QR01 and the new architecture used by HR01, two slightly different versions have been developed. *A2VO-v3f* denotes the version running on QR01 on ROS2 Foxy, and its topics are the same as *A2VO-v2*. On the other hand, *A2VO-v3h* designates the version of the node updated for the new architecture. The only distinction between the two lies in the choice of the exchanged topics, which have different names and employ different types of messages (see Figure 3.5). For this reason, the following discussion will consider a general *A2VO-v3*, irrespective of the specific architecture or platform in use.

The overall code flow of the `tf_callback()` function has undergone reorganization. Figure 3.6 provides a logical overview summarizing the principal steps comprising the routine. The associated `.yaml` configuration file for the *A2VO* node launch has also been restructured to allow for the selection of various newly implemented methods. Subsequent pages dive into further details concerning the implementation of data cleaning and data fusion techniques.

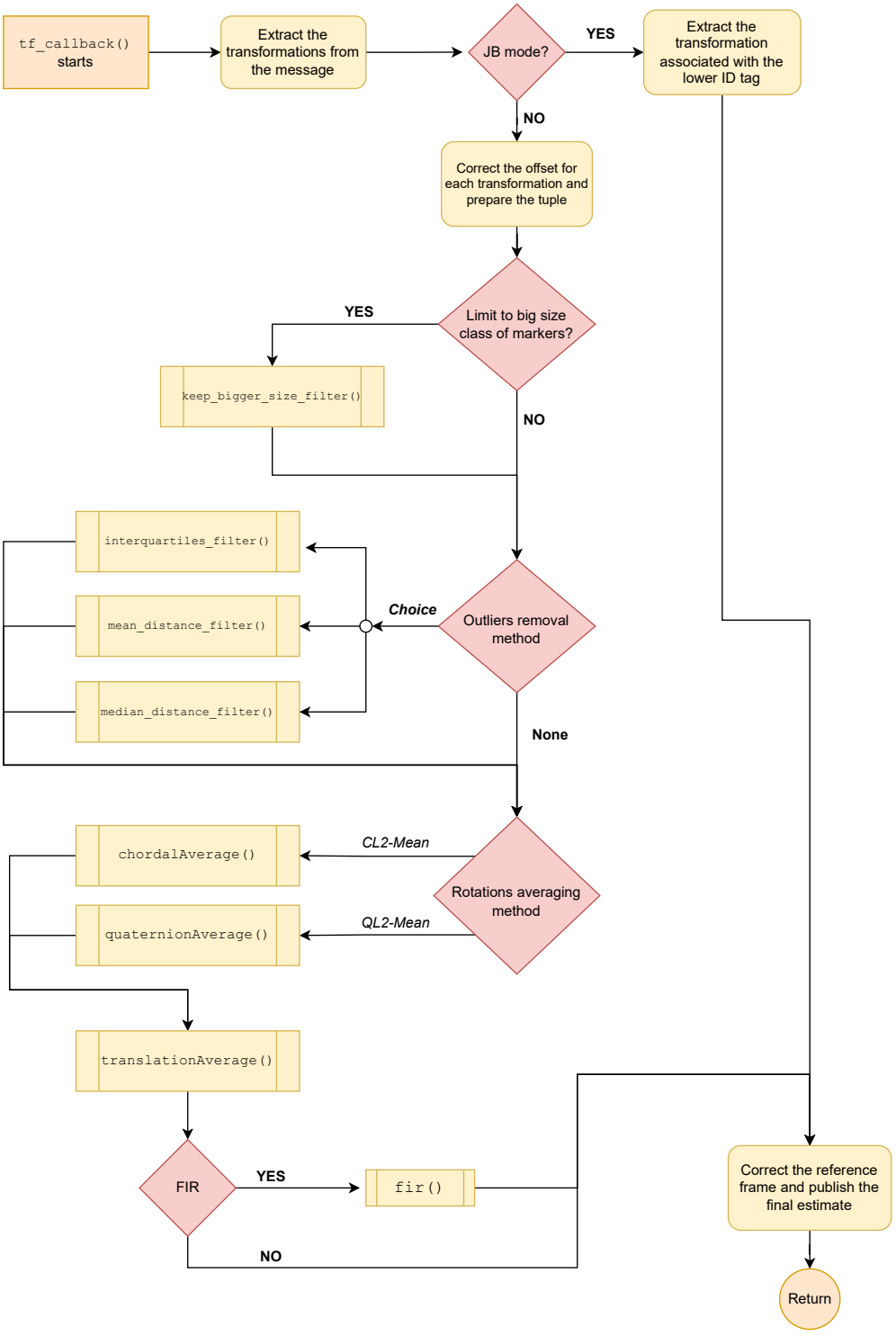


Figure 3.6: Complete logical scheme of the `tf_callback()` routine of *A2VO-v3*.

Outliers removal algorithms implementation

The process of removing outliers has been expanded and refined by implementing the three strategies discussed in Section 3.1:

- The method based on the distance from the mean **DMN-OR** is now incorporated into the function `mean_distance_filter()`, which directly accepts a pointer to the vector of transformations and removes the detected outliers within it. Because the Eigen3 library lacks statistics functions, an auxiliary function called `standard_deviation()` has been introduced to calculate the standard deviation of the vector of transformations. The algorithm's flowchart is presented in Figure 3.7a.
- Similarly, the method based on the distance from the median **DMD-OR** has been implemented, leveraging the functions provided by the Eigen3 library. However, since Eigen3 does not offer a built-in function to compute the median, a dedicated function called `median()` has been developed and incorporated. The algorithm's flowchart is outlined in Figure 3.7b.
- The method involving interquartile ranges **IQR-OR** has also been retained. However, its implementation has been thoroughly revised and the function completely rewritten compared to *A2VO-v2*, aiming to enhance its speed. Sluggish operations on Vector objects and certain loops have been replaced with more efficient operations featuring lower asymptotic complexity. This adjustment ensures superior performance, particularly when handling a substantial set of transformations as input (e.g., when the VO system detects numerous AprilTags). The flowchart for this updated implementation is provided in Figure 3.7c.

In addition to the aforementioned functions, an additional option has been integrated into the data cleaning phase in *A2VO-v3*. This version permits the system to either consider only the larger-sized tags (**OBS-OR**) or solely the two largest sets of tags (**TBS-OR**) visible at each execution of `tf_callback()`. This choice is prompted by two observations made during the initial experimental validation:

1. When larger-sized tags are in view, the UAV platform is typically flying at high altitudes. In such situations, smaller tags appear very small within the camera image, rendering the associated pose estimates more susceptible to errors and uncertainties.
2. When larger-sized tags are visible at high altitudes, the multitude of smaller tags becomes less reliable and less significant. Opting to focus solely on the larger markers helps reduce the computational burden of subsequent operations, enhancing speed and diminishing latency.

Both options are implemented within a dedicated function called `keep_bigger_size_filter()`. Users can select between **OBS-OR** and **TBS-OR** via a configuration parameter. The flowchart detailing this process is presented in Figure 3.7d. This technique employs stacks to temporarily organize the transformations based on the size of the associated marker. Subsequently, only the transformation belonging to the first (or the two first) non-empty buckets is retained, while the rest are discarded. This function is called at the outset of the code, preceding the execution of other outlier removal functions, and can be combined with them to create hybrid strategies, as demonstrated in Chapter 4.

3.4.3 Data Fusion algorithms implementation

The implementation of data fusion methods aligns with their theoretical descriptions in Section 3.2. Here are the details:

Position Averaging (POS-AVG) A function to calculate the Center of Mass of the positions of transformations is provided. While it follows the same basic formulas, the implementation of the `translationAverage()` function has been revised from *A2VO-v2* to leverage the Eigen3 library. The flowchart is depicted in Figure 3.8.

Chordal L_2-Mean (CL2-AVG) This method has been implemented within the `chordalAverage()` function (which was previously named *quaternionAverage* in *A2VO-v2*). The new version has been entirely rewritten to rely on the Eigen3 Library, employing vector and matrix operations that replace all the loops. An auxiliary function named `process_transformations()` has been introduced to facilitate the change in data structure, transitioning from Vector of Tuples objects representing transformations used in the data cleaning phase to Eigen3 Matrix and vector objects used in the data fusion phase. The flowchart of this function is presented in Figure 3.9a.

Quaternion L_2-Mean (QL2-AVG) This method has been implemented in the `quaternionAverage()` function, following the equations provided in Section 3.2.2. Special attention has been paid to the issue of quaternion signs arising from the double coverage property. To resolve this problem, for each rotation $\mathbf{q}_i \in \mathcal{Q}$, the sign of all the quaternions is determined as the minimizer between $\| \mathbf{q}_i - \mathbf{q}_{EKF} \|$ and $\| \mathbf{q}_i + \mathbf{q}_{EKF} \|$, where $\mathbf{q}_{EKF}$ represents the previous EKF estimate (obtained via the `fmu/vehicle_odometry/out`

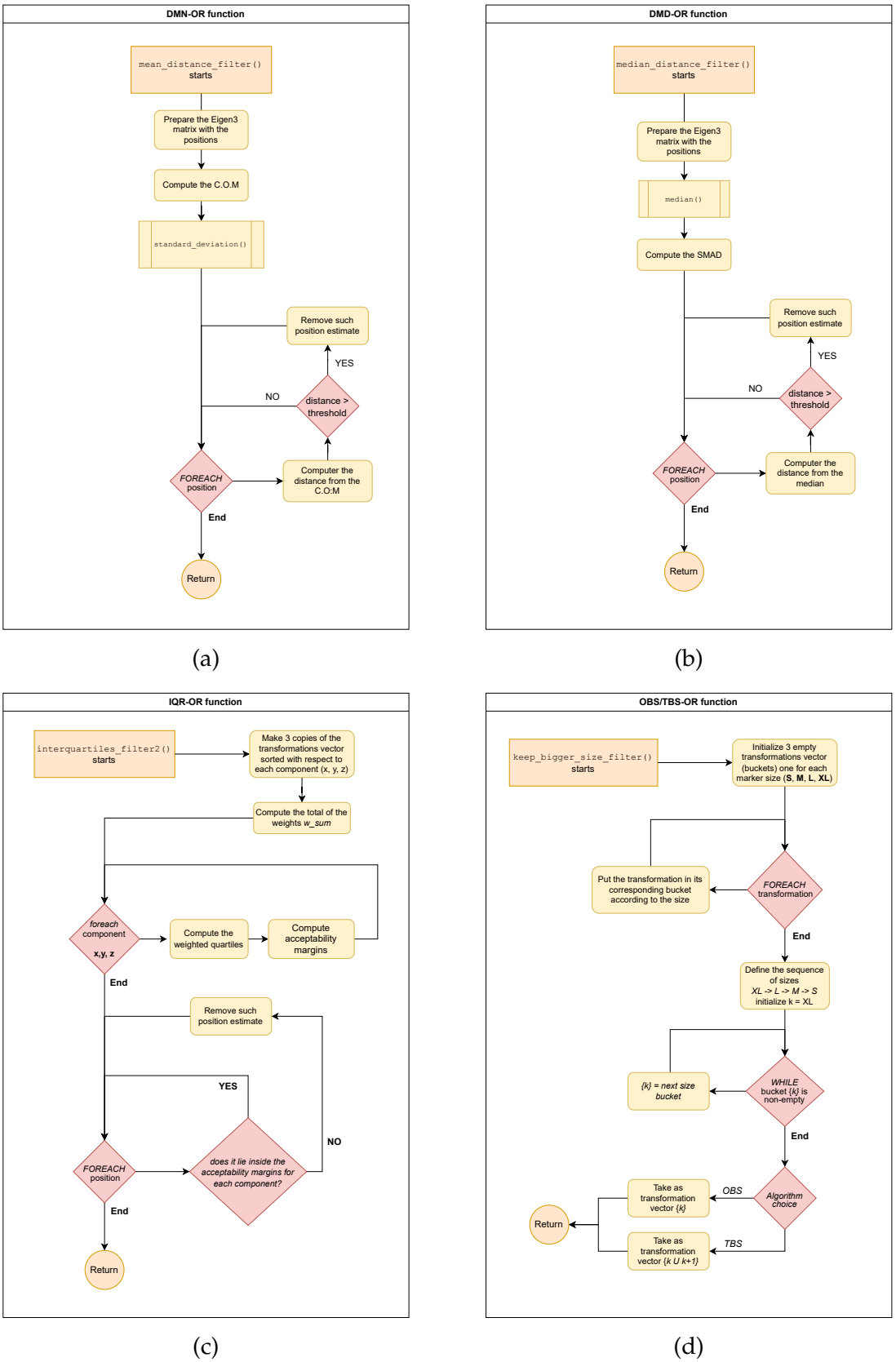


Figure 3.7: Logical scheme of the novel outliers removal algorithm implemented in *A2VO-v3*.

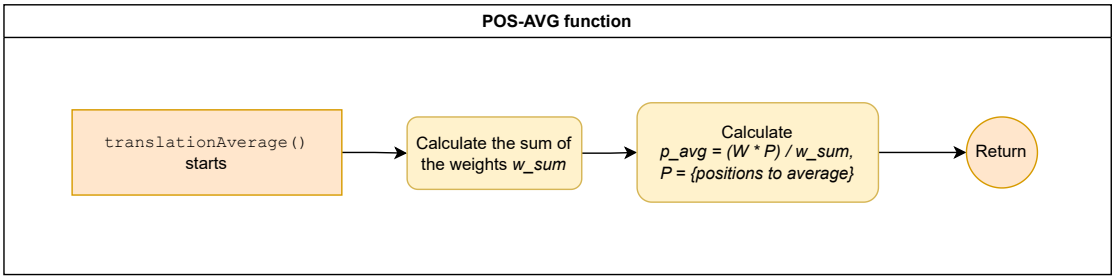


Figure 3.8: Graphical representation of the algorithm that performs positions averaging in *A2VO-v3*.

message). This computation is delegated to the `process_transformations()` function for computational convenience. The flowchart of this function is depicted in Figure 3.9b.

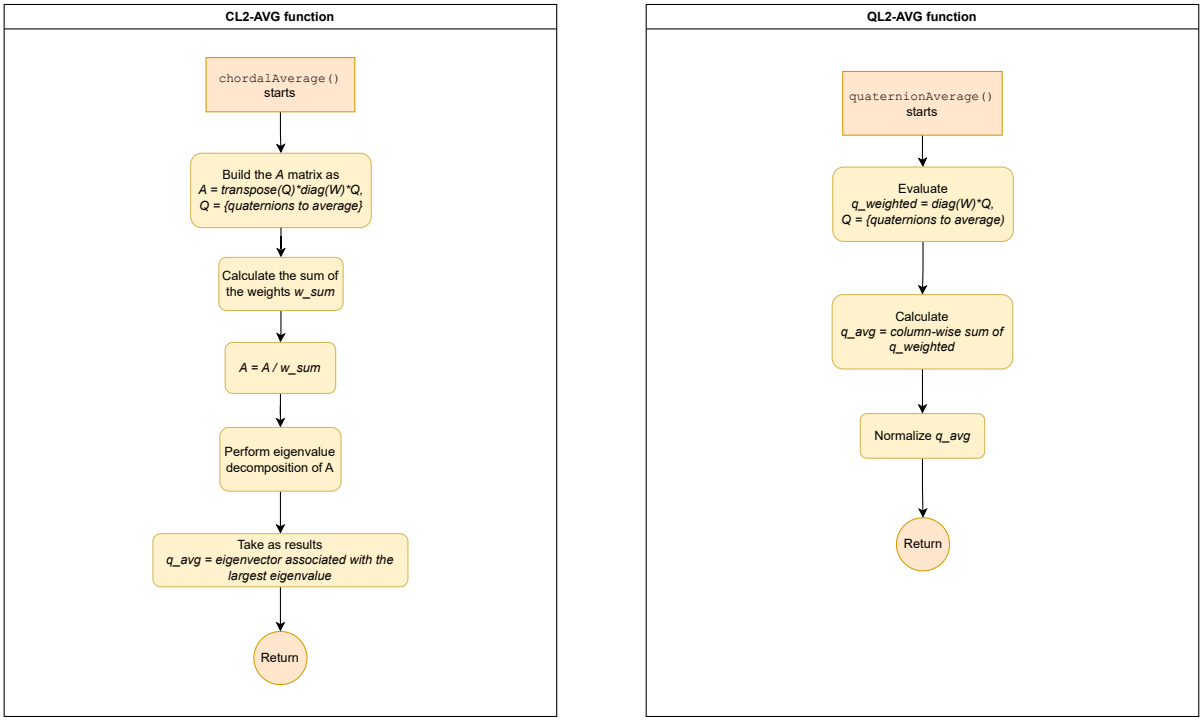
Regarding the weights used in all averaging operations and applied by **IQR-OR**, Chapter 4 addresses two strategies, each employing distinct sets of basic weights:

Static Weights These weights are statically assigned to the transformations based on the associated tag size. For each tag size $k = 1 \dots 4$, a different weight W_k is considered, with larger sizes related to greater weight. These sets of weights can be chosen to be more extreme, such as using a quadratic relation, or more balanced, like employing a linear relation. In particular, two sets of weights will be considered are the following: **W1** with $w_i = 4^h$ and **W2** with $w_i = 2^h$ where $h = 0, 1, 2, 3$ depending on the i-th marker size i.e., in this order, *S*, *M*, *L*, *XL*.

Dynamic Weights This approach aims to account for the distance between the camera and the tag. To achieve this, the static weights are made "dynamic" by applying the formula: $w_i = \frac{W_k}{\|\{\mathbf{p}^C_{T_i}\}_i\|}$.

For further insights into average execution times and the actual computational complexity of these algorithms, refer to Chapter 4, which discusses the results of experimental tests.

Regarding the configuration file (`.yaml`), significant modifications have been made to accommodate the enhancements and new functionalities. Several parameters have been either removed or revised, and new ones have been intro-



(a) (b)

Figure 3.9: Logical scheme of the novel rotations algorithms algorithm implemented in *A2VO-v3*.

duced. These newly added parameters serve to facilitate the activation or deactivation of specific functionalities, as well as the selection of algorithms to execute. The new parameters include:

- `chordal_averaging`: When set to `true` the system employs the **CL2-AVG** method; otherwise, it uses the **QL2-AVG** algorithm, which it is also the default preference.
- `dynamic_weighting`: When enabled (`true`), the system introduces dynamic weights, which are calculated at each time instant using the `get_dynamic_weight` function, considering the Euclidean distance between the camera and markers.
- `limit_to_big_sizes`: This parameter determines whether to invoke the `keep_bigger_size_filter()` function, allowing for consideration of only the first largest-sized marker set (1) or the first two (2) among all detected markers at each time instant before proceeding with the outliers removal process. When set to (0), all estimates are retained and processed.
- `outliers_filter_choice`: This parameter governs the selection of the outliers removal algorithm to execute, including older deprecated functions from *A2VO-v2*. When set to (0), the outliers removal procedure is bypassed.

4

System testing and validation

4.1 Experimental setup

The experiments were conducted within the *SPARCS laboratory*, located at the Department of Information Engineering, University of Padova. The laboratory setup used for these experiments consisted of three crucial elements:

- The multirotor UAVs.
- The Vicon Motion Capture system.
- The Apriltag Map.

Two co-planar multi-rotor platforms were utilized for these experiments: a custom hexarotor called **HR01** and a smaller custom quadrotor denoted as **QR01**, both of them developed at the *Dipartimento di Tecnica e Gestione dei Sistemi Industriali (DTG)* of the *University of Padova*. Although these platforms shared a common hardware and software framework, they differed in terms of actuation, chosen electronic components, dimensions, and weight. These variations in electro-mechanical components, electronics, and devices allow for a comprehensive validation of the developed localization system by accounting for a wider range of variability in the host system components. Table 4.1 provides a concise overview of the primary hardware and software components for both vehicles, emphasizing the distinctions between them. Both UAVs were equipped

(a) (b)

Figure 4.1: The two star-shaped multi-rotor platforms employed for the experiments, the hexarotor HR01 and the quadrotor QR01.

Platform	HR01	QR01
Propellers	Tarot 1355 Carbon Propellers 13″	5″ Plastic
Chassis	Tarot 680 pro (base 695mm)	Carbon Fiber airframe (base 250mm)
Motors	Tarot 6S 380KV 4008 (brushless PM motors)	DR2205 KV2300 (brushless PM motors)
Electronic speed controller (ESC)	Holybro Tekko32 ESC (35A)	Fully assembled power management board with ESC (custom)
Battery	Turnigy 6600mAh 6S	Turnigy 2200mAh 4S
Flight Controller	PixHawk 6c	Pixhawk 4 Mini
Companion computer	Raspberry Pi4 model B	Raspberry Pi4 model B
Camera	Intel RealSense Depth Camera D435	Raspeberry Pi Camera
Radio Control Device	RadioLink AT9S pro	RadioLink AT9S pro
Weight	3.5kg	0.5kg

Table 4.1: An overview of the major components that characterize the two multi-rotor UAV platforms used in the experiments.

with a downward-facing camera to capture data from the AprilTags, which constituted the map. This map was positioned on the floor of the flying area within the SPARCS laboratory and measured $4.9m \times 3.45m$. Its generation followed the pattern detailed in Section 2.2.3. Figure 4.2 showcases the AprilTag map generated for subsequent experimental tests.

To evaluate the accuracy of the localization system, a more precise reference for localization is essential. This reference is provided by the *Vicon Motion Capture system*, a technology designed for high-precision tracking of both the position and orientation of objects or individuals within its three-dimensional field of view. The Vicon system comprises a network of strategically placed cameras within the area of interest. These cameras are meticulously calibrated and synchronized to capture synchronized images of the target from various viewing angles. For proper recognition and tracking by the cameras, the target objects must be equipped with markers, in this case, small light reflectors. The data

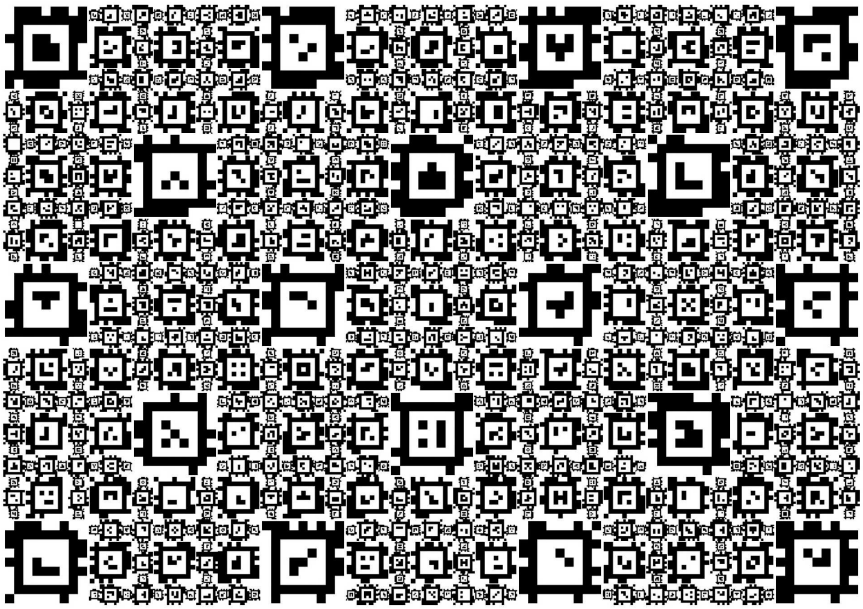


Figure 4.2: The complete map of dimensions ($4.9m \times 3.45m$) retained inside the *SPARCS (Space aerial and ground control system)* laboratory of the *Dipartimento di Ingegneria dell'informazione*, University of Padova.

acquired by the cameras are subsequently processed in real-time through dedicated software, which calculates the comprehensive pose of the target by merging information obtained from each camera's perspective.

The Motion Capture system in use within the SPARCS laboratory consists of ten cameras uniformly positioned around the ceiling border, thereby encircling the AprilTag map. This system offers sub-millimeter localization accuracy, operating at a frequency of approximately 100Hz. Moreover, it can be integrated into the ROS network through a dedicated node, allowing it to publish the estimated position of the target. These estimates can be either stored or directly employed to provide input to the EKF and close the feedback control loop of the UAVs. This operational scenario is commonly referred to as "off-board localization flight." Conversely, when the Vicon data is not utilized for the flight of the multi-rotor platforms (which instead rely on the on-board camera sensor to complete the feedback loop), it is termed an "on-board localization flight." It is noteworthy that the center $\mathbf{O}_V$ of the Vicon's reference system $\mathcal{F}_V$ coincides with the center $\mathbf{O}_W$ of the reference system of the map (referred to as the world frame $\mathcal{F}_W$). However, it's important to note that while the world frame's z-axis points downwards through the ground, the Vicon reference system's z-axis is oriented

upwards, towards the ceiling.

4.2 Experiments Design

This experimental campaign is conducted with the primary objective of assessing and validating the performance of the new localization system under various conditions, encompassing both stationary and dynamic scenarios. Performance evaluation rests on the localization error, specifically in terms of both vehicle position and orientation, by comparing the estimates provided by the Vicon system (serving as the ground truth) with those from the VIO system. To be precise, at each time instance t, the errors are computed as follows:

$$e_{\mathbf{p}}(t) = \| \hat{\mathbf{p}}^{\text{Vicon}}(t) - \hat{\mathbf{p}}^{\text{VIO}}(t) \| \tag{4.1}$$

$$e_{\angle}(t) = d_{\angle}\big(\hat{\mathbf{R}}^{\text{Vicon}}(t), \hat{\mathbf{R}}^{\text{VIO}}(t)\big) \tag{4.2}$$

The errors are often summarized using their mean and standard deviation. The willingness would be to achieve a significant enhancement compared to the previous VIO approach validated in [5] (strategy referred to as **JB-ALG** hereinafter), both in static and dynamic conditions. The experiments are divided into two distinct phases.

"**Hovering tests**": The initial phase of the experimental campaign exclusively involves the hexarotor HR01. The primary objective of this set of experiments is to compare and evaluate the performance of the algorithms detailed in Sections 3.1 and 3.2 to identify the optimal configuration in terms of pose estimation accuracy. During this phase, Vicon pose estimates are employed in the feedback control loop (*off-board localization*), with VIO estimates being collected simultaneously for subsequent offline performance assessment.To introduce variability, flights are conducted at different altitudes, each corresponding to varying numbers of tags of different sizes. Specifically, the UAV performs hovering flights at altitudes of 0.8 meters, 1.4 meters, and 2 meters above the ground. Throughout these flights, the vehicle maintains a constant orientation and zero linear velocity and acceleration. Six independent tests are executed, with each test initiated from a distinct position within the map. Notably, the last test, labeled as **Lc45**, involves the UAV flying over an L-sized marker encircled by smaller markers while maintaining a yaw angle of 45° with respect $\mathcal{F}_W$ (and therefore the underlying

Test ID	*Description*
XL	Take-off over one of the biggest-size markers
Ld	Take-off over an L-size marker placed in the diagonal between two XLs
Ls1	Take-off over an L-size marker placed just to the right of an XL one
Ls2	Take-off over an L-size marker placed just above an XL one
Lc	Take-off over an L-size marker centred inside a square of six
Lc45	Same position as **Lc**, but maintaining a yaw angle of 45°

Table 4.2: Description of the take-off position and orientation maintained in each hovering flight test performed with the HR01.

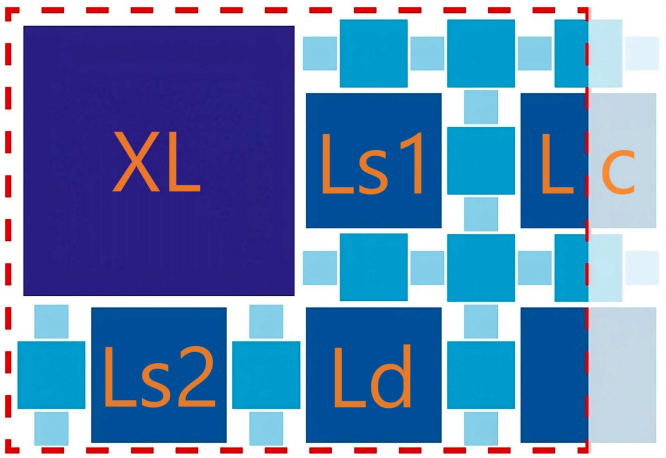


Figure 4.3: Visual representation of the hovering flights tests take-off positions inside the map.

tag). This particular test, **Lc45**, serves the purpose of evaluating the localization system under diverse conditions and map perspectives. For reference, the six take-off positions and orientations, along with their respective acronyms used as identifiers, are concisely summarized in Table 4.2, and a visual representation is presented in Figure 4.3.

"**Dynamic trajectories tests**": The subsequent phase of the experimental campaign involves both the HR01 and QR01 aerial platforms, utilizing the optimal configurations identified during Phase 1. Notably, during this phase, Vicon pose estimates are collected for comparative purposes but are not employed for feedback in flight control. Instead, the flight controller relies exclusively on estimates derived from the onboard camera and the IMU (*onboard localization*). In these tests, both vehicles are tasked with following predefined trajectories designed to assess performance under diverse conditions. Three distinct

trajectories have been designed for evaluation. These trajectories consist of set points generated using a fifth-order polynomial (*spline*) approximation, aiming to ensure their smoothness. The points are relayed to the PX4 flight controller through `offboard_control`, a ROS2 node developed explicitly for this purpose and fully configurable via a designated `.yaml` configuration file. The initial two trajectories, Square (**T1**) and Steps (**T2**), have been adapted from previous work found in [5] and [29]. In the Square trajectory, the vehicle is required to trace a 2-meter square over the map while maintaining a consistent altitude. In contrast, the Steps trajectory involves the vehicle following a constant reference in the x and y axes while altering its flight altitude along the z-axis. This trajectory resembles the one employed in hovering tests but incorporates variations in altitude, which occur sequentially as follows: 0.8m, 1m, 1.3m, 1.6m, 1.3m, 1m, and 0.8m. To facilitate a more comprehensive evaluation of the localization system, both the Square and Steps trajectories are subdivided into phases. For the Square trajectory, each phase (F - forward, R - right, B - backwards, L - left) corresponds to one side of the square. For the Steps trajectory, each phase can be categorized as ascending (as A1, A2, A3) or descending (D1, D2, D3), including the take-off maneuver (S0). These movements correspond to the transition between successive altitudes with steps of 0.3m so that the UAV's height from the ground varies from 0.7 during S0 to 1.6m before starting to descend. A novel trajectory, denoted as the *S-trajectory* (**TS**), has been introduced, requiring the vehicle to trace an S-shaped path on the xy plane to $\mathcal{F}_W$, while also varying its altitude. This trajectory represents the most complex and comprehensive movement within the testing campaign. Multiple tests have been conducted in correspondence to each different path considered. The final performance indices have been then obtained by averaging all the single tests performed on the same trajectory. For visual reference, Figure 4.4 provides a graphical depiction of the three trajectories.

Following the execution of the experiments, the flight data is systematically collected and archived within ROS2 bags. These data can be readily extracted using various tools, such as *PlotJuggler2*, or extracted directly using ROS2 echo commands and subsequently saved in `.csv` file format. These `.csv` files are conveniently prepared for importation into MATLAB for subsequent analysis. Specific topics have been preserved for this purpose:

- `/fmu/in/trajectory_setpoints`, messages containing the reference tra-

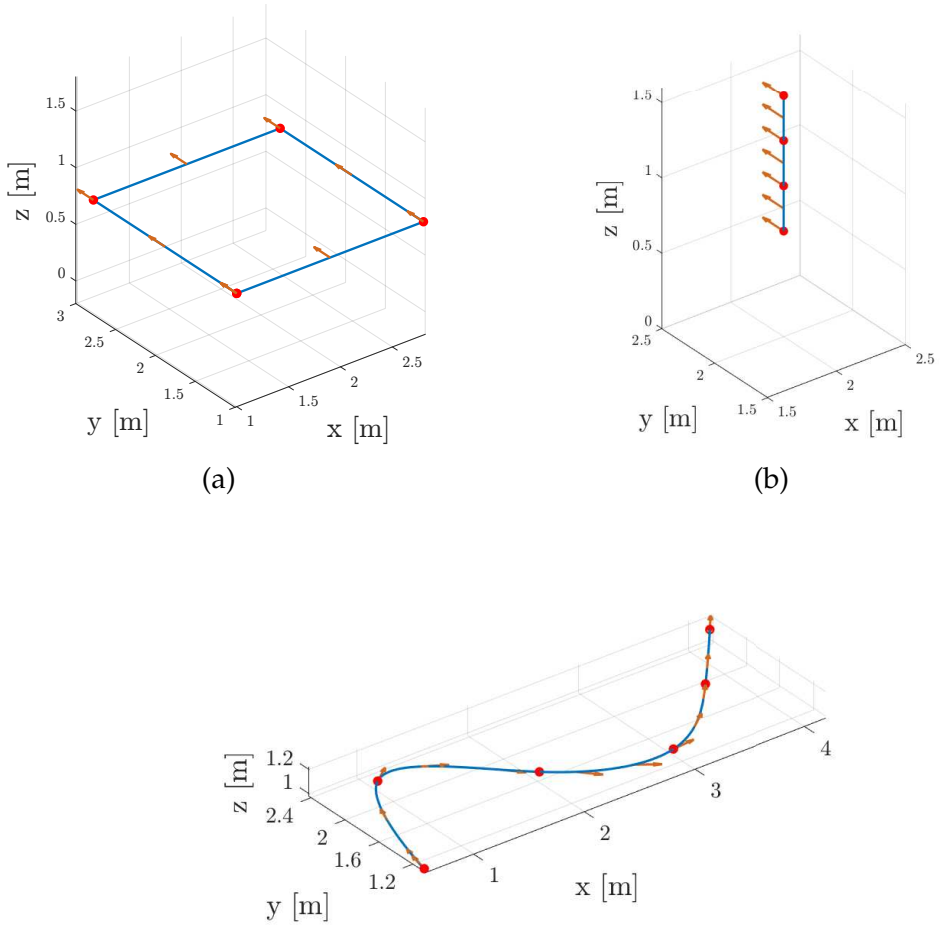


Figure 4.4: A graphical representation of the three trajectories adopted in the second phase of the tests. In this order, are shown: Square trajectory (**T1**), Steps trajectory (**T2**), S-trajectory (**TS**). The orange arrows highlight the prescribed orientation for the vehicles.

jectory set-points generated by the `offboard_control` node.

- `/fmu/in/vehicle_vicon_odometry`, messages containing the pose estimate provided by the Vicon system.
- `/fmu/in/vehicle_visual_odometry`, messages published by A2VO containing the pose estimate calculated through VIO localization.
- `/fmu/out/vehicle_odometry`, messages containing the final vehicle pose provided at the exit of the EKF.
- `tf_vio`, output of the `apriltag_ros` node, necessary to replicate offline the flight conditions inside simulation.

4.3 Results discussion

In this section, a comprehensive presentation and analysis of the results derived from the flight experiments will be provided. The emphasis will be placed on performance metrics, particularly estimation accuracy, for both Phase 1 (*hovering flights*) and Phase 2 (*dynamic trajectory flights*). Subsequently, attention will be directed towards assessing the computational complexity of the algorithms.

4.3.1 Phase 1: Hovering flights

After having conducted all the hovering tests with the HR01 within the laboratory setup outlined in Section 4.1 with the assistance of Vicon guidance in the flight control, the VO localization system will be validated by post-processing the recorded data. This validation will make use of data collected from ROS2 Bags to mimic the original flight conditions within a simulation environment. Each iteration of the simulation will assess a different algorithm.

Simulations have the objective of comparing various OR methods across the six take-off and hovering conditions to determine the optimal approach. In order to maintain fairness in the comparison of algorithms, all other VIO parameters remain unaltered throughout each simulation. These parameters include:

- Default usage of **QL2-AVG** as the rotations averaging algorithm.
- Default choice of **static W2** as the set of weights.
- The FIR filter remains **disabled**.

By varying only the OR method employed, a balanced assessment of the algorithms is ensured. Evaluation of the position component of the pose employs the Euclidean distance error between the marker-based estimate and the Vicon estimate, denoted as $\| \mathbf{e}_p(t) \|$. For orientation, the geodesic distance between the two estimates, represented as $e_\angle(t)$, is used as the second performance index. The considered outliers removal algorithms encompass:

- **NO-OR** (all OR methods disabled)
- **DMN-OR**
- **DMD-OR**
- **IQR-OR** (as implemented in *A2VO-v3*)
- **OBS-OR** (only)
- **TBS-OR** (only) **HYB-OR**, a hybrid solution involving the sequential execution of TBS-OR + IQR-OR.

Tables 4.5 presents the mean and standard deviation of errors from the initial comparison of OR algorithms. Notably, the results corresponding to **NO-OR** highlight the poorest performance registered, underlining the significance of introducing an outliers removal procedure before proceeding with the averaging of transformations.The best outcomes are consistently achieved by methods focusing solely on larger-sized markers (**OBS-OR**, **TBS-OR**, **HYB-OR**). Additionally, the strategy of employing a hybrid approach, which preemptively removes smaller-sized markers while simultaneously conducting regular outliers detection on a restricted sample set, proves to be effective. In terms of Euclidean distance errors, the **HYB-OR** method consistently yields the best results. Conversely, for angular distance errors, different methods yield the minimum values in various test scenarios. Nevertheless, once more, the hybrid approach demonstrates the most favorable overall performance, considering both the position and angular errors.

The following step involves the selection of the rotation averaging method alongside a suitable set of weights. Considered in this analysis are the CL_2-Mean and the QL_2-Mean, accompanied by two distinct sets of weights (**W1** and **W2**), as expounded in Section 3.4.3. Moreover, the alternative strategy of dynamic weighting is assessed, denoted as **D1** and **D2** when utilized in conjunction with

	NO-OR	DMN-OR	DMD-OR	IQR-OR	OBS-OR	TBS-OR	HYB-OR
				Position error $\mathbf{e}_p(t)$			
XL	18.03 ± 13.08	13.8 ± 10.46	16.72 ± 13.33	8.88 ± 6.45	3.79 ± 1.94	4.71 ± 2.72	**3.34 ± 1.76**
Ld	17.14 ± 12.21	14.3 ± 9.93	17.5 ± 11.95	9.11 ± 5.79	4.39 ± 2.46	5.28 ± 2.69	**2.95 ± 1.69**
Ls1	17.8 ± 12.81	13.61 ± 9.84	16 ± 11.48	8.47 ± 5.97	4.55 ± 2.24	4.59 ± 2.54	**3.01 ± 2**
Ls2	18.5 ± 13.65	14.65 ± 10.97	18.6 ± 13.28	8 ± 5.61	4.34 ± 3.62	3.26 ± 2.36	**2.73 ± 1.76**
Lc	16.35 ± 8.08	12.05 ± 5.34	12.62 ± 3.81	7.07 ± 2.44	3.86 ± 1.85	3.21 ± 1.55	**3.05 ± 1.2**
Lc45	29.35 ± 3.37	23.08 ± 4.11	28.5 ± 6.87	11.21 ± 2.73	5.03 ± 4	4.4 ± 1.73	**3.61 ± 1.6**
				Angular error $e_\angle(t)$			
XL	7.82 ± 2.19	5.92 ± 2	6.22 ± 2.68	4.29 ± 1.05	3.88 ± 0.7	**3.2 ± 0.4**	3.24 ± 0.41
Ld	4.67 ± 2.2	**3.52 ± 1.72**	4.19 ± 2.15	3.45 ± 1.19	3.72 ± 1.06	4.72 ± 0.99	4 ± 0.68
Ls1	4.95 ± 1.86	4.17 ± 1.18	4.47 ± 1.4	**3.77 ± 1.03**	5.99 ± 0.63	5.06 ± 0.7	4.5 ± 0.63
Ls2	7.43 ± 2.24	5.33 ± 2.18	5.54 ± 2.77	3.82 ± 1.02	3.05 ± 0.66	**3 ± 0.37**	3.19 ± 0.47
Lc	4.84 ± 2.8	4.8 ± 1.95	4.62 ± 1.6	4 ± 0.58	4.29 ± 0.22	**3.84 ± 0.5**	3.98 ± 0.44
Lc45	6.86 ± 1.14	7.75 ± 1.2	7.48 ± 1.19	6.83 ± 0.97	3.8 ± 0.56	5.03 ± 0.55	**4.92 ± 0.56**

Table 4.3: Comparison of the mean and the standard deviation of the Euclidean distance error $\| \mathbf{e}_p(t) \|$ and the the angular distance error $e_\angle(t)$ obtained from the Phase 1 offline simulations, varying the outliers removal strategy. The errors are reported in cm and degrees, respectively.

the base sets of weights **W1** and **W2**, respectively. It should be noted that the selected type of weighting influences not only the averaging of transformations but also impacts calculations of the quartiles within the **IQR-OR** algorithm. The parameters that remain consistent across these simulations encompass:

- **HYB-OR** as the optimal outliers removal algorithm.
- The FIR filter remains **disabled**.

Tables 4.4 detail the mean value and standard deviation of errors obtained during this second comparison of the averaging algorithms. Numerical results indicate that, in terms of orientation estimate precision, the **QL2-AVG** algorithm marginally outperforms **CL2-AVG**. Interestingly, dynamic weighting demonstrates limited effectiveness, as its performance closely resembles that of static weighting. This can be attributed to the fact that the relative differences in distance between the various recognized tags and the camera are not substantial enough to significantly impact the estimation process. Regarding position estimate errors, none of the methods exhibit significant superiority over the others. However, in terms of orientation estimate, the combination **QL2-AVG + W2** appears to offer a more balanced outcome across the results of the six different

	CL2+W1	**CL2+W2**	**CL2+D1**	**CL2+D2**	**QL2+W1**	**QL2+W2**	**QL2+D1**	**QL2+D2**
					Position error $\mathbf{e}_p(t)$			
XL	3.2 ± 1.72	3.34 ± 1.76	**3.17 ± 1.67**	3.35 ± 1.76	3.2 ± 1.72	3.34 ± 1.76	**3.17 ± 1.67**	3.35 ± 1.75
Ld	**2.92 ± 1.48**	2.95 ± 1.69	2.96 ± 1.48	2.93 ± 1.67	**2.92 ± 1.48**	2.95 ± 1.69	2.94 ± 1.68	2.96 ± 1.48
Ls1	3.45 ± 1.53	**3.01 ± 2**	3.44 ± 1.54	3.37 ± 2.2	3.45 ± 1.53	**3.01 ± 2**	3.37 ± 2.2	3.44 ± 1.54
Ls2	3.61 ± 1.88	2.73 ± 1.76	3.58 ± 1.84	2.73 ± 1.74	3.6 ± 1.88	2.73 ± 1.76	3.58 ± 1.84	**2.73 ± 1.73**
Lc	3.07 ± 1.26	**3.03 ± 1.24**	3.05 ± 1.28	3.09 ± 1.12	3.07 ± 1.25	3.05 ± 1.2	3.1 ± 1.1	3.05 ± 1.28
Lc45	3.07 ± 1.36	3.6 ± 1.6	**2.94 ± 1.31**	3.79 ± 1.73	3.07 ± 1.37	3.61 ± 1.6	**2.94 ± 1.31**	3.79 ± 1.73
					Angular error $e_\angle(t)$			
XL	4.21 ± 6.73	3.9 ± 8.64	4.16 ± 5.61	3.73 ± 5.75	3.48 ± 0.44	**3.24 ± 0.41**	3.51 ± 0.44	3.25 ± 0.41
Ld	4.61 ± 6.7	4.5 ± 3.55	4.63 ± 5.83	4.92 ± 7.78	3.94 ± 0.72	4 ± 0.68	4.01 ± 0.67	**3.94 ± 0.71**
Ls1	6.04 ± 7.48	5.2 ± 7.36	6.14 ± 7.2	5.79 ± 9.96	5.17 ± 0.63	**4.5 ± 0.63**	4.71 ± 0.64	5.18 ± 0.63
Ls2	3.7 ± 5.45	3.91 ± 6.51	3.61 ± 3.93	3.79 ± 5.67	**3.04 ± 0.54**	3.19 ± 0.47	**3.04 ± 0.54**	3.2 ± 0.47
Lc	4.07 ± 0.17	4.82 ± 1.58	4.06 ± 0.17	5.32 ± 1.98	4.05 ± 0.18	**3.98 ± 0.44**	4.03 ± 0.45	4.06 ± 0.18
Lc45	5.53 ± 7.8	5.85 ± 7.2	5.7 ± 8.55	6.19 ± 7.93	**4.44 ± 0.52**	4.92 ± 0.56	4.46 ± 0.53	5.02 ± 0.62

Table 4.4: Comparison of the mean and the standard deviation of the Euclidean distance error $\| \mathbf{e}_p(t) \|$ and the the angular distance error $e_\angle(t)$ obtained from the Phase 1 offline simulations, varying the rotations averaging method and the weighting. The errors are reported in cm and degrees, respectively.

tests performed. Consequently, the configuration **QL2-AVG + W2** is selected as the optimal choice.

The final step involves the evaluation of the FIR filter's impact while varying the filter's order and the set of weights used for its samples. To accomplish this, the weights are defined as $\begin{bmatrix} w_{t-4} & w_{t-3} & w_{t-2} & w_{t-1} & w_t \end{bmatrix}$. The following configurations are examined:

- Configuration **F0**: FIR filter is **disabled**.
- Configuration **F1**: FIR filter is **enabled**, with a 3rd order, and weights are set to $[1, 1, 1]$.
- Configuration **F2**: FIR filter is **enabled**, with a 4th order, and weights are set to $[1, 1, 1, 1]$.
- Configuration **F3**: FIR filter is **enabled**, with a 5th order, and weights are set to $[1, 1, 1, 1, 1]$.
- Configuration **F4**: FIR filter is **enabled**, with a 4th order, and weights are set to $[1, 1, 2, 2]$.
- Configuration **F5**: FIR filter is **enabled**, with a 5th order, and weights are set to $[1, 2, 2, 3, 3]$.
- Configuration **F6**: FIR filter is **enabled**, with a 5th order, and weights are set to $[1, 2, 3, 4, 5]$.

The remaining configuration parameters are held constant, employing the **HYB-OR** algorithm alongside **QL2-AVG** + **W2**.

	F0	F1	F2	F3	F4	F5	F6
				Position error $\mathbf{e}_p(t)$			
XL	3.34 ± 1.76	3.07 ± 1.51	3.03 ± 1.47	**3 ± 1.44**	3.04 ± 1.48	3.01 ± 1.45	3.02 ± 1.46
Ld	2.95 ± 1.69	2.52 ± 1.26	2.44 ± 1.17	**2.39 ± 1.12**	2.45 ± 1.19	2.4 ± 1.14	2.41 ± 1.15
Ls	3.01 ± 2	2.63 ± 1.66	2.56 ± 1.59	**2.51 ± 1.54**	2.57 ± 1.61	2.52 ± 1.56	2.54 ± 1.58
Lsu	2.73 ± 1.76	2.35 ± 1.48	2.3 ± 1.42	**2.27 ± 1.38**	2.31 ± 1.44	2.28 ± 1.4	2.29 ± 1.41
Lc	3.05 ± 1.2	**2.82 ± 1.27**	2.96 ± 1.38	3.23 ± 1.42	2.88 ± 1.32	3.02 ± 1.36	2.95 ± 1.33
Lc45	3.61 ± 1.6	3.12 ± 1.21	3.03 ± 1.13	**2.98 ± 1.08**	3.04 ± 1.15	2.99 ± 1.1	3 ± 1.11
				Angular error $e_\angle(t)$			
XL	3.24 ± 0.41	**3.21 ± 0.36**	3.21 ± 0.37	3.21 ± 0.38	**3.21 ± 0.36**	**3.21 ± 0.36**	**3.21 ± 0.36**
Ld	4 ± 0.68	3.98 ± 0.55	3.98 ± 0.53	**3.97 ± 0.52**	3.98 ± 0.53	**3.97 ± 0.52**	**3.97 ± 0.52**
Ls	4.5 ± 0.63	4.48 ± 0.49	4.48 ± 0.46	**4.48 ± 0.45**	4.48 ± 0.47	**4.48 ± 0.45**	**4.48 ± 0.45**
Lsu	3.19 ± 0.47	3.16 ± 0.41	3.16 ± 0.41	3.16 ± 0.41	**3.16 ± 0.4**	**3.16 ± 0.4**	**3.16 ± 0.4**
Lc	**3.98 ± 0.44**	3.99 ± 0.21	4.01 ± 0.21	4.06 ± 0.19	4 ± 0.23	4.02 ± 0.21	4.01 ± 0.21
Lc45	4.92 ± 0.56	4.9 ± 0.46	4.9 ± 0.44	4.9 ± 0.43	4.9 ± 0.43	**4.9 ± 0.42**	4.9 ± 0.43

Table 4.5: Comparison of the mean and the standard deviation of the Euclidean distance error $\| \mathbf{e}_p(t) \|$ and the the angular distance error $e_\angle(t)$ obtained from the Phase 1 offline simulations, varying the FIR filter's order and set of weights. The errors are reported in cm and degrees, respectively.

Finally, Tables **??** and **??** provide an overview of the mean and variance of the errors resulting from this last comparison. The tables illustrate that the application of the FIR filter can lead to a reduction in estimation error, both in position and orientation, by up to 20%. Among all the filter orders and sets of weights under consideration, Configuration **F3** (a 5th-order filter without weights) exhibits marginally superior results compared to the other configurations, particularly in terms of position estimation. Consequently, **F3** is selected for the optimal configuration, reaffirming the effectiveness of the previously adopted strategy in *A2VO-v2* when employing the FIR filter.

4.3.2 Phase 2: Dynamic trajectory tests

In light of the Phase 1 experiments, the optimal configuration for *A2VO-v3*, denoted as **OPT-ALG** encompasses the following settings:

- **HYB-OR** is the chosen method for removing outliers.
- **QL2-AVG** is employed as rotations averaging method alongside the static set of weights **W2**.
- A 5th-order FIR filter is activated without any additional weighting (**F3**).

	Phase		F	R	B	L
e_p	QR	**JB-ALG**	7.40 ± 2.63	9.64 ± 2.88	6.38 ± 2.99	5.71 ± 3.94
		OPT-ALG	$\mathbf{7.23 \pm 1.86}$	$9.72 \pm \mathbf{1.40}$	$8.00 \pm \mathbf{2.27}$	$\mathbf{2.62 \pm 1.13}$
	HR	**JB-ALG**	6.78 ± 3.38	8.67 ± 3.61	5.53 ± 2.21	10.27 ± 7.10
		OPT-ALG	$6.96 \pm \mathbf{1.76}$	$\mathbf{8.45 \pm 2.96}$	$\mathbf{5.00 \pm 1.98}$	13.10 ± 8.41
$e_\angle$	QR	**JB-ALG**	2.70 ± 0.22	2.29 ± 0.34	2.77 ± 0.26	3.09 ± 0.24
		OPT-ALG	$\mathbf{2.67 \pm 0.12}$	$2.41 \pm \mathbf{0.23}$	$\mathbf{2.66 \pm 0.21}$	$\mathbf{2.92 \pm 0.15}$
	HR	**JB-ALG**	6.43 ± 2.32	6.72 ± 2.95	5.42 ± 1.87	5.88 ± 3.86
		OPT-ALG	$\mathbf{4.31 \pm 1.17}$	$\mathbf{6.09 \pm 2.51}$	$\mathbf{3.70 \pm 1.00}$	$\mathbf{5.68 \pm 3.41}$

Table 4.6: Phase 2, trajectory T1 - mean and and standard deviation of the position error (in cm) and orientation (in °).

This configuration is consistently applied throughout the Phase 2 experiments, enabling a direct comparison of results with those obtained using the previous **JB-ALG** algorithm across all three prescribed trajectories.

Square trajectory

Figure 4.5 visually represents the position estimates for the square trajectory executed by the HR01. While analogous plots pertain to the QR01, they are omitted here for simplicity. Meanwhile, Table 4.6 provides comprehensive details on both position and orientation errors in the VIO estimates, effectively highlighting the various movements within the trajectory. A detailed examination of the table reveals a significant improvement in accuracy, particularly concerning orientation estimation. Comparative analysis of error values between the new **OPT-ALG** method and the older **JB-ALG** shows consistent advantages, including smaller mean and standard deviation values for $e_\angle$. Regarding position estimates, enhanced precision is observed in most cases, although an exception arises during the L movement of the HR01. This discrepancy may be attributed to an unfavorable section of the map, diminishing the reliability of smaller tags in this specific scenario.

Steps trajectory

Figure 4.6 provides a graphical depiction of the position estimates for the square trajectory executed by the HR01 (with the corresponding plots for the QR01 omitted for conciseness). In Table 4.7, the mean and variance of both posi-

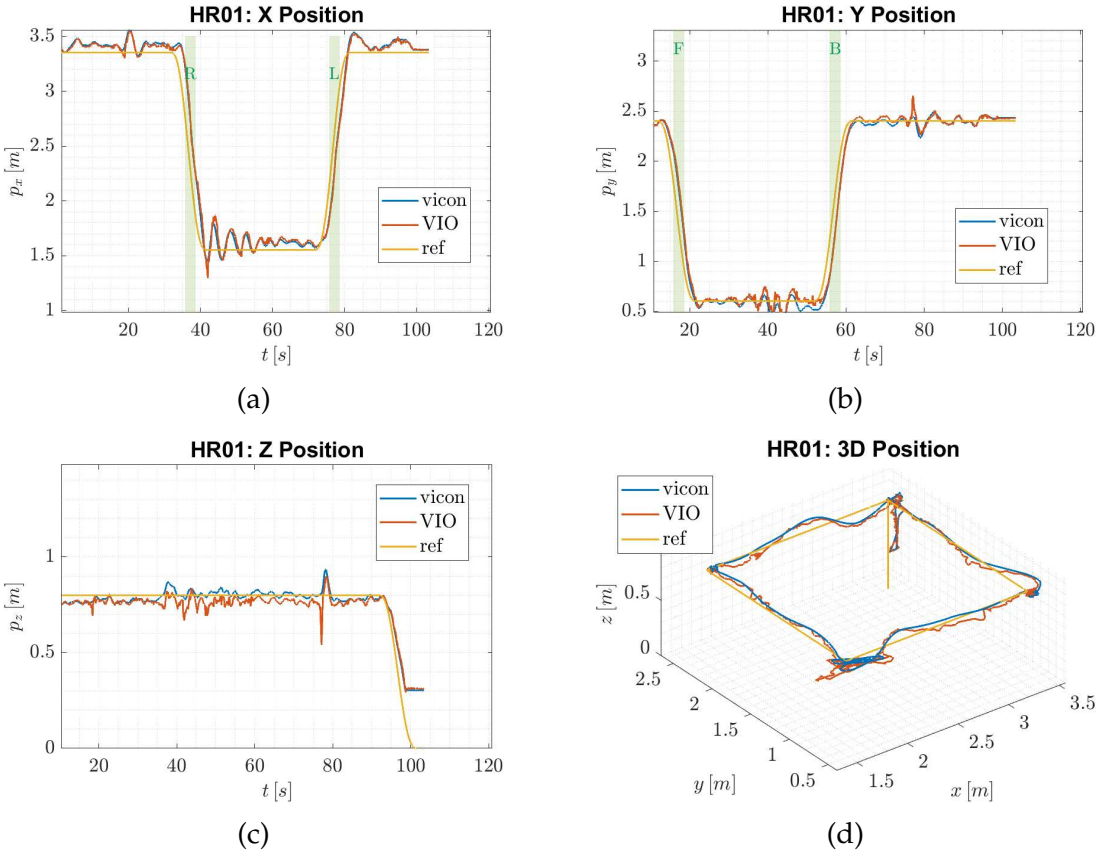


Figure 4.5: Trajectory reference and position estimates produced by the Vicon system and by the marker-based localization during the first test conducted on the HR01 platform, executing T1.

		Phase	S0	A1	A2	A3	D1	D2	D3
e_p	QR	**JB-ALG**	4.67 ± 1.66	8.09 ± 2.65	10.17 ± 4.96	10.72 ± 3.74	9.29 ± 3.98	7.60 ± 2.64	5.34 ± 1.38
		OPT-ALG	**2.89 ± 1.35**	**4.00 ± 0.90**	**4.47 ± 1.29**	**4.98 ± 1.39**	**4.31 ± 1.47**	**3.66 ± 1.23**	**3.39 ± 1.21**
	HR	**JB-ALG**	4.48 ± 1.66	6.60 ± 1.66	9.70 ± 2.77	12.85 ± 4.96	9.72 ± 2.76	7.23 ± 1.72	4.81 ± 1.84
		OPT-ALG	**1.65 ± 0.68**	**2.88 ± 1.06**	**3.87 ± 1.18**	**5.41 ± 1.92**	**3.93 ± 1.60**	**2.85 ± 0.83**	**1.88 ± 0.75**
$e_\angle$	QR	**JB-ALG**	2.75 ± 0.17	2.54 ± 0.36	3.15 ± 0.45	2.59 ± 0.28	3.18 ± 0.41	2.44 ± 0.43	2.77 ± 0.18
		OPT-ALG	2.75 ± **0.15**	2.64 ± **0.18**	**2.92 ± 0.23**	2.81 ± **0.23**	**2.93 ± 0.26**	2.60 ± **0.26**	**2.63** ± 0.19
	HR	**JB-ALG**	5.92 ± 1.69	6.23 ± 0.96	6.03 ± 1.42	6.24 ± 1.70	6.22 ± 1.28	6.43 ± 0.90	4.86 ± 1.72
		OPT-ALG	**3.90 ± 0.64**	**4.36 ± 0.45**	**3.45 ± 0.68**	**3.75 ± 0.73**	**3.63 ± 0.66**	**4.45 ± 0.41**	**3.57 ± 0.66**

Table 4.7: Phase 2, trajectory T2 - mean and and standard deviation of the position error (in cm) and orientation (in °).

tion and orientation errors in the VIO estimates are presented, emphasizing the distinct movements within the trajectory. A detailed examination of the table reveals noticeable enhancements in position estimate accuracy for both UAVs, as evidenced by substantial reductions in the Euclidean distance error, e_p, across all movements. The error is reduced by up to 50% for maneuvers performed at higher altitudes (A3 and D1) In contrast, improvements in orientation estimates vary. While the HR01 consistently benefits from reduced orientation errors, the QR01 experiences mixed results, with error reductions often manifesting in either mean values or standard deviations for each movement.

S-TRAJECTORY

Finally, Figure 4.7 presents the graphical depiction of the position estimate for the S-trajectory in the flight conducted with the HR01. Table 4.8 provides details on the mean and variance of errors in both position and orientation of the VIO estimate. From the table, a modest improvement in accuracy is observed overall for both UAVs. However, it's noteworthy that the **JB-ALG** still outperforms in terms of orientation estimation for the QR01. In all other cases, there is a reduction in error, albeit less pronounced compared to the other two trajectories, especially the steps trajectory. This observation suggests that the proposed strategy may have limitations in effectiveness when dealing with faster and more demanding flight dynamics.

4.3.3 NOTES ABOUT ALGORITHMS EXECUTION TIME

The measurement of execution times for the principal algorithms integrated into *A2VO-v3* was performed using inbuilt tools provided by the ROS2 C++ library. All the execution times relate to complete flight simulations conducted

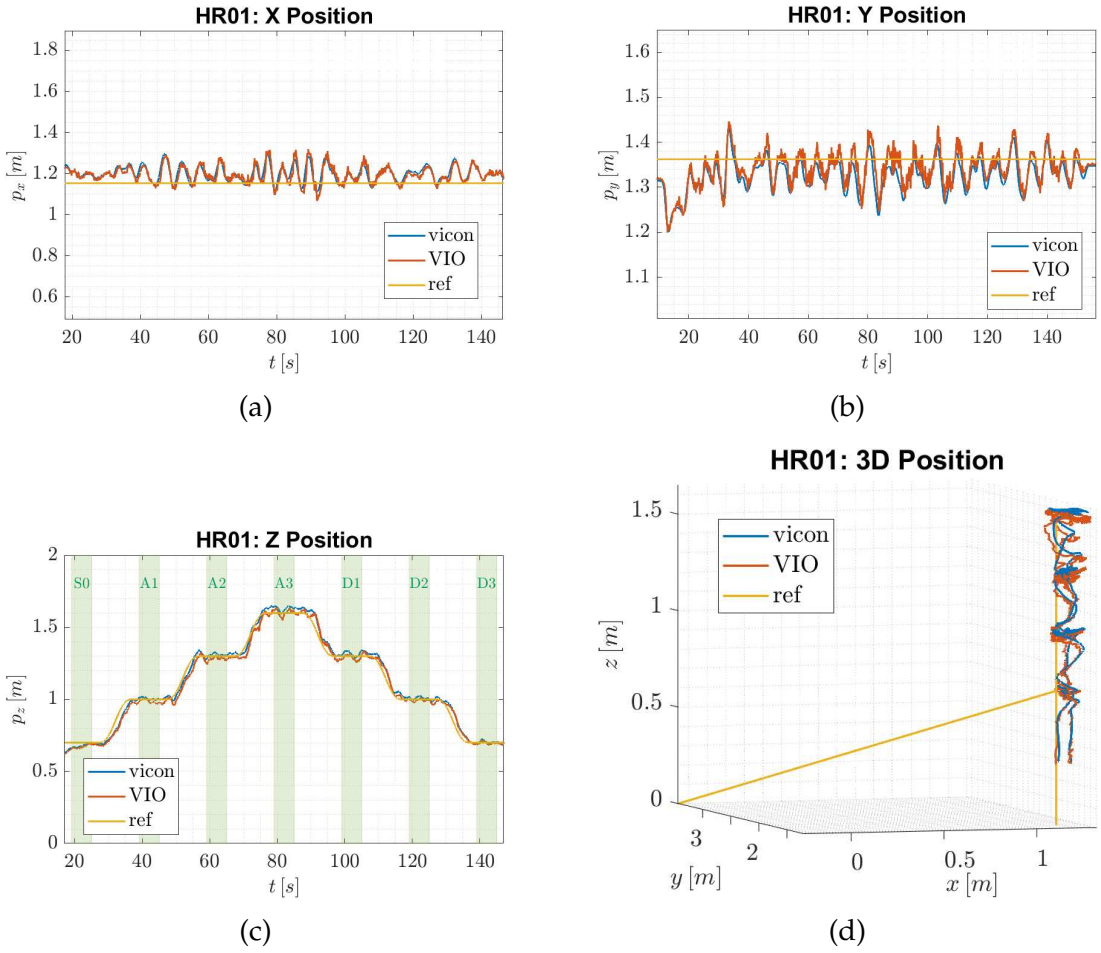


Figure 4.6: Trajectory reference and position estimates produced by the Vicon system and by the marker-based localization during the first test conducted on the HR01 platform, executing T2.

		Entire S path	
e_p	QR	**JB-ALG**	37.35 ± 16.91
		OPT-ALG	**36.38 ± 15.08**
	HR	**JB-ALG**	12.65 ± 7.78
		OPT-ALG	**10.65 ± 5.67**
$e_\angle$	QR	**JB-ALG**	3.01 ± 3.04
		OPT-ALG	3.32 ± 3.28
	HR	**JB-ALG**	5.74 ± 2.91
		OPT-ALG	**5.27 ± 2.30**

Table 4.8: Phase 2, trajectory TS - mean and and standard deviation of the position error (in cm) and orientation (in °).

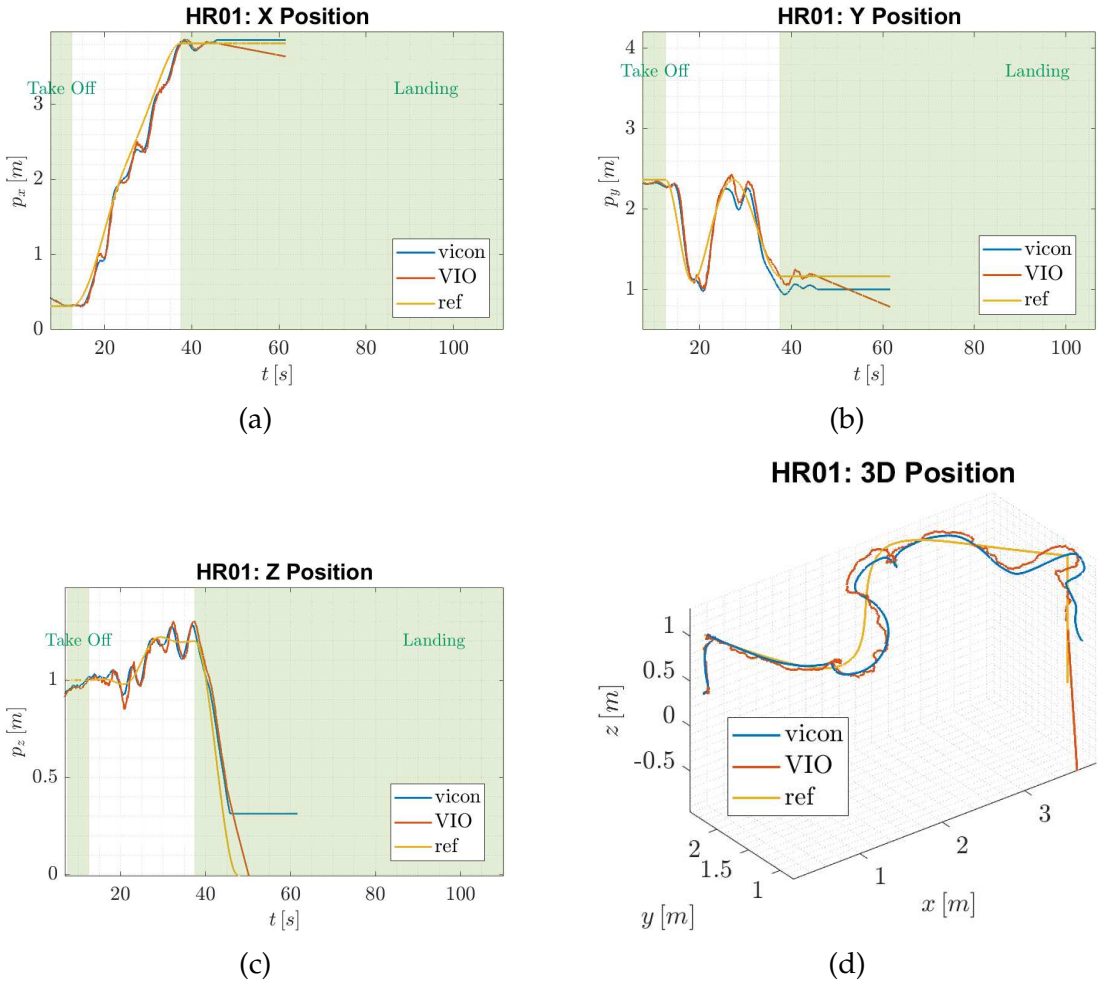


Figure 4.7: Trajectory reference and position estimates produced by the Vicon system and by the marker-based localization during the first test conducted on the HR01 platform, executing TS.

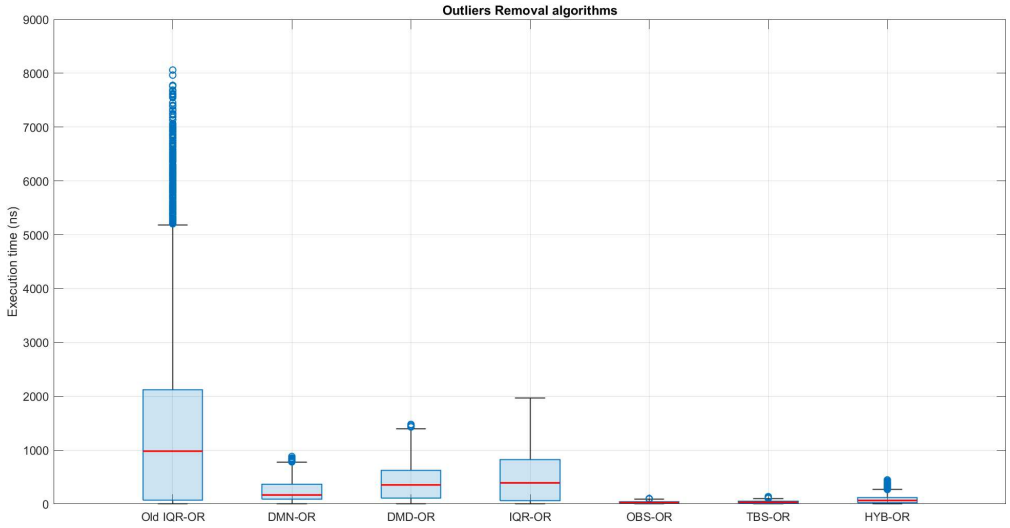
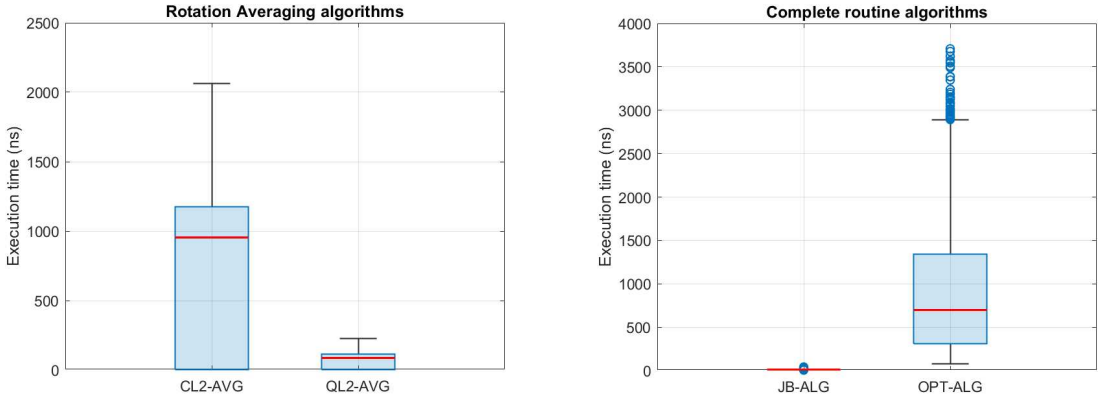


Figure 4.8: Execution times that have been obtained for each different OR algorithm through simulation over the same complete recorded flight.

offline on a workstation (not on the Raspberry Pi). The simulation environment runs a ROS2 Humble distribution on Ubuntu 20.04. The times presented below are not indicative of the real execution times required on the target platform but rather offer valuable insights into the relative complexity differences among the exposed algorithms.

The initial simulation aims to compare the performance of each individual algorithm. This includes both outlier removal methods and averaging methods, along with their respective implementations. Figure 4.8 provides the mean values and standard deviations of execution times for each outlier removal algorithm over a full flight simulation. The considered implementations encompass Pescante's **IQR-OR** (from *A2VO-v2*), **DMN-OR**, **DMD-OR**, **IQR-OR** (*A2VO-v3*), **OBS-OR** (only), **TBS-OR** (only), and **HYB-OR**. Figure 4.9a, on the other hand, presents a comparison of execution times achieved with the two different rotation averaging algorithms (**CL2-AVG** and **QL2-AVG**) during the same complete flight simulation when **TBS-OR** is applied.

It is unsurprising that the best-performing OR algorithms are those restricting themselves to the larger classes of tags, including **OBS-OR** (average of $30.7ns$), **TBS-OR** (average of $34.8ns$), and **HYB-OR** (average of $95.3ns$). Notably, the disparity between the older **IQR-OR** implementation from *A2VO-v2* ($1594ns$ avg) and the new one ($490ns$ avg) confirms the substantial detrimental impact that



(a) (b)

Figure 4.9: Execution times that have been obtained for each different AVG algorithm (Figure 4.9a) and by measuring the total execution time of the routine (Figure 4.9b) through simulation over the same complete recorded flight.

using standard C++ classes can have on performance in such tasks. Also noteworthy is the decrease in complexity as the set of markers for processing is confined to the largest-size classes (as evident in the comparison between the new **IQR-OR** version and **HYB-OR**). When contrasting the performance of **CL2-AVG** and **QL2-AVG**, one can appreciate the lighter computational load of the Quaternion L_2-Mean, as previously anticipated in theoretical considerations in Section 3.2.

A final simulation was conducted to compare the total execution time of the entire `tf_callback()` routine between the **OPT-ALG** and **JB-ALG** methods. The results are depicted in Figure 4.9b. It is immediately evident how the complexity of the callback routine has increased with the introduction of OR and AVG procedures, resulting in a significant increment in total execution time (from an average of $13.9ns$ to $959.5ns$). In the pursuit of enhancing and refining the localization system, it is crucial to consider the computational complexity aspects of the adopted strategies. This consideration becomes even more vital when dealing with hardware-constrained platforms. In dynamic applications like UAV flights, delays introduced by the addition of new algorithms can lead to system instability or negate the benefits brought by these new features. Active code optimization and the quest for the optimal balance between advanced algorithms and their computational costs have been essential in achieving the best localization performance within the confines of the hardware's capabilities.

5

Conclusions and Future Works

This thesis has been centered on the examination and enhancement of a previously developed Visual-Inertial Odometry (VIO) localization system tailored for UAVs navigating indoors. This approach relies on recognizing AprilTags situated on the floor and constructing a dense map for navigation. The primary contribution of this work lies in the thorough investigation of the challenges encountered in harnessing data from multiple AprilTag recognitions. It encompasses the creation of novel algorithms and the evolution of the ROS2 node, `apriltag_to_visual_odometry`, often abbreviated in *A2VO-v3*, a pivotal component in marker-based VIO localization.

Specifically, substantial enhancements have been made to the data pre-processing, an essential step to eliminate outliers from the estimates generated by the fiducial marker detector. New algorithms have been incorporated, providing the capability to restrict the system to consider estimates exclusively from the largest-sized tag sets, resulting in noteworthy enhancements in both accuracy and computational efficiency. Additionally, this work delves into the problem of averaging transformations, exploring an alternative mathematical approach to compute rotation means, thus introducing a more precise and efficient algorithm. Throughout these advancements, the entire *A2VO-v3* codebase has been scrutinized and refined, ensuring that the core structure remains unchanged to preserve backward compatibility.

The validation of this system entailed a meticulously designed series of experiments employing two distinct multi-rotor platforms: the HR01 and the QR01. These experiments encompassed three distinct flight trajectories, each characterized by varying degrees of static and dynamic maneuvers. This strategic approach was aimed at evaluating the system's reliability across diverse conditions. The initial phase of the tests involved controlled hovering flights of the HR01 over predefined sections of the map. Experiments outcomes obtained with various algorithms were subsequently compared to pinpoint the optimal configuration. Subsequently, this optimal configuration was employed in the second phase of the experiments, wherein both UAV platforms executed flights along three prescribed reference trajectories: a square trajectory, a vertical steps trajectory, and a more complex S-shaped trajectory. The evaluation was based on two performance indices: the Euclidean distance error and angular distance error computed by contrasting the marker-based localization estimates with the Vicon-derived estimates, serving as ground truth. Notably, the analysis considered both the mean value and standard deviation of these errors.

Phase 1 of the tests yielded valuable insights into the localization system's performance. A noteworthy confirmation was the system's independence from the visible portion of the map, irrespective of the node's configuration. This indicates that the achieved results were not contingent on a particularly advantageous UAV position during hovering. The most remarkable accuracy in pose estimation, encompassing both position and orientation, was attained through the combined utilization of **HYB-OR** for outlier removal, **QL2-AVG** in conjunction with **W2** for rotation averaging, with dynamic weights disabled and the FIR filter activated. In Phase 2, the newly optimized localization system (**OPT-ALG**) was assessed against the legacy localization framework (**JB-ALG**). This comparison led to two crucial observations:

Hardware Adaptability: The framework upgrade did not favor one platform over the other. Both UAVs experienced performance enhancements, albeit not uniformly across identical tests or maneuvers. This suggests that this approach can be effective with diverse hardware and platforms.

Trajectory Dependency: The improvements were most pronounced in predominantly static flights, such as those in Phase 1 and the steps trajectory, while dynamic trajectories like the S-trajectory exhibited limited accuracy gains.

Several factors might contribute to this phenomenon, including motion blur effects that can compromise the reliability of pose estimates derived from smaller tags. Additionally, the introduction of the more complex algorithm in the localization routine might have introduced delays (see 4.3.3). It's important to note that despite the efforts to optimize the new algorithms' computational complexity, the execution time of the new routine (**OPT-ALG**) significantly surpassed that of the old routine (**JB-ALG**). Future investigations should delve deeper into understanding how these delays impact localization accuracy and how to mitigate their effects.

5.1 Possible future works

This thesis is situated within a broader research endeavor centered on exploring the interaction between UAVs and humans. The ultimate goal is to facilitate the integration of these cutting-edge technologies into industrial and operational settings in a novel cooperative manner, with a particular emphasis on indoor applications. Localization represents a pivotal component of this research project, and the current VIO system can be further refined in the future. Several potential enhancements emerge from the findings presented in this thesis:

Delay Impact Assessment: Although the presence of an EKF at the end of the localization pipeline grants a certain level of robustness to the localization estimate, investigating the influence of computational load-induced delays on localization accuracy is essential. The primary sources of delay encompass the *A2VO* and `apriltag_ros` nodes and can introduce to delays in the order of ms. Optimizing the performance of `apriltag_ros` by conducting an in-depth examination of the AprilTags recognition and identification routine, along with associated algorithms, could be highly beneficial.

Exploring Advanced Fiducial Markers: The possibility of adopting more advanced types of fiducial markers merits consideration. The extensive literature on this subject and the continuous evolution of fiducial marker technology offer opportunities to enhance pose estimate precision and reliability. Comparative studies regarding new fiducial marker solutions, as [17], could yield valuable insights.

Mitigating Motion Blur Effects: Understanding how motion blur affects pose estimation during dynamic flight conditions is crucial. Developing strategies to mitigate the adverse effects of motion blur, potentially drawing inspiration from [31], could improve accuracy.

Camera Upgrades: Exploring the feasibility of transitioning from rolling shutter to global shutter cameras is advisable. Global shutter cameras can contribute to improving image quality during dynamic flight conditions.

Switch between different configurations: Assessing the viability of dividing the main algorithm into two or more operational configurations is worthwhile. A first configuration could be suited for hovering or slow maneuvers demanding maximum localization precision, leveraging complex and computationally intensive algorithms. A second configuration, tailored for dynamic contexts where speed takes precedence over localization accuracy, can rely on less sophisticated but more efficient estimation algorithms to minimize computational costs. The system may be programmed to embrace the best configuration depending on the required task.

Ultimately, to extend the applicability of this technology beyond the laboratory and into real-world autonomous applications in various contexts, the transition from a dense map of tags to a more practical and less expensive sparse map is the next step. This transition would involve reducing the total number of markers while maintaining robust navigational guidance.

References

[1] Syed Muhammad Abbas et al. "Analysis and Improvements in AprilTag Based State Estimation". In: *Sensors (Basel, Switzerland)* 19 (2019). URL: https://api.semanticscholar.org/CorpusID:209388315.

[2] Mohammad Soleimani Amiri and Rizauddin Bin Ramli. "Visual Navigation System for Autonomous Drone using Fiducial Marker Detection". In: *International Journal of Advanced Computer Science and Applications* (2022).

[3] Oualid Araar, Nabil Aouf, and Ivan Vitanov. "Vision Based Autonomous Landing of Multirotor UAV on Moving Platform". In: *Journal of Intelligent & Robotic Systems* 85 (2017), pp. 369–384.

[4] Muhammet Fatih Aslan et al. "A Comprehensive Survey of the Recent Studies with UAV for Precision Agriculture in Open Fields and Greenhouses". In: *Applied Sciences* (2022).

[5] Massimiliano Bertoni et al. "Indoor Visual-Based Localization System for Multi-Rotor UAVs". In: *Sensors (Basel, Switzerland)* 22 (2022).

[6] Yuchao Dai et al. "Rotation Averaging with Application to Camera-Rig Calibration". In: *Asian Conference on Computer Vision*. 2009. URL: https://api.semanticscholar.org/CorpusID:8185492.

[7] Gamini Dissanayake et al. "A solution to the simultaneous localization and map building (SLAM) problem". In: *IEEE Trans. Robotics Autom.* 17 (2001), pp. 229–241.

[8] Zheng Fang et al. "Marker-Based Mapping and Localization for Autonomous Valet Parking". In: 2020.

[9] William Rowan Sir Hamilton. "LXXVIII. On quaternions; or on a new system of imaginaries in Algebra: To the editors of the Philosophical Magazine and Journal". In: *Philosophical Magazine Series 1* 25 (), pp. 489–495. URL: https://api.semanticscholar.org/CorpusID:124321762.

[10] Richard I. Hartley, Khurrum Aftab, and Jochen Trumpf. "L1 rotation averaging using the Weiszfeld algorithm". In: *CVPR 2011* (2011), pp. 3041–3048. URL: https://api.semanticscholar.org/CorpusID:18686426.

[11] Richard I. Hartley et al. "Rotation Averaging". In: *International Journal of Computer Vision* 103 (2012), pp. 267–305. URL: https://api.semanticscholar.org/CorpusID:10021879.

[12] Daniel Hein et al. "Integrated UAV-Based Real-Time Mapping for Security Applications". In: *ISPRS Int. J. Geo Inf.* 8 (2019), p. 219.

[13] Van Tien Hoang. "AN INDOOR LOCALIZATION METHOD FOR MOBILE ROBOT USING CEILING MOUNTED APRILTAGS". In: *Journal of Science and Technique* (2022).

[14] Du Q. Huynh. "Metrics for 3D Rotations: Comparison and Analysis". In: *Journal of Mathematical Imaging and Vision* 35 (2009), pp. 155–164. URL: https://api.semanticscholar.org/CorpusID:14622215.

[15] S. M. Riazul Islam. "Drones on the Rise: Exploring the Current and Future Potential of UAVs". In: *ArXiv* abs/2304.13702 (2023).

[16] Youeyun Jung, Hyo-choong Bang, and Dongjin Lee. "Robust marker tracking algorithm for precise UAV vision-based autonomous landing". In: *2015 15th International Conference on Control, Automation and Systems (ICCAS)* (2015), pp. 443–446.

[17] David Jurado-Rodríguez et al. "Planar fiducial markers: a comparative study". In: *Virtual Reality* (2023), pp. 1–17. URL: https://api.semanticscholar.org/CorpusID:257138432.

[18] Michail Kalaitzakis et al. "Fiducial Markers for Pose Estimation". In: *Journal of Intelligent & Robotic Systems* 101 (2021).

[19] Rudolf E. Kálmán and Richard S. Bucy. "New Results in Linear Filtering and Prediction Theory". In: *Journal of Basic Engineering* 83 (1961), pp. 95–108. URL: https://api.semanticscholar.org/CorpusID:8141345.

[20] Navid Kayhani et al. "Improved Tag-based Indoor Localization of UAVs Using Extended Kalman Filter". In: *Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC)* (2019).

[21] Zhou Li et al. "UAV Autonomous Landing Technology Based on AprilTags Vision Positioning Algorithm". In: *2019 Chinese Control Conference (CCC)* (2019), pp. 8148–8153.

[22] Xiao Liang et al. "Moving target tracking method for unmanned aerial vehicle/unmanned ground vehicle heterogeneous system based on AprilTags". In: *Measurement and Control* 53 (2020), pp. 427–440.

[23] Manni Liu et al. "Indoor acoustic localization: a survey". In: *Human-centric Computing and Information Sciences* 10 (2020), pp. 1–24.

[24] Alberto López-Cerón and José María Cañas. "Accuracy analysis of marker-based 3D visual localization". In: *Actas de las XXXVII Jornadas de Automática 7, 8 y 9 de septiembre de 2016, Madrid* (2022). URL: https://api.semanticscholar.org/CorpusID:195254116.

[25] F. Landis Markley et al. "Averaging Quaternions". In: 2007. URL: https://api.semanticscholar.org/CorpusID:202637197.

[26] Mohammad Nahangi et al. "Automated Localization of UAVs in GPS-Denied Indoor Construction Environments Using Fiducial Markers". In: *Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC)* (2018).

[27] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". In: *2011 IEEE International Conference on Robotics and Automation* (2011), pp. 3400–3407.

[28] Shiye Pan and Xinmei Wang. "A Survey on Perspective-n-Point Problem". In: *2021 40th Chinese Control Conference (CCC)* (2021), pp. 2396–2401. URL: https://api.semanticscholar.org/CorpusID:238416916.

[29] Luca Pescante, Michieletto Giulia, and Michieletto Stefano. *Studio e implementazione di algoritmi di localizzazione per UAVs basati su tecniche di Visual Inertial Odometry*. 2022. URL: https://hdl.handle.net/20.500.12608/36606.

[30] A. B. M. Mohaimenur Rahman, Ting Li, and Yu Wang. "Recent Advances in Indoor Localization via Visible Lights: A Survey". In: *Sensors (Basel, Switzerland)* 20 (2020).

[31] Artur Sagitov et al. "Effects of rotation and systematic occlusion on fiducial marker recognition". In: 2017. URL: https://api.semanticscholar.org/CorpusID:54689673.

[32] Martino Segattini, Giulia Michieletto, and Michieletto Stefano. *HR01 PROJECT - progettazione di un esarotore & implementazione di tecniche di sensing e perception*. 2021. URL: https://hdl.handle.net/20.500.12608/6604.

[33] Guanzheng Wang, Zhihong Liu, and Xiangke Wang. "UAV Autonomous Landing using Visual Servo Control based on Aerostack". In: *Proceedings of the 3rd International Conference on Computer Science and Application Engineering* (2019).

[34] Zhanglong Wang, Haoping She, and Weiyong Si. "Autonomous landing of multi-rotors UAV with monocular gimbaled camera on moving vehicle". In: *2017 13th IEEE International Conference on Control & Automation (ICCA)* (2017), pp. 408–412.

[35] Beiya Yang and Erfu Yang. "A Survey on Radio Frequency based Precise Localisation Technology for UAV in GPS-denied Environment". In: *Journal of Intelligent & Robotic Systems* 103 (2021).

[36] Ali Yassin et al. "Recent Advances in Indoor Localization: A Survey on Theoretical Approaches and Applications". In: *IEEE Communications Surveys & Tutorials* 19 (2017), pp. 1327–1346.

[37] Faheem Zafari, Athanasios Gkelias, and Kin Kwong Leung. "A Survey of Indoor Localization Systems and Technologies". In: *IEEE Communications Surveys & Tutorials* 21 (2017), pp. 2568–2599.

[38] Boxin Zhao et al. "Relative Localization for UAVs Based on April-Tags". In: *2020 Chinese Control And Decision Conference (CCDC)* (2020), pp. 444–449.

[39] Guo Zhenglong, Fu Qiang, and Quan Quan. "Pose Estimation for Multicopters Based on Monocular Vision and AprilTag". In: *2018 37th Chinese Control Conference (CCC)* (2018), pp. 4717–4722.

Acknowledgments

My gratitude goes out to my professors, especially Prof. Cenedese and Prof. Michieletto, who allowed me to be part of this fascinating project and guided me in its development. The most special thanks go to Dr. Bertoni for having assisted and advised me throughout the project in all its practical aspects.