Alma Mater Studiorum - Università di Bologna

DOTTORATO DI RICERCA IN

AUTOMATICA E RICERCA OPERATIVA
Ciclo 28

Settore Concorsuale di afferenza: 09 / G1 AUTOMATICA
Settore Scientifico disciplinare: ING-INF / 04 - AUTOMATICA

NONLINEAR STATE ESTIMATION AND CONTROL OF AUTONOMOUS AERIAL ROBOTS: DESIGN AND EXPERIMENTAL VALIDATION OF SMARTPHONE BASED QUADROTOR

Presentata da: Mohammad Radi Mohammad Hayajneh

Coordinatore Dottorato

Prof. Daniele Vigo

Relatore

Prof. Lorenzo Marconi

Esame finale anno 2016

Declaration of Authorship

I'm, Mohammad Radi Mohammad Hayajneh, declare that this thesis titled, (Non-linear State Estimation and Control of Autonomous Aerial Robots: Design and Experimental Validation of Smartphone Based Quadrotor) and the work presented in it are my own. I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University.
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
3. Where I have consulted the published work of others, this is always clearly attributed.
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
5. I have acknowledged all main sources of help.
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Abstract

This work introduces developments of Guidance, Navigation and Control (GNC) systems with application to autonomous Unmanned Aerial Vehicle (UAV). Precisely, this work shows the development of navigation system based on nonlinear complementary filters for position, velocity and attitude estimation using low-cost Micro-Electro-Mechanical System (MEMS) sensors. The proposed filtering method provides attitude estimates in quaternion representations and position and velocity estimates in North-East-Down (NED) frame coordinates by fusing measurements from Inertial Measurement Unit (IMU), position and velocity measurements from Global Positioning System (GPS), and pressure altitude measurements from a barometric altimeter. Least Square Method (LSM) was used in gains tuning to find the best-fitting of the estimated states with precise measurements obtained by a vision based motion capture system. A complete navigation system was produced by integrating both the attitude and the position filters. The integration of the filtering approach based primarily on the ease of design and computational load. Furthermore, the structure of the filtering design allow for straightforward implementation without a need of high performance signal processing. Moreover, the proposed filters in this work can be tuned totally independent of each other. This work also introduces a nonlinear flight controller for stability and trajectory tracking that is practical for real-time implementation. This controller is also demonstrated the ability of a supervisory controller to provide effective waypoint navigation capabilities in autonomous UAV. The implementation of the guidance, navigation, and control algorithms together were adopted in the design of a novel smartphone based autopilot for particular quadrotor aerial platforms. These algorithms are considered the core of the flight software that allows to successfully complete the assigned mission autonomously. The performances of the proposed work are then evaluated by means of several flight tests. The work also includes a design of advanced platform of a quadrotor for the purpose of performing navigation and guidance systems in Search And Rescue (SAR) missions. Primarily, the performance of the navigation and guidance systems were tested in laboratory by simulating position measurements from GPS. This activity facilitates moving by the experiments from indoor to outdoor. This task needed a design of virtual GPS system which depends mainly on converting the position data derived by a vision based motion capture system to geodetic information. The derived geodetic information then were formatted in a specific binary structures using UBX protocol and sent to the autopilot of the quadrotor. Robot Operating System (ROS) was used to implement this activity together with complex

algorithms for the navigation and guidance on Linux computer mounted onboard and to establish the communication bridge with Ground Control Station (GCS).

Acknowledgements

My sincere gratitude goes first to my advisor Prof. Lorenzo Marconi for giving me the opportunity to be one of his research team and to be involved in the research activities carried out in CASY Lab. Being one of his student makes me always proud.

My thanks extends also to Dr. Roberto Naldi who gave access to the laboratory and research facilities. And to Marco Melega (The good person) for his continous supports and his early insights which launched the greater part of this dissertation. Without their precious support it would not be possible to conduct this research.

I thank my friends and fellow colleagues in the Lab for all the experiences we have together and for all the happy times we had during the three years. I also thanks Andrea Sala for his technical support and giving me the opportunity to complete the experimental part with him in his company.

I'm indebted to my family. Words cannot express how grateful I am to my father Radi, my mother Farideh, and to my brothers and sister for providing all means of support throughout my life. I would like express appreciation to my beloved wife Tasneem who spent sleepless nights with me and was always my support in the moments when there was no one to answer my queries. Finally, I hope this work be a source of pride for my daughter Sidra. My life began when you were born, my little girl.

University of Bologna
PhD in Automatic Control and Operational Research

Nonlinear State Estimation and Control of Autonomous Aerial Robots: Design and Experimental Validation of Smartphone Based Quadrotor

Mohammd Radi M. Hayajneh

Advisor: Prof. Lorenzo Marconi

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, number of researchers worked on the field of UAV development, particularly multirotor aerial vehicles, for many civilian applications. Quadrotor became a standard platform in many research projects because it has several advantages among UAVs that qualify it as a good platform to complete challenging missions autonomously. It has vertical takeoff and landing ability and high maneuverability as well as it is characterised by a simpler structure with respect to other flying machines [7]. Researchers continually put their effort on developing quadrotors (or quadcopters) to increase its abilities by making advances in communication, environment exploration, and maneuverability. Nowadays, quadrotors are widely used in university research in different fields, including flight control theory [30] [12], [33] navigation [7], real time systems [38] [35], and robotics [17]. In addition to reference [22] summarises the main quadrotor applications proposed currently and the different control techniques studied in the literature.

The development of more advanced aerial vehicles has encouraged many people to use these machines in different civilian applications that benefit human life [43], such as search and rescue operations, environment monitoring and security surveillance [49] [58] [54] [40]. For this reason researchers try to find appropriate Guidance, Navigation and Control (GNC) systems for these aerial vehicles to accomplish missions autonomously. Designing a suitable GNC system is an essential in aerial robot systems that perform tasks autonomously. Robot navigation depends mainly on the robot localization and the path planning. both localization and planning requires good estimation of the robot's current position and a position of a target location [26]. The development on MEMS gives the opportunity to equip UAVs with various sensors to accurately measure the state variables.

Vehicle state estimation is a crucial step in the development of GNC systems for autonomous aerial platforms and it is achieved by fusing information from many sources [15]. In outdoor applications, a popular approach is followed for estimating a vehicle's state based on fusing long term accuracy information (*e.g.* GPS) with sensors characterized by high output rate, high dynamics, and reliability (*e.g.* IMU) [59], [32]. Usually the raw signals from different sensors need processing to rebiuld smoothed and accurate state estimates [53] and to identify online bias terms [37]. The aerial vehicles in outdoor operations are subjected to an important accelerations for accomplishing missions faster and making smooth transitions between waypoints. These accelerations can not be avoided in estimation process of the vehicle's state.

According the Standard Positioning Service (SPS), civilian users can not achieve very precise positions and velocities using one standalone GPS receiver. Thus the integrated solution of GPS with IMUs is unavoidable to provide better positioning during the case of GPS drop-outs [59] [29]. Kalman filtering techniques have been widely applied on the aerial systems for state estimation. Many people have used Extended Kalman Filters (EKFs) [25] [52], [46], [32], [14]. Although EKF is used extensively in navigation problems it requires high efforts in computing Jacobian matrices [53]. To overcome the computational difficulties in EKF, the unscented transformation method is used in Unscented Kalman Filter (UKF). The UKFs provide more accurate results than EKFs and the results show better performance for a longer time under GPS outage conditions [61], [57]. A particular interest has been focused on the estimation of the noise variance-covariance matrices of the state and the measurement models which lead to what is known by the adaptive kalman filter [6]. Filtering approaches using Kalman techniques require online computation of the covariances and the gains, which have proved difficulty to apply to small UAVs with low-cost and high-noisy sensors [15], [47]. Moreover it may consume most of the few computational resources found in low-power processing hardware used in small and low-cost UASs [15].

Another approach followed in improving position accuracy can be achieved effectively by cancelling out the common errors and biases by using multi-antenna GPS receivers more can be effectively cancelled out technique becomes the most common in robotics field to take advantage improve the positioning accuracy by using multi-antenna GPS receiver [23], [6], [14]. By increasing number of GPS receivers in miniature aircraft vehicles will increase the power consumption and the payload in aircraft.

Sensor fusion based on complementary filters have been used widely in different robotic problems. The simple structure of the complementary filters allow for handy and direct implementation without the need of high performance signal processing in limited computational resources. This work introduces a design of filtering approach to estimate the vehicles velocities, positions, orientation based on nonlinear complementary filters, related to the works in [37] [21] [24] . In this approach, all the states are included in the estimation process while updates are generated by GPS, magnetometer, and pressure measurements. The accelerometer and gyroscope biases have been estimated and compensated online to give better accuracy in long-term. The attitude and positon complementary filters are integrated in a complete navigation system. The choice of this integration between the two filters is based primarily on the ease of design and computational load. Furthermore, the structure of the filtering design allow for straightforward implementation without a need of high performance signal processing. The proposed filtering approach is extensible and allow for integrating new measurements resource such as camera. The filters in this work also can be tuned totally independent of each other and the adopted gains are computed offline using an auxiliary design system to achieve robustness in most flight conditions. In this structure, the attitude is always provided and the measurement of altitude is always available even in GPS drop. The navigation system expected to produce accurate attitude and position estimation, which will be crucial to the stability of the aerial quadrotor and support the implementation of reliable trajectory tracking control strategies.

Most recent advanced works choose an approach aiming to obtain autonomous navigation capabilities using monocular camera [4], or using Red, Green, Blue and Depth (RGB-D) camera [56]. Others equipped the UAV with several cameras to capture more critical views which ensure the long term pose tracking and to build a map that includes more details about the environment [62]. Other groups use laser scanners on UAV to navigate in cluttered environments such as forests [16]. A combination between the two approaches above, i.e., the use of cameras and laser scanners, is demonstrated in [8, 55, 51] to compensate the drawbacks of using only one of them. In addition to the GPS- based navigation approach, the vision based and the laser-based approaches increase the accuracy and the robustness of the UAV. These examples proposed so far highlights a typical limitation for quadrotors, that is the low payload, which limits the set of sensors that can be placed onboard.

Nowadays the big challenge in the improvement of such aerial robots is not their implementation, but rather developing compact structure of hardware and software qualified for making them autonomously perform tasks as well as increase their reliability and ability to carry payload. A second significant contribution provided in this work is a developed quad-rotor using custom-built structure and avionics with off-the-shelf motors and batteries, to be a highly reliable experimental platform. The flight dynamics, navigation system and controller are built inside smartphone to stabilize flights based on information provided by different sensors onboard. This work shows the design of a novel smartphone-based autopilot for quadrotor aerial platforms. The proposed solution consists of two different layers, a high-level human-robot communication layer and a low-level navigation and control layer.

Android is the most widely used operating system for smartphones in the world. It is open source and released by *Google*. It allows therefore people to edit the source code and it gives the benefit of developing quite sophisticated *Android* applications. In addition, *Android* provides a Hardware Abstraction Layer (HAL) which gives developers the tools to create the interface between the *Android* platform stack and the hardware. Therefore, people started using *Android* smartphones on quadrotors to implement smaller and faster aerial platforms [36] [7]. In particular, Bjalemark and Bergkvist [10] took the advantages of the sensors and computational power of an *Android* smartphone to implement a quadrotor based on Proportional-Integral-Derivative (PID) controllers. Moreover, in Leichtfried *et al.* [31] the smartphone is used on a quadrotor as onboard computer to execute an algorithmic approach for mapping and localization in autonomous flight operations. in addition to the *Quadroid* project [48] already accomplished a flight test of a similar concept, *i.e.* an *Android* smartphone acting as on-board computer of the quadrotor and remotely controlled by another smartphone.

UAVs now can fly autonomously based on pre-programmed flight plans based on the objectives of the mission, and the flight plan is defined in a sequence of waypoints [50]. The quadrotor concept here described is used to develop the preliminary implementation of highly automated drone with the capability to share information and receive commands from multiple human operators. Such characteristics are very powerful capability in different operating scenarios, in which low-altitude drones, like quadrotors, can be effectively employed to speed-up operations, *i.e.* search & rescue and security & monitoring applications. This operating mode allows in fact the drone to act as a smart agent able to promptly communicate and receive commands from all the agents involved in the operations, like one of the human agents. On the other hand, the autonomous flight capabilities allow the drone itself to have a clearer and wider view of the search area and to avoid the ground obstacles impeding to reach another search area. In order to implement this control capability, it was decided to avoid the development of a dedicated web-based control application and to implement for the drone the capability to connect to a *Twitter* account. The human agents commands are published on this account and the drone itself can upload the acquired data. A preliminary demonstration of this concept is given in this work considering a single human agent communicating with the drone *Twitter* account in order to control its flight.

1.1 Objectives and organisation of the thesis

This work includes a new design of quadrotor concept having the advantage to provide several effective solutions in terms of development and widening of operating capabilities. The core of its implementation is a *Google Nexus 5* smartphone acting as a flight control system. The quadrotor model adopted in this thesis and the nonlinear attitude and position controller are introduced in chapter 2.

The flight dynamics of the quadrotor and the state estimation are based on several sensors onboard (e.g. IMU, GPS, Barometer) for in-door and out-door navigation. The model of the sensory system as well as nonlinear complementary filters for attitude and position estimations provided in chapter 3. This chapter introduces also sensor calibrations and tuning process for filters gains and parameters.

The overall design of the hardware and software structures of the flight platform and the design of the quadrotor are presented in chapter 4. This chapter includes also a real-time implementations of the Guidance, Navigation and Control (GNC) systems and gives a description about the supervisory control algorithm adopted in this work that receive operator's commands and translate them into reference set points for position control law.

chapter 5 presents advance guidance system has been implemented on complex and developed quadrotor platform to increase its capability in performing challenging tasks. This chapter provides also a design of virtual navigation system to simulate the GPS information in the laboratory. The implementation of this structure based on ROS running in a computer mounted on board.

To test and to evaluate this work, several practical experiments have been conducted on the quadrotor platform in indoor and outdoor flight tests. The results of the performed experiments are summarized in chapter 6.

Finally, the conclusion and recommendations for future works are provided in chapter 7.

1.2 Mathematical definitions

A rotation vector, quaternion, Euler angles, and the Direction Cosine Matrix (DCM) are commonly used in the inertial navigation systems of the aerial vehicles. This section introduces the definitions and notations used in this document

1.2.1 Vectors and rotation matix

For two vectors $a, b \in \mathbb{R}^3$, the vector cross product $\times$ between them can be computed as

$$a \times b = S(a)b \tag{1.1}$$

such that for any vector $v = (v_{(1)}, v_{(2)}, v_{(3)})^T \in \mathbb{R}^3$, the corresponding skew-symmetric matrix $S(v) \in \mathbb{R}^{3x3}$ is defined as follows

$$S(v) = \begin{bmatrix} 0 & -v_{(3)} & v_{(2)} \\ v_{(3)} & 0 & -v_{(1)} \\ -v_{(2)} & v_{(1)} & 0 \end{bmatrix} \tag{1.2}$$

And the invers operator $\wedge$ is defined for extracting the vector such that $S(v)^\wedge = v$. Any vector v_B in body coordinate system can be transformed to the inertial coordinate system by using the orthogonal rotation matrix as follows:

$$v_I = Rv_B \tag{1.3}$$

where $R \in \mathbb{R}^{3\times3}$ is the rotation matrix. The subscripts I and B symbols the vector in the inertial and body coordinate systems respectively. In contrast any vector in inertial coordinate system can be represented in body coordinate system using the inverse of the rotation matrix. The inverse of an orthogonal rotation matrix can be obtained by the transpose operation.

$$v_B = R^{-1}v_I = R^T v_I \tag{1.4}$$

The Frobenius norm is defined for a matrix (e.g. $Q \in \mathbb{R}^{m\times n}$) as the square root of the matrix trace of QQ^T, where Q^T is the conjugate transpose.It is also equal to the square root of the sum of the absolute squares of its elements:

$$|Q|_F = \sqrt{Tr(QQ^T)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_ij)^2} \tag{1.5}$$

where Q^T is the conjugate transpose

For a vector-valued random variable $\eta = [v_1, v_2, ..., v_n]^T$ has a Gaussian distribution with mean $\mu \in R^n$ and covariance matrix $\Theta \in R^{n\times n}$ is given by its probability density function as follow [18]

$$\eta \sim \mathcal{N}(\mu, \Theta) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Theta|^{\frac{1}{2}}} exp\left(-\frac{1}{2}(v-\mu)^T\Theta^{-1}(v-\mu)\right) \tag{1.6}$$

An alternative way to characterize the covariance matrix

$$\Theta = E[(v-\mu)(v-\mu)^T] = E[vv^T] - \mu\mu^T \tag{1.7}$$

Specifically, for a set of k observations, measuring 3 variables, the variance-covariance matrix Θ is given by

$$\Theta = \frac{1}{k-1}\sum_{i=1}^{k}(v_i-\mu)(v_i-\mu)^T \tag{1.8}$$

1.2.2 Quaternion Math

A quaternion $q \in S^3$ is a four-dimensional vector consists of a scalar $r \in \mathbb{R}$ and a vector $v \in \mathbb{R}^3$. Quaternion can be represented in many ways, the following two equations are the most popular approaches are followed

$$q = \begin{bmatrix} r & v \end{bmatrix}^T = [q_0\ q_1\ q_2\ q_3]^T \tag{1.9}$$

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{1.10}$$

According to this definition, the inverse (conjugate) of the quaternion can be expressed as:

$$q^{-1} = \begin{bmatrix} r \\ -v \end{bmatrix} \tag{1.11}$$

All quaternions are assumed to be of unitary length and thus are called unit quaternions. The following property holds for the norm $|q|$:

$$|q| = \sqrt{qq^{-1}} = \sqrt{q^{-1}q} = 1 \tag{1.12}$$

The standard quaternion product $\otimes$ between two quaternions $q_1 \in S^3$ and $q_2 \in S^3$ can then be calculated as:

$$q_1 \otimes q_2 = \begin{bmatrix} r_1 & -v_1{}^T \\ v_1 & r_1 I_3 + S(v_1) \end{bmatrix} \begin{bmatrix} r_2 \\ v_2 \end{bmatrix} \tag{1.13}$$

where $I_3 \in \mathbb{R}^{3\times 3}$ is the identity matrix. The multiplication between two unit quaternion can be given in more detail as follow

$$p \otimes q = \begin{bmatrix} p_0q_0 - p_1q_1 - p_2q_2 - p_3q_3 \\ p_0q_1 + p_1q_0 + p_2q_3 - p_3q_2 \\ p_0q_2 - p_1q_3 + p_2q_0 + p_3q_1 \\ p_0q_3 + p_1q_2 - p_2q_1 + p_3q_0 \end{bmatrix} \tag{1.14}$$

Unit quaternion can be used as a rotation operator. This rotates the vector $b \in \mathbb{R}^3$ from the Inertial frame to the body frame represented by q

$$b_{body} = q \otimes \begin{bmatrix} 0 \\ b_{inertial} \end{bmatrix} \otimes q^{-1} \tag{1.15}$$

The standard rotation matrix $R \in \mathbb{R}^{3\times3}$ can finally be computed from the quaternion using the Rodrigues formula as follows:

$$R(q) = (r^2 - v^T v)I_3 + 2vv^T - 2rS(v) \tag{1.16}$$

1.3 Reference frames and transformations

Figure 1.1 shows the main coordinates systems used in most aerial applications. The inertial coordinate reference or inertial frame is an Earth-fixed unmoving reference where the x-axis points north, the y-axis points east, and the z-axis points. Through the entire document we will call this a North-East-Down (NED) reference frame. Note that because the z-axis points down, altitude above ground is actually a negative quantity. On aircraft the body frame is the coordinate system that is aligned with the sensors. Typically x-axis points out the nose, the y-axis aligned with the right side of the body, and the z-axis points out the bottom.

1.3.1 Euler angles

Euler angles are used to represent the orientation of a rigid body or frame of reference in 3-dimensional Euclidean space. Typically they describe the sequence of three rotations about the axes of a coordinate system. They are typically denoted as ϕ, θ, and ψ to represent a rotation around x, y, and z axes respectively.

1.3.2 Rotation matrix and Euler angles

According to Euler's rotation theorem, any rotation may be described using three angles. There are several conventions for Euler angles, depending on the axes about which the rotations are carried out. The most common definition is the rotation given by Euler angles sequentially $(\phi\ \theta\ \psi)$, If the rotations are written in terms of rotation matrices R_ϕ, R_θ, and R_ψ then the complete rotation matrix for moving from the inertial frame to the body frame is given by a general rotation R can be written as

$$R_I^B = R_\psi R_\theta R_\phi \tag{1.17}$$

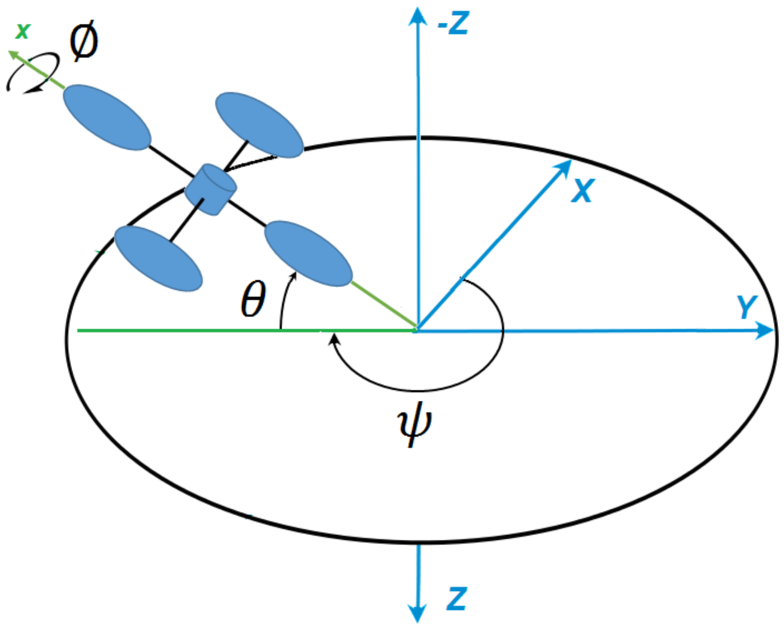


Figure 1.1: Frames of reference

where

$$R_\psi = \begin{bmatrix} cos\psi & sin\psi & 0 \\ -sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R_\theta = \begin{bmatrix} cos\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{bmatrix} \quad R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{bmatrix} \tag{1.18}$$

The rotation matrix for moving from the body frame to the inertial frame is given by opposite rotations as following

$$R_B^I = R_\psi R_\theta R_\phi \tag{1.19}$$

1.3.3 Quaternion and Euler angles

Presenting rotations as unit quaternions instead of matrices help to decrease the computational time in low-cost embedded systems. In addition, quaternions are considered numerically more stable and its interpolation is more straightforward. Furthermore quaternions don't suffer from gimbal lock unlike Euler angles. The transformation from Euler angles to quaternion can be performed by employing the following equation.

$$q = \begin{bmatrix} cos(\psi/2)cos(\theta/2)cos(\psi/2) + sin(\phi/2)sin(\theta/2)sin(\psi/2) \\ sin(\phi/2)cos(\theta/2)cos(\psi/2)cos(\phi/2)sin(\theta/2)sin(\psi/2) \\ cos(\phi/2)sin(\theta/2)cos(\psi/2) + sin(\phi/2)cos(\theta/2)sin(\psi/2) \\ cos(\phi/2)cos(\theta/2)sin(\psi/2)sin(\phi/2)sin(\theta/2)cos(\psi/2) \end{bmatrix} \tag{1.20}$$

In contrast, the conversion from quaternion to Euler angle can be achieved by utilizing the following equation respectively. This property is very useful in case that the aim is to represent an orientation in angles, while executing the overall dynamics of the system in a quaternion form.

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} atan2(2(q_0q_1 + q_2q_3), q_0^2 - q_1^2 - q_2^2 + q_3^2 \\ asin(2(q_0q_2q_3q_1)) \\ atan2(2(q_0q_3 + q_1q_2), q_0^2 + q_1^2 - q_2^2 - q_3^2 \end{bmatrix} \tag{1.21}$$

1.3.4 Geodetic to North-East-Down (NED)

The Global Positioning System (GPS) provide absolute location of a receiver by defining its latitude and longitude as well as the altitude. Most robotics application use the relative positions in North-East-Down (NED) frame. The small changes in the North and East positions can be calculated from small changes in Latitude and Longitude according to the following formula

$$\begin{cases} dN = \frac{d\mathcal{U}}{atan\left(\frac{1}{R_M}\right)} \\ \\ dE = \frac{d\mathcal{L}}{atan\left(\frac{1}{R_N cos\mu_0}\right)} \end{cases} \tag{1.22}$$

where dN and dE are the small change in North and East respectively. $d\mathcal{U} = \mathcal{U} - \mathcal{U}_0$ $d\mathcal{L} = \mathcal{L} - \mathcal{L}_0$ are the small changes in Latitude and Longitude respecively. R_N and R_M are defined by the following relationships:

$$\begin{cases} R_N = \frac{R}{\sqrt{1-(2f-f^2)sin^2\mathcal{U}_0}} \\ \\ R_M = R_N\frac{1-(2f-f^2)}{\sqrt{1-(2f-f^2)sin^2\mathcal{U}_0}} \end{cases} \tag{1.23}$$

where R and f are the equatorial radius and the flattining factor of the earth

1.3.5 Earth's magnetic field

Typically, the heading of an aircraft is related to the North direction and can be provided by a compass onboard that measure the direction of magnetic North. The magnetic field vector (the intensity) is differ from location to another. It is defined by the declination angle from true north, and inclination angle that measured with the horizontal. The Earth's magnetic field can be represented by a three-dimensional vector in NED frame. by a three-dimensional vector in NED frame.

Chapter 2

Quadrotor UAV Modeling and Trajectory Controller Design

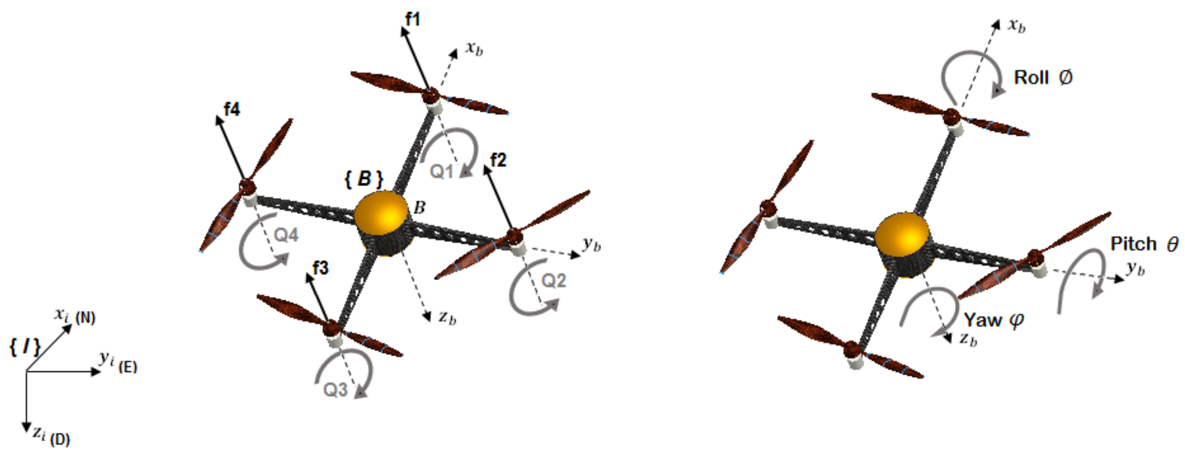
2.1 Introduction

Mathematical description of the flight quadrotor model is necessary for building an appropriate control to ensure stability, high maneuverability and tracking. Simple models for different configurations of quadrotor have been discussed extensively in literature (e.g. [11]. Considering quasi-stationary maneuvers of a quadrotor gives minimum number of inputs and states and thus simplifies the construction of the control law. If the aerodynamics, blade flapping, and the drag forces are taken in account then the computation of the thrust input will not be straightforward due to the fact that the drag forces are a function of the vehicle's orientation. A design of robust nonlinear controller can provide high flight accuracy against external disturbances and model uncertainties.

This chapter presents the adopted flight model of a quadrotor used in this study. The most basic modeling process assumes the quadrotor as a rigid airframe in the space with symmetrical counter-rotating propellers attached to it. The control of the quadrotor is achieved by varying the rotation speed of the propellers. Thus this chapter also introduces a nonlinear controller to achieve the vehicle's stability and to track a desired trajectory.

2.2 Quadrotor model

Figure 2.1 shows a quadrotor model. Four motors are mounted in a special configuration to allow the quadrotor to rotate around each axis to make rolling and pitching and to determine heading and the desired altitude. Two of the motors rotate clockwise and the other two rotate counterclockwise. Two frames of reference (see figure 2.1) are used to represent the dynamics and the kinematics of the quadrotor: an Inertial fixed-frame $I = \{O; x_i,\ y_i\ z_i\}$ mapped to North-East-Down (NED), and body fixed-frame $B = \{B; x_b,\ y_b\ z_b\}$ connected to the quadrotor's Center of Mass (CoM). The position of the vehicle is associated with CoM



(a) (b)

Figure 2.1: Quadrotor model: (a) Control forces and torques and (b) Quadrotor rotations

The dynamical and kinematical model of an aerial vehicle (rigid body) in the space can be described in term of quaternions as follows:

$$\begin{cases} \dot{p} = v \\ \dot{v} = g_{e3} - \frac{1}{m}R(q)T + F_a \\ \dot{q} = \frac{1}{2}q \otimes P(\omega) = R(q)S(\omega) \\ J\dot{\omega} = -\omega \times J\omega + \tau + \tau_a \end{cases} \tag{2.1}$$

where

- $p := [x\ y\ z]^T \in \mathbb{R}^3$ is the position of the vehicle's CoM represented in inertial reference frame I.
- $v := [\dot{x}\ \dot{y}\ \dot{z}]^T \in \mathbb{R}^3$ is the vehicle's velocity expressed inertial reference frame I.
- $\omega := [\omega_x,\ \omega_y,\ \omega_z]^T \in \mathbb{R}^3$ is the angular velocity of the frame B expressed in B.
- $q \in \mathbb{S}^4$ is unit quaternion represents rotation.
- $R \in SO(3)$ is rotation matrix representing the orientation of the body frame B relatively to the inertial frame I.
- $m \in \mathbb{R}$ and $J \in \mathbb{R}^{3\times 3}$ (i.e. $J = J^T > 0$) are the total mass and inertia matrix of the system respectively.
- $g_{e3} = [0\ 0\ g]^T \in \mathbb{R}^3$ denotes the gravitational vector with g is the local gravity acceleration.

- T is the total thrust and drage forces generated by motors expressed in body frame.
- τ is the vector of the reaction torques generated by motors expressed in body frame.
- F_a and τ_a are shorten to the vehicle drage force and the remaining aerodynamic torques respectively.

As shown in figure 2.1 the rotation of each rotor i generates a force $f_i = b_T \omega_i^2 e_3$ and a torque $Q_i = b_Q \omega_i^2 e_3$. The total thrust and torques generated by the rotors can be expressed with

$$T = \begin{bmatrix} 0 & 0 & b_T \sum_{i=1}^{n} \omega_{mi}^2 \end{bmatrix}^T \tag{2.2}$$

$$\tau = \sum_{i=1}^{n} (Q_i + d \times f_i) \tag{2.3}$$

where n is the number of rotors. ω_{mi} is the angular speed of the motor i. b_T and b_Q are positive aerodynamic constants depending on density of air and the radius, the pitch angle, and the shap of the blade. The control torques $\tau = [\tau_x \ \tau_y \ \tau_z]^T$ generated by the motors can be expressed in more detailed form as follows

$$\begin{cases} \tau_x = d(f_4 - f_2) \\ \tau_y = d(f_1 - f_3) \\ \tau_z = Q_1 - Q_2 + Q_3 - Q_4 \end{cases} \tag{2.4}$$

where d is the distance between the spinning motor axis and the center of mass of the quadrotor. The forces and torques applied to the quadrotor in (2.2) and (2.4) respectively can be arranged in a proper allocation matrix as follows:

$$\begin{bmatrix} T \\ \tau \end{bmatrix} = \begin{bmatrix} b_T & b_T & b_T & b_T \\ 0 & -b_T d & 0 & b_T d \\ b_T d & 0 & -b_T d & 0 \\ b_Q & -b_Q & b_Q & -b_Q \end{bmatrix} \begin{bmatrix} \omega_{m1}^2 \\ \omega_{m2}^2 \\ \omega_{m3}^2 \\ \omega_{m4}^2 \end{bmatrix} \tag{2.5}$$

2.3 Trajectory tracking controller

The model of a typical quadrotor has six DoF (three DoF for its translational motion in three directions, and three DoF for the angular motion), while only four of them can be actuated. Due to the underactuated nature in typical quadrotor, the translational motion is related to the nonlinear coupling term between the lifting thrust magnitude and the direction thrust. Thus the translational motion can be controlled by using intermediate control variable between them. The proposed non-linear control law inspired by [41] and [44] is adopted for quadrotor stabilization and trajectory tracking and illustrated in figure 2.2. Position and attitude control are coupled

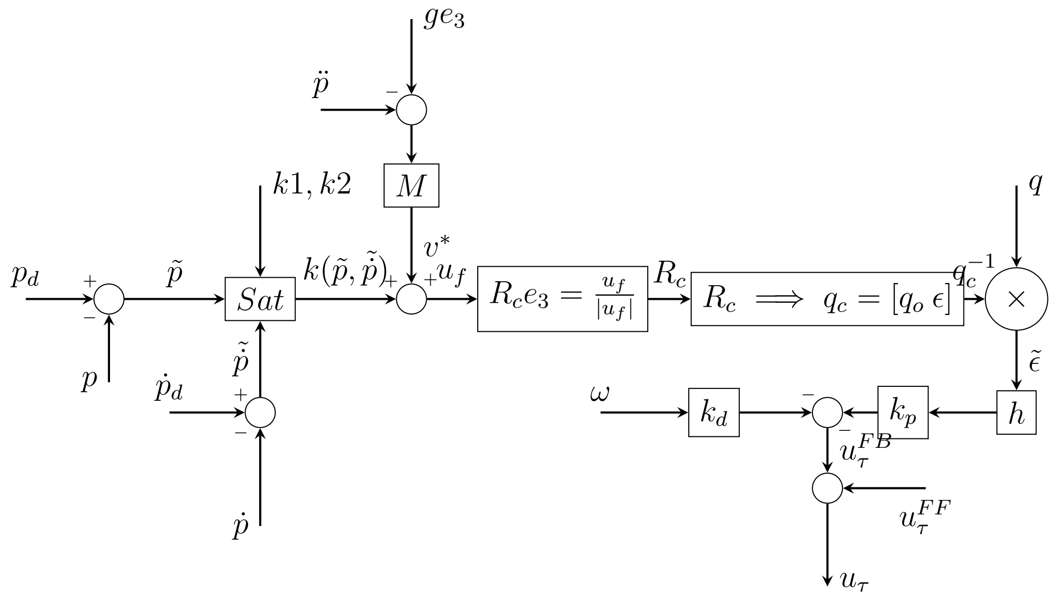


Figure 2.2: Trajectory Controller

tasks, despite the control strategy here considered allow to treat them as separate problems.

2.3.1 Model for control design

The quadrotor UAV model proposed in (2.1) can be rewritten as follow

$$\begin{cases} M\ddot{p} = -TR(q)e_3 + F_e \\ \dot{q} = \frac{1}{2} q \otimes P(\omega) \\ I\dot{\omega} = -\omega \times I\omega + u_\tau + u_e \end{cases} \tag{2.6}$$

where T is the thrust force and F_e refers to the sum of all remaining forces that acting on the vehicle. u_e indicates to the sum of all acting moments on the vehicle. Through the direct relationship of the vehicle thrust and torque with the squared rotation speed of the rotor (see (2.2) and (2.3)), one can see $T \in \mathbb{R}$ and $u_\tau \in \mathbb{R}^3$ as control inputs of the system (2.6). The modeled system of the quadrotor shown in (2.6) clearly confirms that the rotational dynamics are fully actuated. Thus, it is easy to achieve the convergence of the angular velocity to any bounded desired value. In order to implement the control system, a hierarchical control approach with attitude and position control as respectively inner and outer loops is inspired by the control approaches introduced in [41] and [44].

2.3.2 Position controller

The target of the position controller concentrates on tracking the reference trajectory $p_d \in \mathbb{R}^3$ and stabilizing the translational velocity $\dot{p}$ to a reference vector $\dot{p}_d \in \mathbb{R}^3$

using the thrust magnitude u_f and the thrust direction vector $Re_3 = R_c e_3$. By referring to (2.6) the translational error dynamics is:

$$\ddot{\tilde{p}} = -\frac{1}{M} T R(q) e_3 + \frac{F_e}{m} - \ddot{\tilde{p}}_d \tag{2.7}$$

where $\ddot{p}_d$ is the time derivative of a reference velocity vector $\dot{p}_d$. Then the position control law is based on the following vectored-thrust approach:

$$\begin{cases} u_f = V_c + \kappa(\tilde{p}, \dot{\tilde{p}}) \\ V_c = \frac{F_e}{M} - \ddot{p}_d \end{cases} \tag{2.8}$$

where $u_f \in \mathbb{R}$ is the force control input and V_c is a reference control force vector. $\tilde{p} = p - p_d$ and $\dot{\tilde{p}} = \dot{p} - \dot{p}_d$ are the error coordinates of position and velocity respectively and p_d denoting the reference desired position. Again M is the vehicle's mass and $e_3 = (0, 0, 1)^T$.

$\kappa(\tilde{p}, \dot{\tilde{p}})$ is a static state feedback law that insures the stability of the equilibrium such that $\kappa(0, 0) = 0$ and it is a function of a nested saturation law and given as:

$$\kappa(\tilde{p}, \dot{\tilde{p}}) = \begin{bmatrix} \lambda_{2xy}\sigma_{xy}\left(\frac{k_{2xy}}{\lambda_{2xy}}\left(\dot{\tilde{p}}_{x,y} + \lambda_{1xy}\sigma_{xy}\left(\frac{k_{1xy}}{\lambda_{1xy}}\tilde{p}_{x,y}\right)\right)\right) \\ \lambda_{2z}\sigma_{z}\left(\frac{k_{2z}}{\lambda_{2z}}\left(\dot{\tilde{p}}_{z} + \lambda_{1z}\sigma_{z}\left(\frac{k_{1z}}{\lambda_{1z}}\tilde{p}_{z}\right)\right)\right) \end{bmatrix} \tag{2.9}$$

where $\sigma_\Delta(\chi) = \chi min(1, \Delta/|\chi|)$ is a saturation function and the values of the positive parameters and gains $\lambda_1, \lambda_2, k_1, k_2 \in \mathbb{R}$ are chosen to make the function $\kappa(\tilde{p}, \dot{\tilde{p}})$ regularly smaller than the gravity constant g in order to insure the stability. This yields to the following outer loop control that defined by a direction unit vector $R_c e_3$ and thrust intensity T:

$$R_c e_3 = \nu_c = \frac{u_f}{|u_f|}, \quad T = M|u_f| \tag{2.10}$$

In practice, The external acceleration $\frac{F_e}{m}$ cannot be estimated accurately whatever the method adopted for that. Gravity, blade flapping and the aerodynamic drag are the most important forces acting on the system. However in most traditional control designs of small UAVs, the gravity is considered to be the only external force included in the system. According to inaccurate knowledge of $\frac{F_e}{m}$, an integral term is added into the proposed control law in order to further increase its robustness. The following bounded integrator of $\tilde{p}$ is adopted [44]

$$\begin{cases} \ddot{s}_{x,y} = -2k^s_{xy}\dot{s}_{x,y} - (k^s_{xy})^2(s_{x,y} - \sigma^s_{xy}(s_{x,y})) + \sigma^{ps}_{xy}(k^{ps}_{xy}\tilde{p}_{x,y}) \\ \ddot{s}_z = -2k^s_z\dot{s}_z - (k^s_z)^2(s_z - \sigma^s_z(s_z)) + \sigma^{ps}_z(k^{ps}_z\tilde{p}_z) \\ s(0) = 0, \quad \dot{s}(0) = 0 \end{cases} \tag{2.11}$$

where $\tilde{p}_s = \tilde{p} + s$ and $\dot{\tilde{p}}_s = \dot{\tilde{p}} + \dot{s}$ indicate to new tracking error variables with positive and sufficiently small numbers $k^s_{xy}, k^s_z, k^{ps}_{xy}$,$k^{ps}_z$ and classical saturation function $\sigma_\Delta(\chi) = \chi min(1, \Delta/|\chi|)$. The control law in (2.8) is concluded as:

$$\begin{cases} u_f = V_c + \kappa(\tilde{p}, \dot{\tilde{p}}) + \ddot{s} \\ V_c = \frac{F_e}{M} - \ddot{p}_d \end{cases} \tag{2.12}$$

The objective now is to stabilize the thrust orientation to the desired reference attitude R_c which specified in (2.10). The Attitude controller is introduced in more details in the following section.

2.3.3 Attitude controller

The attitude error coordinates can be defined as follow:

$$\tilde{q} = q_c^{-1} \otimes q, \quad \tilde{\omega} = \omega - \bar{\omega}_c \tag{2.13}$$

where q_c is the desired attitude in quaternion can be obtained from the thrust orientation vector ν_c, and $\bar{\omega}_c = R(\tilde{q})^T \omega_c$, with $\omega_c = ({R_c}^T \dot{R}_c)^\wedge$ the desired angular velocity associated with R_c . then the error attitude dynamics can be computed as following:

$$\begin{cases} \dot{\tilde{q}} = \frac{1}{2}\tilde{q} \otimes \begin{bmatrix} 0 \\ \tilde{\omega} \end{bmatrix} \\ I\dot{\tilde{\omega}} = \mathcal{S}(\tilde{\omega}, \bar{\omega}_c)\tilde{\omega} + S(I\bar{\omega}_c)\bar{\omega}_c - IR(\tilde{q})^T \dot{\tilde{\omega}}_c + \mu_\tau + \mu_e \end{cases} \tag{2.14}$$

with $\mathcal{S}(\tilde{\omega}, \bar{\omega}_c) := S(I\tilde{\omega}) + S(I\bar{\omega}_c) - S(\bar{\omega}_c)I - IS(\bar{\omega}_c)$, where $S(.)$ is the skew-symmetric matrix. The design of the proposed attitude control input is given by the following relationship:

$$\begin{cases} u_\tau = u_\tau^{FF} + u_\tau^{FB} \\ u_\tau^{FF} = JR(\tilde{q})^T \dot{\omega}_c - S(J\bar{\omega}_c)\bar{\omega}_c \\ u_\tau^{FB} = -k_p h \tilde{\epsilon} - k_d(\omega - \bar{\omega}_c) \end{cases} \tag{2.15}$$

where u_τ^{FF} and u_τ^{FB} denote a feedforward and feedback terms, respectively. And $\tilde{q} = q_c^{-1} \otimes q$ is error quaternion associated with its vector part $\tilde{\epsilon}$. The rotation matrix $R(\tilde{q})$ is obtained from quaternion using Rodrigues formula. k_p and k_d are positive gains and $h \in (-1, 1)$ is calculated through a specific hybrid system as explained in [41].

$$\begin{cases} \dot{h} = 0 & h\tilde{\eta} \geq -\delta \\ h^+ \in \overline{sgn}(\tilde{\eta}) & h\tilde{\eta} \leq -\delta \end{cases} \tag{2.16}$$

with $\delta \in (0, 1)$ and $\overline{sgn}$ is the set-valued function defined by the following relation

$$\overline{sgn}(s) = \begin{cases} sgn(s) & |s| > 0 \\ \{-1, 1\} & s = 0 \end{cases} \tag{2.17}$$

2.3.4 Linear mapping of control inputs

The resultant force u_f from (2.8) and the resultant torque u_τ from (2.15) can be computed as four thrusts generated from quadrotor's propellers. We can find a linear mapping from the square of the rotors angular velocity to the total thrust u_f and torque u_τ in a proper allocation matrix accordingly to mechanical parameters such as "plus" mounted quadrotor as follows:

$$\begin{bmatrix} u_f \\ u_\tau \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ 0 & 0 & -d & d \\ -d & d & 0 & 0 \\ -K_{tm} & -K_{tm} & K_{tm} & K_{tm} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} \tag{2.18}$$

where d is the distance between the spinning motor axis and the center of gravity. For each $i = 1, ..., 4$, T_i is the generated thrust from each rotor and it is a function of the angular speed of the motor. K_{tm} is proportional to the aerodynamic torque produced by the propeller.

Chapter 3

State Estimation and Navigation System Design

3.1 Introduction

To control the stability of an aerial vehicle, the attitude estimation are required especially in hard maneuvers. Also the estimates of the vehicle position, velocity, and heading angle are necessary for autonomous control capability. And these information can be derived from sensors onboard. But every individual sensor gives limited and less accurate information due to its uncertainties. Fusing several data from different sensors is important for estimating a vehicle state such that the resulting information has less uncertainty. This chapter introduces the sensory system model used in small UAVs and the preprocessing static calibration. Furthermore, this chapter describes an approach of sensor fusion based on nonlinear complementary filters that allow the autonomous flight of the quadrotor.

3.2 Sensory system model

In last decades, the developments of new low-cost and low-weight machines in robotics field have allowed and encouraged the development of MEMS sensors. These devices surpass many of the features that have prevented the use of inertial systems in many applications where cost, size and power consumption are playing important roles in design. Obviously the size reduction has resulted in sensors that have bias and measurement noise greater than other types. Therefore, this led to the development of techniques capable to compensate for such systematic errors in real time. Of these techniques are described in detail in next sections. This section introduces the sensors used in small UAV and their characteristics.

3.2.1 Inertial Measurement Unit (IMU)

A typical MEMS IMU used in UAV consists of gyroscopes, accelerometers, and also magnetometers to measures velocity and orientation. Theoretically, once true rotation rates and linear acceleration have been determined, vehicle attitude and position may be obtained by integration of the rotation rates of the gyros and double integration of the acceleration respectively. In practice, inertial sensors are influenced by bias, drift and noises in measurements. Direct integration on these measurements lead to unbounded increase in attitude and position errors with time, unless preprocessing and fusion algorithms are applied on the measurement data. The sensor signals of IMU measurments can be modeled as:

$$\begin{cases} \Omega_m = (1+S_\Omega)\Omega + b_\Omega + \eta_\Omega \\ a_m = (1+S_a)a + b_a + \eta_a \\ m_m = (1+S_m)m + b_m + \eta_m \end{cases} \tag{3.1}$$

where Ω_m, a_m and $m_m \in \mathbb{R}^3$ refer to the raw measurements of the gyroscope, accelerometer, and the magnetometer respectively. And Ω and a, and $m \in \mathbb{R}^3$ represent the true values of these sensors. b, S, $\eta \in \mathbb{R}^3$ are the bias error, the scaling error, and the random noise respectively on the signal of each sensor. η is assumed to be a zero-mean Gaussian white noise.

3.2.2 Global Positioning System (GPS)

Global Navigation Satellite Systems (GNSS) such as Global Positioning System (GPS) provides position and velocity information in an earth fixed coordinate system for navigation. The GPS receiver provides reliable location when four or more satellites are available and clear in the Line of the Sight (LOS). All receivers used in aerial robots have several source of errors such as atmospheric and multi path effects. GPS devices operates at low frequencies and samples can generally obtained at (1-5) Hz. In practice determining the velocity and position from stand-alone accelerometer can cause a drift in measurements even with good calibration process. So the GPS is integrated with accelerometer measurements to correct the drift as well as to provide continues estimate of the position when GPS lock is lost through the motion.

3.2.3 Barometer

Although, the GPS can provide altitude at a location its accuracy is unreliable and it is not available all the time. The estimation of elevation can be achieved by using a barometer. It is well known that the atmospheric pressure affected by the change in elevation and the air pressure can be measured by a barometer. The absolute height above the sea level can be computed from the atmospheric pressure as follow:

$$h = -\frac{T_0}{L}\frac{P^{\frac{RL}{gM}}}{P_0} - 1 \tag{3.2}$$

where h is the height above the sea level. P and P_0 are the air pressure at height h and sea level respectively. T_0 is the standard air temperature, M is the molar mass of air and R is the gas constant of air. g is the gravitational acceleration and L is the temperature lapse rate.

The digital pressure sensors suffers from noises and drifts because of temperature change. Calculating a vehicle attitude only from the pressure sensor is not accurate enough. Therefore the height information from the pressure sensor can be integrated with measurements from IMU to give better estimation for altitude.

3.2.4 Signals boundedness

In order to proof that the size of a signal with function $s(t)$ is bounded or finite, we need to know the characteristics of its size. For discrete-time signals, the indication for the size of a signal can be characterized in the energy, the power, and the amplitude of that signal *i.e.*

$$\begin{cases} E = \sum_{n} s^2[n] \\ P = \lim_{N \to \infty} \frac{1}{2N+1} \sum_{n=-N}^{N} s^2[n] \\ A = max|s[n]| \end{cases} \tag{3.3}$$

where E, P, and A represent the energy, the power, and the amplitude respectively. In real world, signals from sensors are bounded in energy because there is no infinite source of energy is available.

3.3 Static calibration

The bias errors and the scaling factors in inertial sensor can be eliminated with standard static calibration process such as six-position static test [5]. The accuracy of the static calibration depends on how well the MEMS inertial sensor is aligned with the vertical axes of the local level frame. This method requires static data collection of long period ($>$ 1 hour) at each axis of the sensor pointing alternately up and down. By aligning the IMU using the method in six positions and assuming zero mean random error, the bias and scale factors in (3.1) for each axis can then be calculated as follow:

$$\begin{cases} b_{axis} = \frac{\bar{M}^{up}+\bar{M}^{down}}{2} \\ S_{axis} = \frac{\bar{M}^{up}+\bar{M}^{down}}{2K} - 1 \end{cases} \quad axis = x, y, z \tag{3.4}$$

where $\bar{M}$ is average value of the measurement obtained by an accelerometer or a gyroscope by pointing each sensitive axis alternately up and down. K refers to the gravitational constant or the magnitude of the earth rotation rate at a given latitude.

The rotation of an aerial vehicle can generate centrifugal acceleration must be considered in the corresponding acceleration signal

The magnetometer suffers from hard- and soft- iron effects. The hard-iron error is constant regardless of orientation and can be compensated by simple static calibration. The soft-iron distortion is produced by material that influences a magnetic field. Unlike hard-iron effect the impact resulted by soft iron materials is dependent upon the orientation of the material relative to the sensor and the magnetic field. The calibration of the magnetometer based on [27]

The Hard-iron compensation is typically achieved by rotating the sensor through a minimum of 360^o in all directions then determining the scale factor and offsets as follows:

$$\begin{cases} b_{axis} = \frac{M^{Max}+M^{min}}{2} \\ S_{axis} = \frac{M^{Max}+M^{min}}{2K_m} - 1 \end{cases} \quad axis = x, y, z \tag{3.5}$$

where M^{max} and M^{min} are the maximum and the minimum measurements obtained from the magnetometer at each axis. K_m is the earth's magnetic field at a given location. The soft-iron effects cannot be compensated with a simple constant; instead, a more complicated procedure is needed.

3.4 Attitude and position complementary filters

This section introduces complementary filters for attitude and position estimation and the derivation of their stability and performance. The motivation by designing the filters in frequency domain is understanding the complementary characteristics of the inertial sensors in that domain.

3.4.1 Attitude estimation

The quaternion kinematics is described by

$$\dot{q} = \frac{1}{2} q \otimes P(\Omega) \tag{3.6}$$

The equivalante system in discrete-time of the system (3.6) is obtained by the zeroth- order integration method based on the Taylor series of $q(t_k + T)$ around the time $t = t_k$ as follow

$$q_{k+1} = q_k + T\dot{q}_k + \frac{T^2}{2!}\ddot{q}_k + \frac{T^3}{3!}\dddot{q}_k + ... \tag{3.7}$$

where T is the sampling time interval and the sub-index k abbreviate the time instant $t = kT$. By considering the angular rate Ω_k constant over the period $\Delta t = t_{k+1} - t_k$, we have $\dot{\Omega}_k = 0$ and then the equation above can be rewritten as

$$q_{k+1} = q_k \otimes P\left(1 + \frac{T}{2}\Omega_k + \frac{1}{2!}(\frac{T}{2}\Omega_k)^2 + \frac{1}{3!}(\frac{T}{2}\Omega_k)^3 + ...\right) \tag{3.8}$$

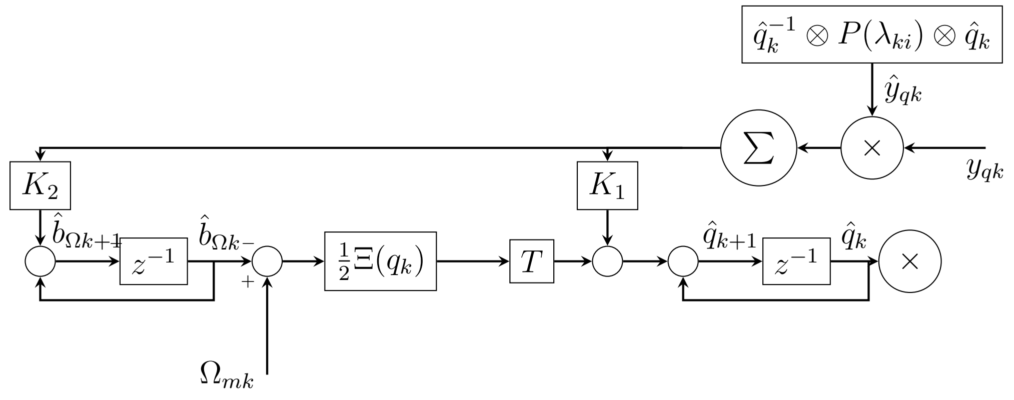


Figure 3.1: Attitude complementary filter

Taking in account a sufficient small sampling interval T, we can ignore the high order terms in previous equation for simplicity. Then the approximation of system (3.6) in discrete time can be given as follow:

$$q_{k+1} = q_k + \frac{T}{2} q_k \otimes P(\Omega_k) \tag{3.9}$$

In this work, the attitude can be derived from the angular velocity measured by a rate gyro. the proposed attitude filter estimates the vehicle's attitude expressed in quaternion and compensates for the rate gyro bias. Including the bias characteristic, the quaternion kinematic can be defined in state space form as follow

$$\begin{bmatrix} q_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} I_4 & -\frac{T}{2}\Xi(q_k) \\ 0 & I_3 \end{bmatrix} \begin{bmatrix} q_k \\ b_{\Omega k} \end{bmatrix} + \begin{bmatrix} \frac{T}{2}\Xi(q_k) \\ 0 \end{bmatrix} \Omega_m + \begin{bmatrix} -\frac{T}{2}\Xi(q_k) & 0 \\ 0 & I_3 \end{bmatrix} \begin{bmatrix} \eta_{\Omega k} \\ \eta_{bk} \end{bmatrix} \tag{3.10}$$

We consider the following nonlinear feedback system as the proposed attitude filter that includes the bias and the error update :

$$\begin{gathered} \begin{bmatrix} \hat{q}_{k+1} \\ \hat{b}_{k+1} \end{bmatrix} = \begin{bmatrix} I_4 & -\frac{T}{2}\Xi(q_k) \\ 0 & I_3 \end{bmatrix} \begin{bmatrix} \hat{q}_k \\ \hat{b}_{\Omega k} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Xi(q_k) K_{1q} \\ K_{2q} \end{bmatrix} \sum_{i=1}^{n} (y_{ik} \times \hat{y}_{ik}) \\ \hat{y}_{qk} = q_k^{-1} \otimes P(\lambda_k) \otimes q_k \end{gathered} \tag{3.11}$$

where K_{1q}, $K_{2q} \in R^3$ are positive constant gains vectors, and $y \times \hat{y}$ measures the inconsistency between the estimated vector and the measured one in the space of angular rate. $\hat{q}_k$ is the prediction of the attitude at time t. The vector $y \in \mathbb{R}^3$ is provided from a set of known inertial directions expressed in the body-fixed frame and measured by the available sensors (e.g. the Earth's gravitational and magnetic fields). $\hat{y}$ is the predicted vector expressed in the inertial frame and estimated in the body-fixed frame. Figure 3.1 shows the block diagram of the proposed attitude filter.

Practically, the set of vectors in NED frame λ_i includes the low-frequency estimation of λ_a of the acceleration reference in NED frame and the local direction λ_m of the Earth's magnetic field obtained by measurements at a local site. The contribution of the error of each vector is multiplied by a weight k_i selected according to the relative confidence in the inertial direction estimate

$$\sum_{i=1}^{2}(y_i \times \hat{y}_i) = k_a(y_1 \times \hat{y}_1) + k_m(y_1 \times \hat{y}_1) \tag{3.12}$$

In the proposed filter, the feedback gains K_1 and K_2 together with the weights k_a and k_m are identified with a tuning technique that satisfies asymptotically stable filter for any attitude trajectory parameterized by quaternion configurations. The adopted tuning technique is detailed later in subsection 3.5.2.

3.4.2 Position estimation

The position kinematics are given by

$$\begin{cases} \dot{p} = v \\ \dot{v} = R(q)a = \end{cases} \tag{3.13}$$

where v and p are the velocity and position in the inertial NED frame, R is the rotation matrix from body frame to inertial frame and can be derived from the equivalent quaternion. a is the obtained acceleration in body frame. The equivalent discrete-time form of the above system is given by

$$\begin{cases} p_{k+1} = p_k + Tv_k + \frac{T^2}{2}R(q_k)a_k \\ v_{k+1} = v_k + TR(q_k)a_k \end{cases} \tag{3.14}$$

In a rigid body, the measurements of the accelerometer are the difference between the inertial and gravitational accelerations. The real accelerometer output can be expressed as follow

$$\begin{cases} a_{mk} = (a_k + b_a) - g_k + \eta_{ak} \\ b_{ak+1} = b_{ak} + \eta_{bk} \end{cases} \tag{3.15}$$

where a_k and g_k are the inertial and the gravitational accelerations respectively expressed in body frame. $\eta_a \sim \mathcal{N}(0, \Theta_a)$ is zero-mean, Gaussian white noise. Then the kinematics system 3.14 can be given in state space form by

$$\begin{aligned} \begin{bmatrix} p_{k+1} \\ v_{k+1} \\ b_{k+1} \end{bmatrix} = \begin{bmatrix} I & TI & -\frac{T^2}{2}R(q_k) \\ 0 & I & -TR(q_k) \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} p_k \\ v_k \\ b_{ak} \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2}I \\ TI \\ 0 \end{bmatrix} (R(q_k)a_{mk} + g_{e3}) \\ + \begin{bmatrix} I & -\frac{T^2}{2}R(q_k) & 0 \\ 0 & -TR(q_k) & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \eta_{pk} \\ \eta_{ak} \\ \eta_{bk} \end{bmatrix} \end{aligned} \tag{3.16}$$

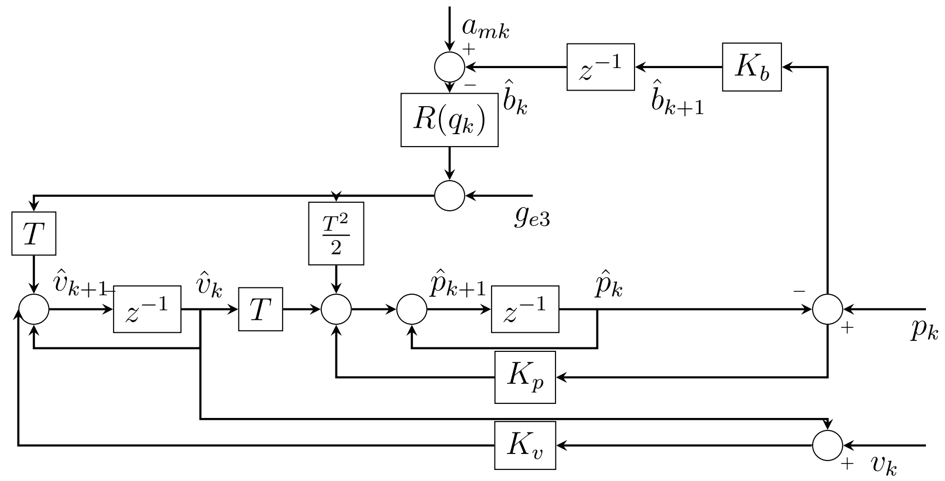


Figure 3.2: Position complementary filter

where g_{e3} is the gravitational acceleration constant in inertial frame, and $\eta_{pk} \sim \mathcal{N}(0, \Theta_p)$ is zero-mean, Gaussian white noise that resulted from small disturbances in the position. The proposed position observer kinematics, illustrated in figure 3.2 are given as follow

$$\begin{bmatrix} \hat{p}_{k+1} \\ \hat{v}_{k+1} \\ \hat{b}_{k+1} \end{bmatrix} = \begin{bmatrix} I & TI & -\frac{T^2}{2}R(q_k) \\ 0 & I & -TR(q_k) \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{p}_k \\ \hat{v}_k \\ \hat{b}_k \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2}I \\ TI \\ 0 \end{bmatrix} (R(q_k)a_{mk}+g_{e3}) + \begin{bmatrix} K_p(p_k - \hat{p}_k) \\ K_v(v_k - \hat{v}_k) \\ K_b(v_k - \hat{v}_k) \end{bmatrix} \tag{3.17}$$

where p_k is the position measured by the GPS reciever. K_p, K_v and K_b are constant gians. The covariance matrices Θ_p and Θ_a are tuned to shape the frequency response of the filter. In the design of the position filter, the covariance matrices and the accelerometer bias are compensated with pre-processing procedure.

Define the estimation errors $\tilde{p}_k = p_k - \hat{p}_k$ and $\tilde{v}_k = v_k - \hat{v}_k$. This results in the following error between the estimated and the real states.

$$\begin{bmatrix} \tilde{p}_{k+1} \\ \tilde{v}_{k+1} \\ \tilde{b}_{ak+1} \end{bmatrix} = \begin{bmatrix} I - K_p & TI & -\frac{T^2}{2}R(q_k) \\ 0 & I - K_v & -TR(q_k) \\ 0 & I - K_b & I \end{bmatrix} \begin{bmatrix} \tilde{p}_k \\ \tilde{v}_k \\ \tilde{b}_{ak} \end{bmatrix} + \begin{bmatrix} I_3 & -\frac{T^2}{2}R(q_k) & 0 \\ 0 & -TR(q_k) & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \eta_{p_k} \\ \eta_{a_mk} \\ \eta_{bk} \end{bmatrix} \tag{3.18}$$

3.4.3 Altitude estimation

Although, the GPS can provide altitude at a location its accuracy is unreliable and it is not available all the time. An accurate estimation of elevation can be acheived by fusing barometer measurement with IMU. The vertical velocity and position estimation method combines between barometer and accelerometer measurements is inroduced in this section. The method estimate the bias in the accelerometer to increase the accuracy in the estimation.

The position can be calculated by integrating the linear acceleration in inertial frame. Then the discrete-time vertical position kinematics are given by

$$\begin{cases} p_{zk+1} = p_{zk} + Tv_{zk} + \frac{T^2}{2}a_{zk} \\ v_{zk+1} = v_{zk} + Ta_{zk} \end{cases} \tag{3.19}$$

where p_z, v_z, and a_z are the position, velocity, and the acceleration in the inertial NED frame. The acceleration is measured in body frame and rotated by R to the inertial frame. By taking in acount the accelrometer bias v_o, then we can inroduce an observer in state spase form as follow:

$$\begin{bmatrix} \hat{p}_{zk+1} \\ \hat{v}_{zk+1} \\ \hat{v}_{ok+1} \end{bmatrix} = \begin{bmatrix} 1 & T & -\frac{T^2}{2}R(q_{zk}) \\ 0 & 1 & -TR(q_{zk}) \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{p}_{zk} \\ \hat{v}_{zk} \\ \hat{v}_{ok} \end{bmatrix} + \begin{bmatrix} \frac{T^2}{2} \\ T \\ 0 \end{bmatrix} (a_{m_z k} + g_{e3}) + \begin{bmatrix} K_{zp}(p_{zk} - \hat{p}_{zk}) \\ K_{zv}(v_{zk} - \hat{v}_{zk}) \\ K_{v_o}(v_{zk} - \hat{v}_{zk}) \end{bmatrix} \tag{3.20}$$

where p_{zk} is the position measured by the barometer. K_{zp}, K_{zv} and K_{zv_o} are constant gians.

As discussed in previous section, as long as the observer 5.3 apply the stability, then the estimated velocity $\hat{v}_n$ converge to the real one v_n as $t \to \infty$. Then Define the estimation errors $\tilde{p}_{zk} = p_{zk} - \hat{p}_{zk}$ and $\tilde{v}_{zk} = v_{zk} - \hat{v}_{zk}$. This results in the following error between the estimated and the real states.

$$\begin{bmatrix} \tilde{p}_{zk+1} \\ \tilde{v}_{zk+1} \\ \tilde{v}_{ok+1} \end{bmatrix} = \begin{bmatrix} 1-K_{zp} & T & -\frac{T^2}{2}R(q_{zk}) \\ 0 & 1-K_{zv} & -TR(q_{zk}) \\ 0 & 1-K_{v_o} & 1 \end{bmatrix} \begin{bmatrix} \tilde{p}_{zk} \\ \tilde{v}_{zk} \\ \tilde{v}_{ok} \end{bmatrix} + \begin{bmatrix} 1 & -\frac{T^2}{2}R(q_{zk}) & 0 \\ 0 & -TR(q_{zk}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_{p_{zk}} \\ \eta_{a_{zk}} \\ \eta_{vk} \end{bmatrix} \tag{3.21}$$

The error vector e can be rewritten as follow

$$e(k+1) = (A - KC)e(k) \tag{3.22}$$

the error of the the state i.e. $(\hat{x} - x)$ converges to zero as $k \to \infty$ if and only if the matrix $(A - KC)$ is stable:

$$A - KC = \begin{bmatrix} 1-K_{zp} & T & -\frac{T^2}{2}R(q_{zk}) \\ 0 & 1-K_{zv} & -TR(q_{zk}) \\ 0 & 1-K_{v_o} & 1 \end{bmatrix} \tag{3.23}$$

3.5 Navigation system implementation

3.5.1 Complementary filters coupling

This section describes the overall architecture of the proposed navigation system based on nonlinear complementary filters (see figure 3.3). The system combines

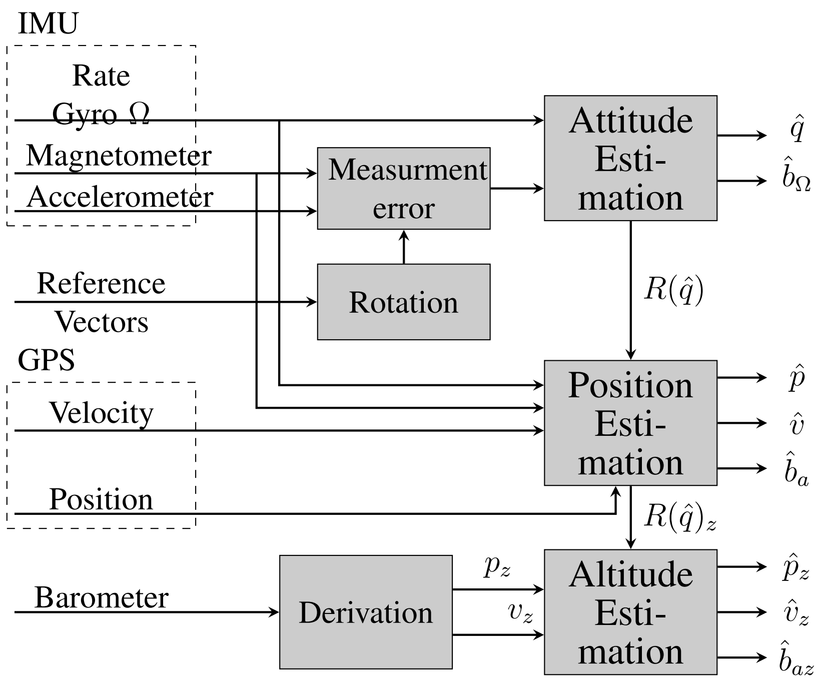


Figure 3.3: Navigation system architecture

between separated complementary filters for attitude, position, and altitude. The design of the navigation system structure is based primarily on the ease of implementation and the low computational load in a low-cost, low power hardware architecture. Furthermore, in this design the filters can be tuned totally independent of each other. In this structure, the attitude is always provided and the measurement of altitude is always available even in GPS drop.

The entries to the attitude filter structure as shown in figure 3.3 are measurements from IMU and a reconstruction of vector observations, such as Earth's magnetic field and the gravitational acceleration. A typical IMU consists of rate gyros, accelerometers, and a Magnetometer. The proposed attitude filter estimates the attitude of the vehicle expressed in quaternion and compensates for the rate gyro bias online. The filter ensures almost global stability of the observer error and provide a formulation in terms of the measurement errors. The continuous time form of the filter can be expressed as follow:

$$\begin{cases} \dot{\hat{q}} = \frac{1}{2}\hat{q} \otimes P\left(\Omega - \hat{b} + K_{\mathrm{P}} \sum_{i=1}^{n} k_i v_i \times \hat{v}_i\right) \\ \dot{\hat{b}} = -K_{\mathrm{I}} \sum_{i=1}^{n} k_i v_i \times \hat{v}_i \end{cases} \tag{3.24}$$

The attitude quaternion from the attitude filter is adoptd in the kinematics of the position filter for low accelerated vehicles (i.e. quadrotors).

$$a_n = \dot{\hat{v}}_n - g e_3 \tag{3.25}$$

where $\dot{\hat{v}}_n$ is the estimated linear acceleration in NED frame obtained by the estimation process in the position filter. For high accelerated vehicles, the velocity complementary filter introduced in [24] is considered to estimate a vehicle's velocities and linear accelerations in NED coordinates. This filter requires measurments of the angular speed from a gyroscope and the body accelerations from an accelerometer, as well as the velocity measured in NED frame by a GPS receiver.

$$\begin{cases} \dot{\hat{v}}_n = k_1(v_n - \hat{v}_n) + ge_3 + Qa \\ \dot{Q} = QS(\Omega - \hat{b} + k_v(v_n - \hat{v})a^T \end{cases} \tag{3.26}$$

where k_1 and k_v are positive gains. Q $\in \mathbb{R}^{3\times3}$ is a virtual rotation matrix and $\Omega \in \mathbb{R}^3$ is the angular speed measured by the gyroscope. $v_n \in \mathbb{R}^3$ is the actual velocity meseared in NED coordinates by a GPS recievers and $\hat{v}_n$ is the estimated velocity. Theoretically, This observer assumes perfect measurement to ensure a abounded matrix $|Q|$. In practice, sensor noises and biases can possibly yeild large estimation errors. To improve the gyro measurments in this algorthim the bias estimates $\hat{b}_\Omega \in \mathbb{R}^3$ obtained by the attitude estimation algorthim are used. Also the noises and drifts in the accelrometer were elemenated by static calibration as discussed in section 3.3. With that, Q can be bouned with Frobenius norm to avoid numerical drifts.

$$Q = \frac{Q}{|Q|_F} \tag{3.27}$$

For bounded signals as given in section 3.2.4, this observer can ensure a semi-global exponential convergence of the estimated velocity $\hat{v}$ and the linear acceleration in inertial frmae Qa. furthermore, this observer provide a global exponetial convergence if the acceleration $\dot{v}$ is constant.

The height estimated by the measurements from GPS is not accurate enough to have control in vehicle altitude. Thus the height measurements from pressure sensor are adopted in altitude estimation with the vertical acceleration using a separated complementary filtering approach as follow

$$\begin{cases} \dot{\hat{p_z}} = \hat{v_o} + \hat{v}_{zn} + k_1(p - \hat{p_z}) \\ \dot{\hat{v_o}} = k_2(v_z - \hat{v_z}) \end{cases} \tag{3.28}$$

3.5.2 Filters tuning

The complementary filters discussed in pervious section have many parameters to be tuned. Setting these values randomly is not convenient and can lead to divergence. For this purpose the values of these parameters are calculated according to an approach minimizing the mean square error MSE (variance) for a set of experimental acquisitions in order to achieve the most precise estimation of the vehicle attitude. This error is given by comparing the estimated attitude in quaternion, position, and velocity given from the filters with the attitude, position, and velocity measured by a trusted reference (e.g. Optitrack system) and defined as in the following formula:

$$\begin{cases} MSE_{att} = \frac{1}{n}\sum_{j=1}^{n}\Big(q_j - \hat{q}(x_j, \mathcal{K}_a)\Big)^2 = E(\tilde{q}^2) \\ \\ MSE_{pos} = \frac{1}{n}\sum_{j=1}^{n}\Big(p_j - \hat{p}(x_j, \mathcal{K}_p)\Big)^2 = E(\tilde{p}^2) \\ \\ MSE_{vel} = \frac{1}{n}\sum_{j=1}^{n}\Big(v_j - \hat{v}(x_j, \mathcal{K}_v)\Big)^2 = E(\tilde{v}^2) \end{cases} \tag{3.29}$$

where q_j, p_j, and v_j (for j =1, 2, ... , n) are finite-length, discrete signals of the trusted reference quaternion, position, and velocity respectively. And $\hat{q}_j$, $\hat{p}_j$, and $\hat{v}_j$ (for j =1, 2, ... , n) are the discrete signals of the corrosopinding estimated measurements derived by the filters. The output model of each filter is represented as a function of collection of known signals x and a vector of adjustable constant parameters $\mathcal{K}$.

The goal is to find the parameter values for the given model which fits the given actual data. The least squares method finds its optimum when the MSE is a minimum according to the following relation

$$\mathcal{K} = \min_{\mathcal{K}}\{MSE\}, \; where \; \mathcal{K} \geq 0 \tag{3.30}$$

Chapter 4

Smartphone based Quadrotor: Real-Time Implementation of Guidance, Navigation and Control Systems

In civilian applications where missions don't require direct or continuous human control, an implementation of autonomous guidance system is needed to steer the UAV toward a desired target. The work activities in this chapter includes a development of guidance system that gives some autonomy for quadrotor platform. This system can provide the ability of aerial robot to follow waypoints and to execute automatic takeoff and landing. This chapter also recall the trajectory controller and the navigation system proposed in chapter 2 and chapter 3 respectively. The navigation system plays a key role in fuse information from many sensors to estimate the vehicles attitude, position, angular rates, altitude, velocity with respect to some fixed point of reference or relative to some target. The trajectory control loop receives the guidance commands and uses navigation information to generate the actual actuator signals to follow the guidance targets as well as to stabilize the vehicle's attitude. The overall structure of guidance, navigation and control systems as shown in figure 4.4 are implemented on a developed smartphone based quadrotor.

The first two sections present the hardware structure of the developed smartphone based quadrotor and the adopted flight system architecture for performing autonomous flights. Then the following sections propose the real-time implementations of the guidance, navigation and control systems.

4.1 Smartphone based quadrotor

Figure 4.1 shows the developed smartphone based quadrotor and its structure. The core of this work is to use the smartphone as flight control system for a quadrotor. The concept of this work tries to address several issues typical of quadcopter drones development. First of all, this concept offers the advantage of a reduction of the development time and costs, cause of the integration of different hardware components — *i.e.* flight control system computing unit, dynamics sensors and communication devices — in a commercial product that is characterized by certified performance and operating life requirements. Moreover, the *Android* operating system allows the use of a wide and well-structured framework of Application Programming Interfaces (APIs). This enables an effective interaction with various sensors and communication systems included in the smartphone.

Regarding the operating capabilities, the much more powerful Central Processing Unit (CPU) of the smartphone with respect to most used commercial control boards for civil UAV allows the implementation of complex automatic control solutions. This should make possible also to implement a flying vehicle with the capability to perform completely autonomous flight operations. The wide and powerful communication capabilities (*i.e.* Bluetooth, *WiFi* and 3G/4G phone and internet connections) enable moreover to operate the quadrotor in a novel operating mode in terms of control. This mode allows both autonomous navigation and control of the quadrotor and execution of high-level commands remotely provided on real-time by a human operator placed at a much greater distance with respect to currently available platforms.

4.1.1 Quadrotor platform

The developed small-scale quadrotor used in this work is shown in figure 4.1. The design of this platform is based mainly on important aspects namely the reliability, safety, endurance, price reduction, and ease of use. Its airframe is a carbon fiber tubular structure providing a light-weight and high-stiffness mechanical layout. A set of 4 *Dualsky XM-400* BrushLess Direct Current (BLDC) electric motors [19] are used to generate the thrust. While these motors drive 4 *APC* fixed-pitch propellers [28] with a 10 inches diameter and 4.5 inches pitch. The total maximum thrust can be generated by the four rotors is approximately 2.5 kg. A *3S* Lithium Polymer (Li-Po) battery is used to supply the power to the engine and the rest of the electronics. The overall size of the quadrotor platform is $(600 \times 600 \times 100)$ mm with a total weight of 1150 $gram$ including also the smartphone.

The speed of each electric motor is controlled using Pulse-Width Modulation (PWM) in order to achieve a desired thrust for each rotor . The PWM commands are send to the Electronic Speed Control (ESC) through a *IOIO-OTG* bridge board [9] connected to the Micro-USB connector of the smartphone. This board converts the PWM request generated by the control law described in section 2.3 to a pulse with modular digital signal that is fed to the ESC of each motor. All the components of the proposed quadrotor and their characteristics are summarized in Table 4.1.

1. Vibration Reduction

   The vibration caused by the rotation of the motors is an important issue that must be taken into account during the design of the quadrotor. The motors in the vehicle are pretty powerful, and with imbalances in propellers and high speed rotoations can generate a lot of vibrations that add noise in the inertial

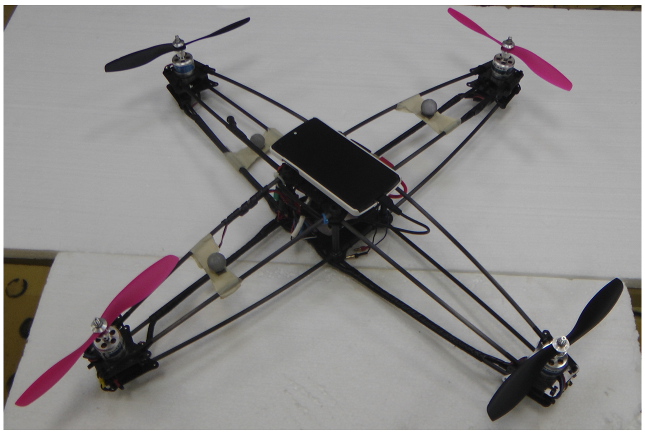


Figure 4.1: The quadrotor prototype

Table 4.1: The quadrotor components and characteristics

Part	Specifications	weight (gram)
Structure	carbon fiber (D=600 mm)	250
4× BLDC Motors	DualSky XM400 Kv:$1180 RPM/V 130W$	4× 50
4× *APC* propellers	10" × 4.5"	4× 10
Li-Po battery	3 cells 11.1 V 2650 mAh	225
4× ESC	20 A (30 Max)	4 × 15
Smartphone	Nexus 5	130
IOIO-OTG board	mini-USB connection	50
Power Module	outputs $5.3V, 2.25A$	15
Wires and screws	-	170
	Total weight	1150

sensors. These noise can effect the state estimations inside the autopilot which is the most important part in the UAV, should be isolated from vibrations, in addition to sensors and cameras. To reduce the vibration, a suitable spring damping system is used for this purpose as shown in figure 4.2

4.1.2 Hardware structure

To evaluate the performance of the developed smartphone based quadrotor the full autopilot has been implemented and tested by real indoor and outdoor flights. To accomplish this task different implementations and experimental setups are needed. Figure 4.3 shows the full hardware structure of the quadrotor.

1. Smartphone

   All of the estimation and control algorithms are executed in a *Google Nexus 5* smartphone [20] with *Android* operating system acting as computing unit for the flight control system. It weighs 130 gram and it is characterised by

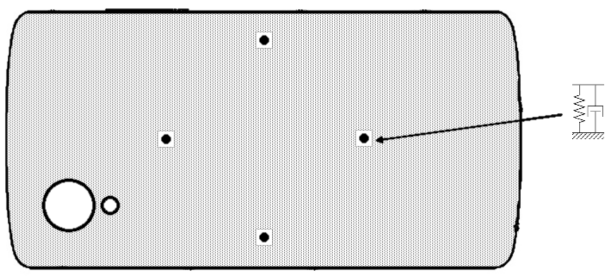


Figure 4.2: Spring and Damper systems for vibration reduction

a hardware structure particularly suitable for quadrotors control. The *Qualcomm Snapdragon 800* 2.26GHz quad-core board used in this smartphone provides a much higher computational power in respect with commercial control boards for civil UAV (*e.g.* PixHawk [38] and Naza [1]) according to the discussion given in [34]. Furthermore, this CPU allows floating point computation. An *Adreno 330* 450Mhz Graphics Processing Unit (GPU) is then used to increase the smartphone computational speed. The *Broadcom BCM4339 5G Wi-Fi combo chip* allows standard wireless Local Area Network (LAN) communication.

The inertial sensors in smartphone provide sufficient bandwidth to accurately capture the the dynamics of the quadrotor. It contains three-orthogonal gyrometers, three-orthogonal accelerometer, and a magnetometer (compass). The smartphone equipped also with a low-frequency GPS receiver, and low-frequency pressure sensor. The sensor fusion and the estimation strategies using those sensors on the quadrotor have been discussed in details in the design of the navigation system in chapter 3.

2. Ground Control Station (GCS) and communications

   The GCS provides various important functionalities. The main role of GCS is to display the flight information transmitted from the UAV in real-time via bluetooth and to send via Wifi the various operating commands. The GCS based on a *Dell Optiplex 980* desktop computer with processor *Intel*® *Core*™ *i7* processor and operating system *Windows 7 Professional SP1* for 64-bits systems is used to control the quadrotor state in real-time in in-door flight environment. And laptop computer is used to control the quadrotor in real-time in outdoor flight environment. The *OptiTrack* system [**?**] communicates to the computer with high-precise estimation in real-time of the quadrotor position and attitude in the local NED reference frame with an update rate of 10 hertz. The data on the quadrotor position is also then sent through User Datagram Protocol (UDP) packets to the smartphone on board of the quadrotor. The GCS acts also as interface between the joypad and the quadrotor. In a *manual mode* the quadrotor autopilot receives the position command inputs from a joypad connected with the GCS as shown in figure 4.3. The joypad is the *Logitech Dual Action*^{TM} gamepad communicating with the GCS through UDP and the UDP packets are then sent to the smartphone. Every UDP packet containing the data about the current state of the joypad, in terms of analog controls position and buttons pressed, is sent every 20 ms. The joypad is enabled also in the autonomous mode as safety precaution in order to be able to switch off motors in case of emergency.

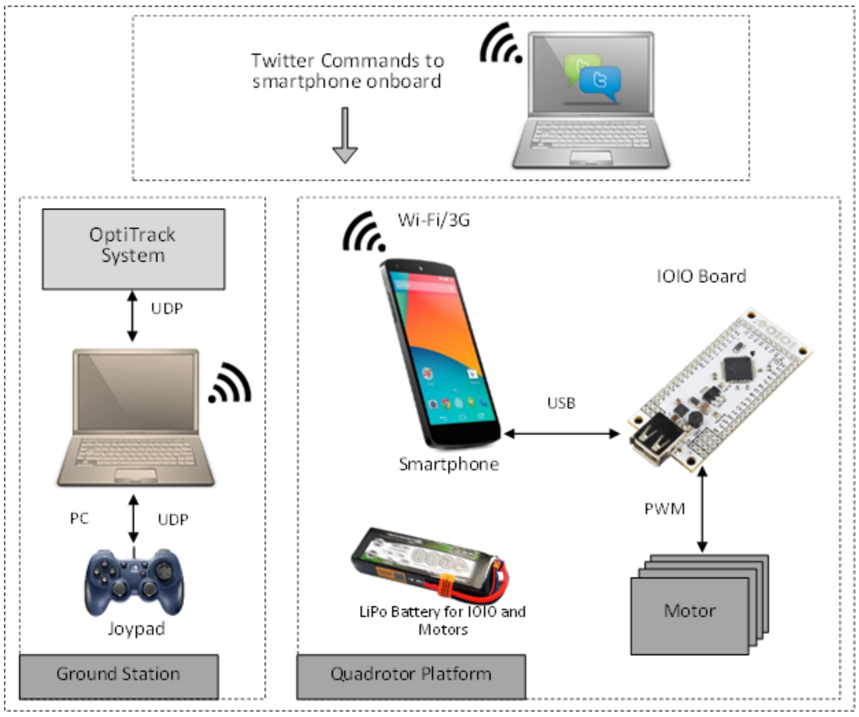


Figure 4.3: Hardware structure

4.2 Guidance, navigation and control architecture

The developed autopilot in smartphone includes onboard guidance, control and navigation systems that were described in details in previous chapters and sections. In this section, we recall briefly the main components of these systems and their real-time implementation as illustrated in figure 4.4. The control architecture of the proposed quadrotor is characterised by the following main functional layers:

- **Reactive supervisory system** allows the human operator to define the mission commands using a handy protocol implemented by *Twitter* client
- **Basic guidance system** generates a real-time and practical trajectory for the assigned mission. This strategy is responsible of generating straight-line path between the defined navigation waypoints and produces the real-time desired trajectories which are the desired 3D position.

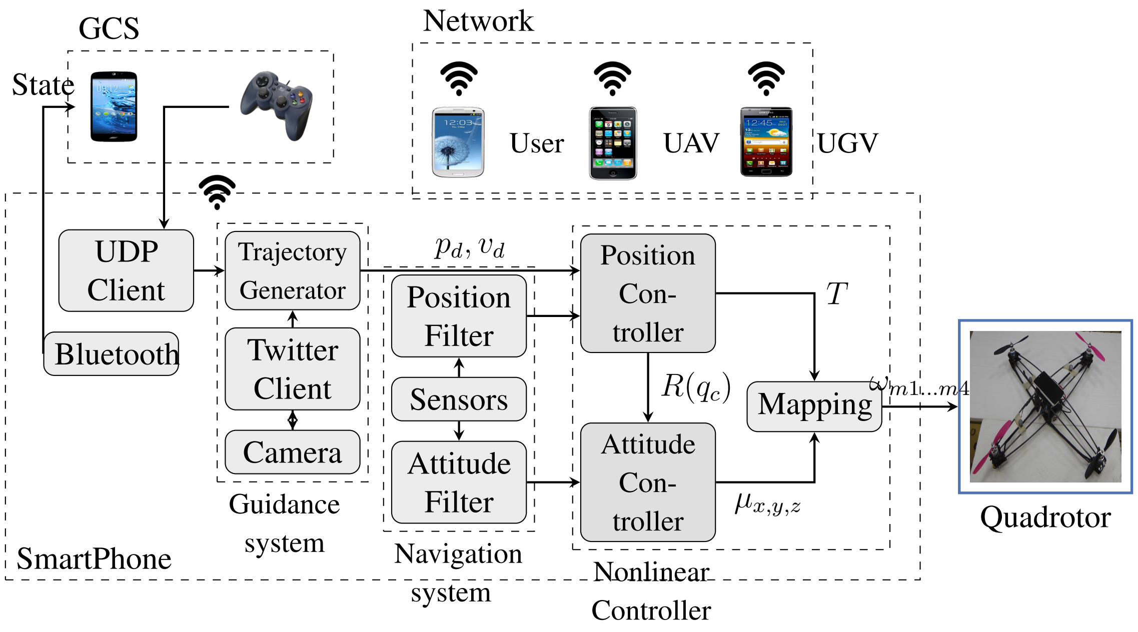


Figure 4.4: System architecture of the UAV

- **Trajectory tracking controller** evaluates the control actions necessary for stability and trajectory tracking. Also it calculates the distribution of the rotors angular speed necessary to get as resulting thrust and torques those required by the control wrench. See chapter 2 for more details on the adopted nonlinear controller.

- **Navigation system** executes nonlinear complementary filters for sensor fusion and state estimation that allow the autonomous flight of the quadrotor. chapter 3 discuss in details about the design of the navigation system.

The adopted four functional layers as shown in figure 4.4 are characterized by different level of computational complexity. More precisely, the first one works as an intermediary environment between the user (operator) and the UAV. This layer allows the user to implement special commands of the mission remotely via *Twitter* client. The operator in this layer can define the waypoints for a mission as well as he can ask for flight information and access the camera. The basic guidance system generates the trajectory based on several waypoints defined by a remote user. This system can provide the ability of aerial robot to follow waypoints and to execute automatic takeoff and landing. The reactive supervisory strategy is introduced in the following sections. Third layer is able to globally stabilize the trajectory generated by the supervisor. The choice of a globally stabilizing tracking controller is motivated by possible deployment scenarios, such as when the quadrotor is launched from arbitrary configuration, in which the vehicle can be overturned in the initial conditions. This layer is also in charge of distributing the rotation speeds between the different rotors in order to obtain the required overall thrust and torques computed by the control. Finally, the navigation system layer plays a key role in fuse information from many sensors to estimate the vehicle's state (*i.e.* attitude, position, angular rates, altitude, and velocity) with respect to some fixed points of reference or relative to some targets. The last two layers namely the trajectory tracking controller and the navigation system are discussed previously in chapter 2 and chapter 3 respectively.

4.3 Reactive guidance system

This section presents the reactive protocol of the mission commands that defined by a normal human operator using a network interface. Thanks to the powerful communication capabilities of the smartphone, it was possible to implement a *Twitter* client acting as communication medium. This *Twitter* client completely frees the developed quadrotor from remote control range limitations. In fact, till the vehicle is in an area covered with mobile phone signal, it is able to connect to internet and to receive operator's commands and to send back its status, if requested by the remote operator. Selected *Twitter* user's timeline is periodically polled in search for new command. In particular, the proposed application is able to recognize messages containing waypoints navigation instructions and/or status report request. The general protocol of each twitter command is given as follows:

@Quadrotor_Unibo # quad_com < progressive number><keyword> [parameters]

where <progressive number> represents the order of each command provided in each mission. The $< keyword >$ is the word TwitterThread switches to detect the required command. And the $< parameters >$ separated by blanks is a function of the keywords. A brief description of these commands is given in Table 4.2.

Table 4.2: definition of the commands issued to the quadrotor through the *Twitter* interface

keyword	Example	Description
Report	#quad_com 1 report	The quadrotor tweets its attitude and position with the current app screen as attachment
Camera	#quad_com 1 camera	The quadrotor uploads a snapshot taken from its camera
Waypoints	#quad_com 1 waypoint 2 2 100 100 100 000 000 000	The autonomous flight mode is engaged and the quadrotor take off and starts following the generated path between waypoints
Land	#quad_com 1 land	The quadrotor land and shut down the rotors then tweets with confirmation message

As soon as a command is detected, the quadrotor autonomously updates its own timeline tweeting its status and uploads its camera current vision, if requested by the remote operator. In the case of a waypoints navigation command, a ramp shape signal for position setpoints is generated based on actual position, desired position and velocity request. The touch-screen user interface permits to set as preferences Twitter client activation. If necessary, it can also be used to specify which user's timeline is going to be scanned.

Developing autonomous UAVs requires some level of safety to avoid risks resulted by unexpected failures. Some undesired attitudes can be faced during the flight which normally caused by improper sensors measurements, controller drawbacks, or mechanical fails. Then the autopilot switch to manual mode to give the control to the operator at the GCS to send the right decision.

4.3.1 Twitter client

The Twitter Client is a class that allows a user to communicate remotely with the quadrotor. In particular, every user can send commands via his personal Twitter account to the quadrotor account based on a specific protocol. This client can provide a working communication protocol of one-to-many which uses the famous social network as intermediate platform as shown in figure 4.5.

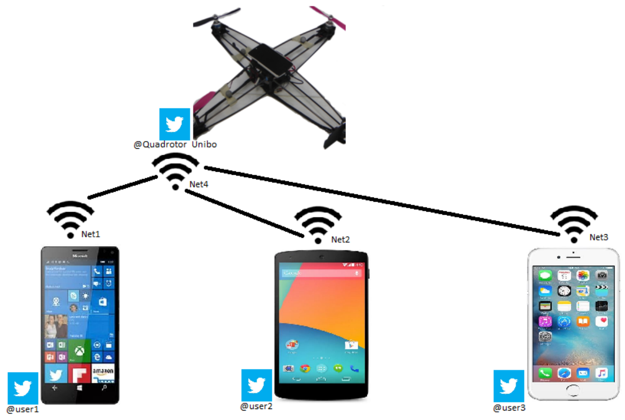


Figure 4.5: Twitter client structure

Twitter client class is created with a constructor that provides all the needed sensors information, controller outputs and the application conditions as shown in figure 4.6. During its execution, Twitter thread uses Twitter instance to connect to the quadrotor twitter account and to access its timeline. Then, a Timer task set to allow new tweets arrived to quadrotor account. Only if the tweets include the right statement order, they will processed and an answer tweet with the mention of requiring user is posted; otherwise, the tweets are discarded.

4.3.2 Trajectories generation

This part introduces the strategy followed in this work to generate a real-time practical trajectory according to the selected flight mode. This strategy is responsible

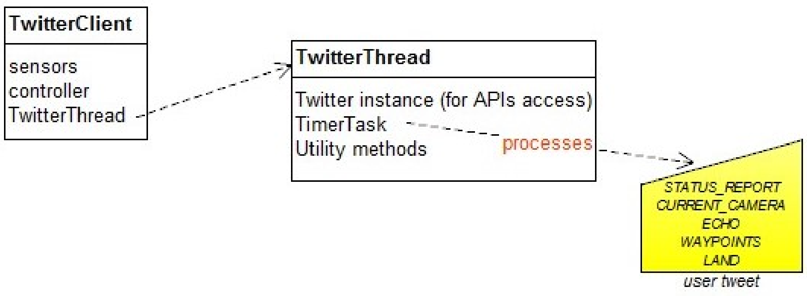


Figure 4.6: Comunication structure for guidance system

of generating straight-line path between the defined navigation waypoints and produces the real-time desired trajectories which are the desired 3D position:

$$\begin{cases} p_d(t) := (x_d(t),\ y_d(t),\ z_d(t)) \\ \dot{p}_d(t) := (\dot{x}_d(t),\ \dot{y}_d(t),\ \dot{z}_d(t)) \\ \psi_d(t) \\ \dot{\psi}_d(t) \end{cases} \tag{4.1}$$

where $p_d \in \mathbb{R}^3$ and $\psi \in \mathbb{R}$ are the desired position and the vehicle's heading respectively. $\dot{p}_d \in \mathbb{R}^3$ and $\dot{\psi} \in \mathbb{R}$ are their time derivatives. The acquired position and velocity trajectories are differentiable with respect to time, which is needed in the control design as explained in chapter 2. A limitation on maximum speed and acceleration has been performed on the kinematic model used for trajectory generation. The trajectory generator will has the following form:

$$\begin{cases} \dot{x}_d(t) = V_d(t)cos(\psi_d(t)) \\ \dot{y}_d(t) = V_d(t)sin(\psi_d(t)) \\ \dot{z}_d(t) = V_{zd}(t), \end{cases} \quad \dot{V}_d(t) = \begin{cases} a_d\ \ Starting \\ 0\ \ V_{max} \\ -a_d\ \ Stopping \end{cases} \tag{4.2}$$

where $V_d \in \mathbb{R}$ is the linear speed along the lateral path, and $V_{zd} \in \mathbb{R}$ is the desired vertical velocity and $a_d \in \mathbb{R}$ is the desired acceleration. When the vehicle reaches one of the waypoint, it starts decelerating in order to arrive the waypoint by zero velocity. This strategy allows better transition from one waypoint to the next without overshoot on sharp corners of the generated line segments. A smooth vertical reference trajectory $z_d(t)$ are generated in real-time based on the desired velocity and the current altitude of the vehicle. This trajectory ensures smooth take-off and soft landing.

4.4 Real-time Implementation of Guidance, Navigation and Control Systems

This section includes implementations of several activities for a complete autopilot capable of performing autonomous tasks. A Graphical User Interface (GUI) helps the user to interact with the application easily by taking in advantages the touch screen in the smartphone. Several windows have been provided for a user to calibrate the inertial sensors and the motors, to tune the filters and the controller, to define a trajectory waypoints, and to start additional activities such as saving photos and videos of the surrounding environment. The application framework provided in this work is based on Android operating system. Android allows to build apps in a Java language environment.

4.4.1 Software structure

From the point of view of the software, the core of this work is an implementation of non-linear controllers and filters for the attitude and the position on this smartphone based quadrotor. It also includes the user interface to arm engines, initialize estimators and controllers states, open the UDP communication interfaces with the GCS and to issue to the quadrotor the command to take-off. The software structure of the autopilot is shown in figure 4.7.

As widely discussed in [7], *Android* is a quite powerful and versatile operating system characterized by a very refined resource management system. This allows to effectively run multi-thread applications with different priority levels. Moreover, its APIs allows an easy access for programmers to the infrastructure managing the interactions of the computing unit with sensors and radio-communication interfaces. Of However, *Android* is not a Real-Time Operating System (RTOS) and it is not therefore able to provide real-time data processing without buffering delays. This makes *Android* significantly less performant in real-time control applications with respect to commercial drone controllers. Furthermore the standard pose estimation algorithms based on smartphone sensors provided by *Android* APIs is not characterized by a good level of precision and accuracy. However the results of this work shows that it is possible to produce software based on *Android* capable of getting good performances for the control of a quadrotor drone with the use of algorithms taking into account these limitations of the *Android* operating system.

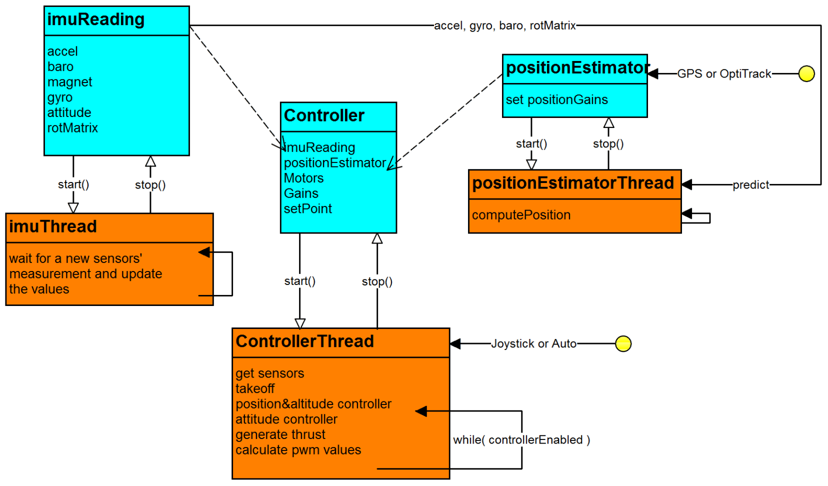
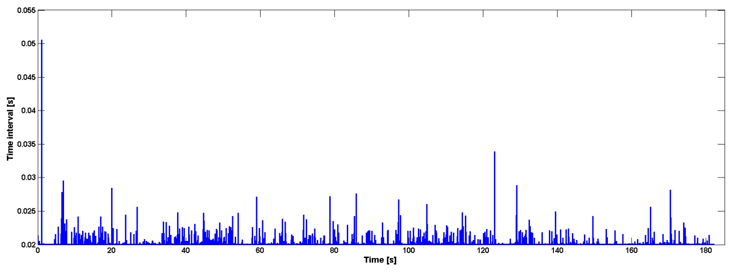
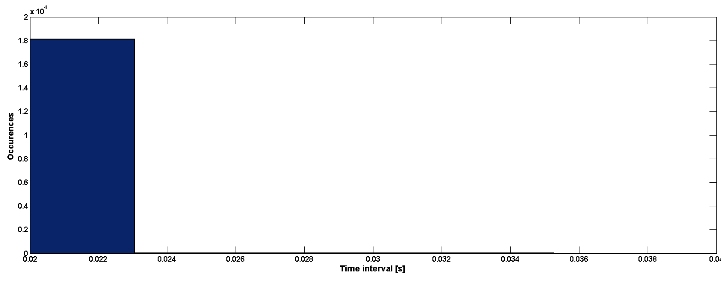


Figure 4.7: Software structure

The adopted algorithms run according to a multi-thread structure (see figure 4.7) designed in order to separate semantically different tasks and provide a light and tidy structure. This architecture allows also to set different priority levels for the different services and supports therefore an efficient resource management of the operating system. The application structure sets every service to run in a thread continuously evaluating the time passed from the previous iteration with a precision of $1e^{-9}s$. This allows to obtain a low-variance update time profile, such as in the samples in figure 4.8. More precisely, during a sampling time of more than $270s$ the mean value of the sample time is $0.0201s$ with a variance of $2.3394e^{-7}.s^2$. Moreover, the sensors acquisitions update task is managed by the *Android* operating system resource manager according to a profile providing the sensors readings as soon as they are updated. This profile provides an update rate between $50Hz$ and $250Hz$ with a mean value of $100Hz$. A fine tuning of the estimation and control algorithms parameters is used to sensibly reduce the problems related to the non-constant update time. A logger thread is also included to allow the record with an update rate of $100Hz$ of real-time flight parameters including NED position, speed, attitude, regulator input and output signals.



(a)

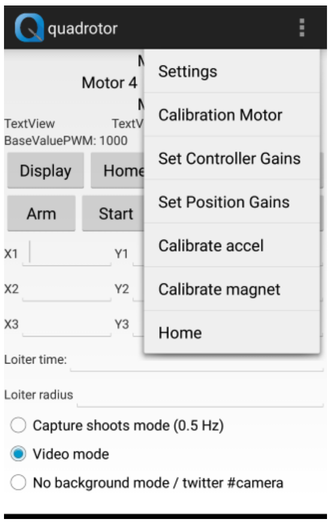


(b)

Figure 4.8: sample time distribution for the *Android* service of the quadrotor position estimation: (a) plot of sampling time versus observation time, (b) hystogram of the sampling time distribution

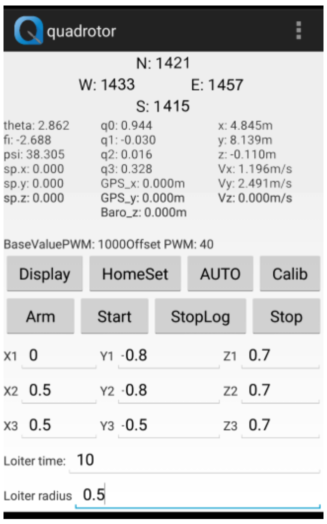
4.4.2 Quadrotor app

The quadrotor app is built as a combination of various components (activities) that can be executed individually for different purposes. Figure 4.9 shows the main window of the app and the drop-down menu for the activities that are described as follows:



Motor 4
TextView TextV
BaseValuePWM: 1000
X1 Y1
X2 Y2
X3 Y3
Loiter time:
Loiter radius
Capture shoots mode (0.5 Hz)
[✓ Video mode]
No background mode / twitter #camera

(a)



```
              N: 1421
      W: 1433          E: 1457
              S: 1415
theta: 2.862   q0: 0.944         x: 4.845m
fi: -2.688     q1: -0.030        y: 8.139m
psi: 38.305    q2: 0.016         z: -0.110m
sp.x: 0.000    q3: 0.328         Vx: 1.196m/s
sp.y: 0.000    GPS_x: 0.000m     Vy: 2.491m/s
sp.z: 0.000    GPS_y: 0.000m     Vz: 0.000m/s
               Baro_z: 0.000m

BaseValuePWM: 1000Offset PWM: 40

X1 0      Y1 -0.8     Z1 0.7
X2 0.5    Y2 -0.8     Z2 0.7
X3 0.5    Y3 -0.5     Z3 0.7
Loiter time:  10
Loiter radius  0.5
```

(b)

Figure 4.9: Quadrotor's app: (a) Drop-down menu for the auxiliary activities (b) Working app

- **Main activity** that execute the crucial part of the system (i.e. sensors readings, filter and controller algorithms, and comunications). This acivity also **Display**s in real time on the screen the important information of the flight which give an indication for the user about the initial conditions before starting the flight. Moreover, on this screen the user can **Arm** the motors and **Start** and **Stop** the controller manually as well as it helps to set the desired waypoints of a flight in **Auto** mode. The user can also stop loging the flight data to the SD card by pushing on **StopLog** and he can choose for the camera option to save pictures or a video during the flight.

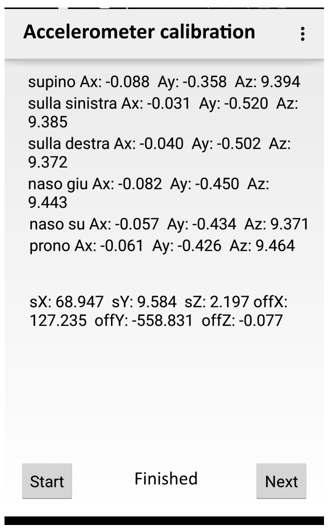
- **Accelerometer calibration** that cancels the static biases and scale factors errors in the accelerometers based on the proposed algorithm described in chapter 3 for better state estimations. The user follows the calibration procedures step by step with the associated instructions as shown in figure 4.10(a).

- **Magnetometer calibration** that used to eliminate the offsets and the scaling factors in the measurements of the magnetic field based on the algorithm that adopted for this purpose as it has been introduced in chapter 3. The user need to run this activity and start rotating the smartphone in all directions as mush as he can to achieve better calibration. 4.10(b) shows the calibration activitiy of the magnetometer.

- **Set Controller gains** allows the developers tuning gains and parameters of the adopted controller as discussed in chapter 2, directly in the experiment location to achieve the quadrotor stability. The defined gains are saved permanently in a memory location for next flights (See the activity in figure 4.10(c)).
- **Set position gains** helps the developer to assigns the best gains for the position filters when he move by the experiments from indoor to outdoor as shown in figure 4.10(d).
- **Motors calibration** provides step-by-step instructions for calibrating the ESCs of the motors to define the minimum and maximum PWM values that the flight controller will send.

4.4.3 Ground control station

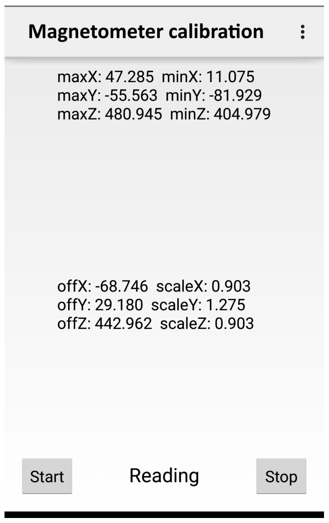
The Ground Control Station (GCS) includes a device that provide the operator with the flight information and allow to define the navigation waypoints for following a trajectory. The implementation of this platform has been performed on android device to work as one of the basic tools which are equipped with the UAV. Therefore it allows you to monitor and control in real time remotely. The GCS can be connected to the aerial platforms with different communication ways. To take in advantages the communications capabilities that provided by smartphones, The Bluetooth can provide an effective connections bridge between GCS and the UAV.


Accelerometer calibration

supino Ax: -0.088 Ay: -0.358 Az: 9.394
sulla sinistra Ax: -0.031 Ay: -0.520 Az: 9.385
sulla destra Ax: -0.040 Ay: -0.502 Az: 9.372
naso giu Ax: -0.082 Ay: -0.450 Az: 9.443
naso su Ax: -0.057 Ay: -0.434 Az: 9.371
prono Ax: -0.061 Ay: -0.426 Az: 9.464

sX: 68.947 sY: 9.584 sZ: 2.197 offX: 127.235 offY: -558.831 offZ: -0.077

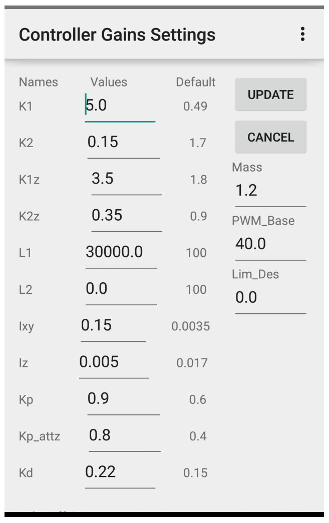
(a)


Magnetometer calibration

maxX: 47.285 minX: 11.075
maxY: -55.563 minY: -81.929
maxZ: 480.945 minZ: 404.979

offX: -68.746 scaleX: 0.903
offY: 29.180 scaleY: 1.275
offZ: 442.962 scaleZ: 0.903

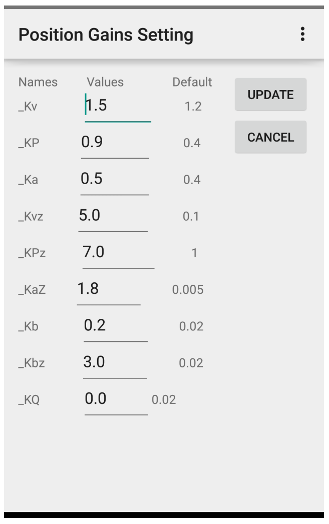
(b)


Controller Gains Settings

Names	Values	Default
K1	5.0	0.49
K2	0.15	1.7
K1z	3.5	1.8
K2z	0.35	0.9
L1	30000.0	100
L2	0.0	100
Ixy	0.15	0.0035
Iz	0.005	0.017
Kp	0.9	0.6
Kp_attz	0.8	0.4
Kd	0.22	0.15

Mass 1.2
PWM_Base 40.0
Lim_Des 0.0

(c)


Position Gains Setting

Names	Values	Default
_Kv	1.5	1.2
_KP	0.9	0.4
_Ka	0.5	0.4
_Kvz	5.0	0.1
_KPz	7.0	1
_KaZ	1.8	0.005
_Kb	0.2	0.02
_Kbz	3.0	0.02
_KQ	0.0	0.02

(d)

Figure 4.10: Auxiliary activities: (a) Accelerometer calibration (b) Magnetometer calibration (c) Set Controller gains (d) Set position gains

Chapter 5

Advanced Guidance System based on Robot Operating System (ROS)

The use of UAVs in Search And Rescue (SAR) operations controlled by a single operator is more efficient in target localizations on difficult terrains with respect to Unmanned Ground Vehicle (UGV) [13]. Generally speaking, the ability of UAVs to operate autonomously by using high level computations onboard makes them ideal platforms for SAR operations. These considerations contributes to the interest of researchers for the development of quadrotors for SAR applications [39]. Many parameters affect the search tasks performances such as the quality of sensory data, energy constraints, data sharing between UAV and operator, and external hazards (wind, obstacles, etc ...) [60]. Many researchers recently focused towards making these vehicles able to autonomously explore and operate in outdoor environments. The work in this chapter moves in the direction of the development of a quadrotor with payload and outdoor navigation performances suitable to SAR applications.

5.1 Quadrotor Application in SAR Missions

The reason supporting the introduction of quadrotors in SAR missions are multiple and can be summarized in the following list:

1. extend the human rescuer situational awareness,
2. enlarge the area patrolled by the human rescuer,
3. reduce the duration of search phases and the risks for human rescuers in the localisation and victims,
4. search of clues of victims presence and emergency signals,
5. first support of discovered victims after their detection

More precisely, the first functionality is obtained through the quadrotor's capability to capture images of the environment through its camera. These images are used for a 3D scene reconstruction based on the fusion of the images on a geo-referenced cognitive map.

The second functionality is related with the capability of the quadrotor to send in real-time to the rescuer the visual information acquired by the camera and the information acquired by other sensors. Through its camera, it act, therefore, as additional flying eyes of the rescuer and provide him up-to-date aerial imagery of the rescue site. The quadrotor can therefore have a very important role in supporting the rescuer in the search phases of SAR missions.

The third one is based on the light and agile nature of the quadrotor. This allows it to fly with an higher maneuverability in respect with larger fixed wing and helicopter UAVs. This permits a more effective deployment in dangerous areas that are inaccessible or difficult to be reached by human rescuers (e.g. cliffs, overhangs, steep terrains), as well as in darkness through infrared-imaging. Moreover, the possibility to employ multiple quadrotors allows to distribute the tasks among multiple agents, allowing a quicker execution and boosting the efficiency of the mission. Quadrotors can finally perform a coarse search prior to the arrival of the human rescuer.

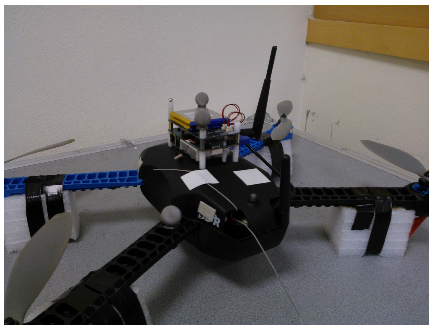
The fourth case is implemented through the elaboration of the infrared-imaging. Moreover, in an avalanche scenario, this task can be supported through the use of ”Appareil de Recherche de Victimes en Avalanche (ARVA)” active avalanche transceivers, used to scan very rapidly the avalanche area and detect the victims presence.

The fifth functionality requires the quadorotor to interact verbally and guide victims to a safe position, when they are out of reach from an human rescuer.

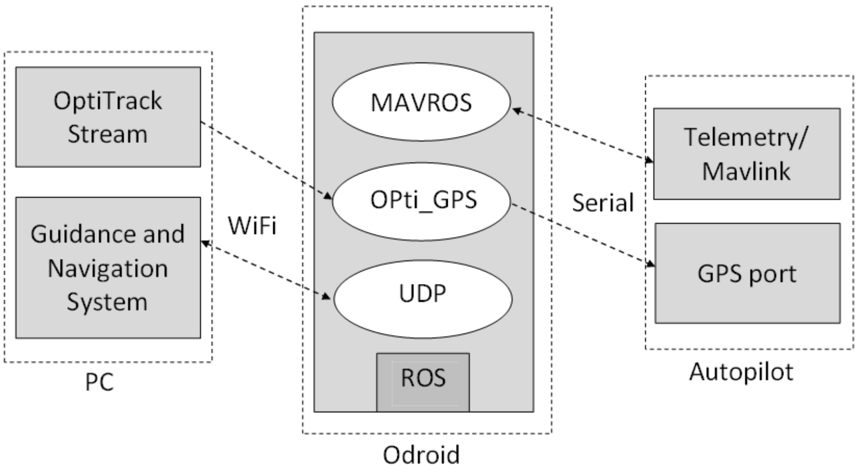
5.2 Developed quadrotor for high-level operations

In missions such as search and rescue operations quadrotors are needed to acomplish several complex tasks autonomously. For this, a high level supervisory guidance system are applied on the quadrotor for making the right decisions based on high level sensors such as cameras. In this work we have modified a standard quadrotor platform with a powerful linux computer mounted onboard as shown in figure 5.1(a). This computer addresses the computational cost for the implementation of complex algorithms using ROS and provides the serial connection with autopilot and the wireless communications with the Ground Control Station (GCS). The idea behind of this integration is to keep the low-level system which is responsible about the quadrotor stability works independently from the high-level system which handles lower priority tasks. This computer consists of 1.7GHz Quad-Core processor and 2GByte RAM and run XUbuntu 13.10 Operating System and can run Android Operating System.

The requiered software structure is built by ROS by establishing necessary nodes for accomplishing the advanced tasks. The main nodes are shown in figure 5.1(b) and their functions will be as follow



(a)



(b)

Figure 5.1: Quadrotor for high-level operations: (a) Developed quadrotor, (b) ROS structure

1. **Mavros** reads the vehicle's state from the autopilot and send RC-override commands.

2. **Opti-GPS** is in charge reading the optitrack stream and deal with the conversion process from position local coordinates to geodetic coordinates and to create the desired GPS messages in UBX protocol.

3. **UDP** handles the wireless communication between the quadrotor platform and the ground station

5.2.1 Guidance system

In a mission the quadrotor needs to perform a specific task by following a trajectory. Sometimes the quadrotor needs to maintain its position (loiter) for a period of time to collect information in the site or to wait for a decision from the user. Loiter mode maintains the quadrotor position, heading and height. And in this mode the the quadrotor positions can be controlled by sending RC command from guidance and navigation system running in onboard-pc. Loiter mode guarantees the stability in the vehicle's attitude and position. Loiter mode in ardupilot requires a GPS lock before the quadrotor can take off. Real GPS receivers can't provide a good estimation of position and velocity inside buildings like the laboratory. For this purpose we created a virtual GPS system to provide the autopilot with a geodetic positions and velocities simulated from an Optitrack system. The virtual GPS system is discussed in details in section 5.3.

A quadrotor's position is fully controlled by four RC commands which control Roll, Pitch, Yaw, and the Throttle as illustrated in Table 5.1. Each command has a range between 1000 and 2000. A desired position can be achieved by generating a suitable rate change in values of two or more of these commands.

Table 5.1: RC command

Name	Description	Range	Limitation
ROLL	Change horizonta Right-Left location	roll left max =1000. no rolling =1500. roll right max=2000.	horizontal speed, default =5m/s
PITCH	Change horizontal front-back location	pitch left max =1000. no pitching =1500. pitchl right. max=2000	horizontal speed, default =5m/s
YAW	Change heading direction	yaw CCW max =1000. zero yaw=1500. yaw CW max=2000.	rotating speed
THROTTLE	Change vertical up-down location	descend max =1000. hovering = 1500. climb max=2000.	vertical speed, default =2.5m/s

Basically the motion path of the quadrotor can be determined by defining several points on that path. The change rate in movement from point to another can be achieved simply by the change rate of the RC commands as described in (5.1)

$$\begin{cases} \frac{dp_E}{dt} = k_E(sin(\psi + \frac{\pi}{2})\frac{dRC_{roll}}{dt} + sin(\psi)\frac{dRC_{pitch}}{dt}) \\ \frac{dp_N}{dt} = k_N(cos(\psi)\frac{dRC_{pitch}}{dt} + cos(\psi + \frac{\pi}{2})\frac{dRC_{roll}}{dt}) \\ \frac{dp_{Alt}}{dt} = k_D\frac{dRC_{Throttle}}{dt} \\ Heading = \psi + \frac{dRC_{yaw}}{dt} \end{cases} \tag{5.1}$$

where K_E,K_N, and K_D are positive gains. P_E, P_N, and P_D are positons in toward north, east and down respectively. We consider in (5.1) positive RC values for rolling to the right, pitching to front, and climbing up. And negative RC values for rolling to the left, pitching to back, and descending down. In this case we have to map the RC value ranges generated from these relation to a values between 1000 and 2000 as described previously in Table 5.1

5.2.2 Flight plan in sequense of waypoints

Missions currently undertaken by unmanned aerial vehicles are predefined. The flight plan is a series of waypoints defined by latitude, longitude, and altitude based on the objectives of the mission. It is assumed that this series of waypoints to be joined by a straight line segments. Or to make the motion oth copter smother the waypoints can be joined by spline segments. Usually the first of these points starts when the vehicle is armed before take-off and it is called home-waypoint and the last point in the landing phase.

5.3 Virtual GPS for purposes of testing aerial robot in the LAB

In projects under research developments people need to conduct many experiments on these aerial platforms. The implementation of such experiments outdoor can increase the risk on people life because of undesired behaviors from these machines. Conducting such experiments in the laboratory before moving it outside will make the experimental process safer and faster. For this reason we designed a virtual GPS based on a motion-tracking system in the LAB. This system converts the positions and velocities provided by the Optitrack system in local frame into the corresponding global positions in geodetic coordinates and velocites in NED coordinates. these data then formatted in a specific binary structures using UBX protocol and provided by a serial connection to the GPS port in the autopilot board. In this work we have conducted our experiments on a quadrotor platform with a powerful linux computer mounted onboard. This computer addresses the computational cost for the implementation of complex algorithms using ROS and provides the serial connection with autopilot and the wireless communications with the ground station. The idea of designing this system is to facilitate moving by experiments from the laboratory to outside and vice versa.

5.3.1 Coordinates conversion

The body's positions from the OptiTrack system are measured in a local frame (xyz) on a flat earth position.The flat Earth coordinate system assumes the z-axis is downward positive.

$$\begin{bmatrix} N \\ E \\ D \end{bmatrix} = \begin{bmatrix} cos\psi & -sin\psi & 0 \\ sin\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{5.2}$$

where ψ is the angle in clockwise between the x-axis and north. To convert the north (N) east (E) down (D) coordinates into a geodetic latitude (Lat), longitude (Lon) and altitude (Alt) we use the following relation.

$$\begin{cases} Lat = lat_0 + dLat \\ Lon = lon_0 + dLon \\ Alt = -P_z - alt_0 \\ dLat = atan(\frac{1}{R_M})dN \\ dLon = atan(\frac{1}{R_N cosLat})dE \\ R_N = \frac{R}{\sqrt{1-(2f-f^2)sin^2 lat_0}} \\ R_M = R_N \frac{1-(2f-f^2)}{\sqrt{1-(2f-f^2)sin^2 lat_0}} \end{cases} \tag{5.3}$$

5.3.2 Wiring and connection

Recently, GPS receivers have been used widely in different ground and aerial robots. U-blox GPS receivers are provided in a low-power, high-performance chips and modules [2]different and some of them comes with a compass i.e 3DR uBlox [3] and some of them such as Zubax GNSS module includes both a compass and barometer. In this document we will focus on the 3DR uBlox module and its comunication with PIXHAWK autopilot. practically GPS receivers are connected with autopilot via a serial port. The serial consist of an RX and a TX line to recieve and transmit specific sequence of messages using a specific protocol. These serial ports operate in asynchronous mode. The baud rates can be configured individually for each serial port [42].

5.3.3 UBX protocol and Messages Specification

GPS receivers use a protocol to transmit GPS data to a host computer using serial ports. This protocol uses 8 Bit binary data checksum protected uses a low-overhead checksum algorithm. the cecksum algorthim will be defined later in the next section. the UBX protocol uses also 2-phase message identifier the class and the message ID. A basic UBX Packet looks as in figure 5.2

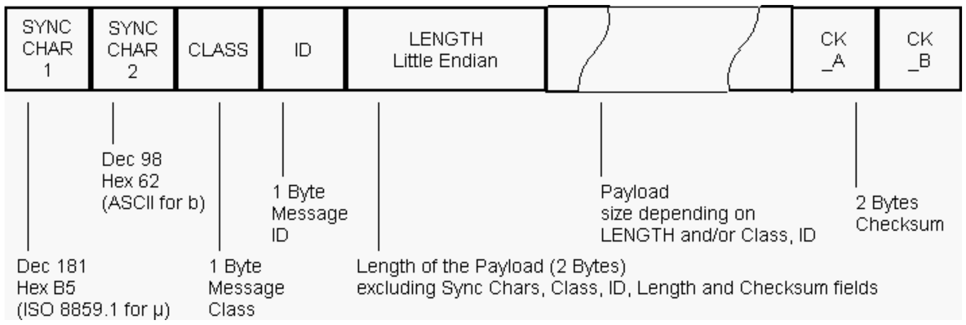

SYNC CHAR 1
SYNC CHAR 2
CLASS
ID
LENGTH Little Endian
CK _A
CK _B
Dec 98 Hex 62 (ASCII for b)
1 Byte Message ID
Payload size depending on LENGTH and/or Class, ID
2 Bytes Checksum
Dec 181 Hex B5 (ISO 8859.1 for µ)
1 Byte Message Class
Length of the Payload (2 Bytes) excluding Sync Chars, Class, ID, Length and Checksum fields

Figure 5.2: UBX Packet

In most navigation systems for autonomous robots the information from GPS provided mainly in five messages which are the NAV-POSLLH which provides the

Geodetic Position information, NAV-SOL which gives positions and velocities information in ECEF (Earth Centered, Earth Fixed) coordinates, NAV-VELNED gives the velocites and speeds in NED (North-East-Down) coordinates, VAL-STATUS provides information on the data accuracy and availability of coverage from satellites, and finally TIM-TP supply the system with time pulses. The structure of all these messages will be described in details in following subsections.

All these messages are sent sequentially so the beginning of each new message is defined by header. In UBX protocol the header contains of two bytes and it is given in hexedicemal as (0xB5 0x62) and in decimal is equal to (181 98). Each message comes under specific class which consists of a set of messages. Each message is identified through a unique ID number within the class. Both the class and the ID are defined in two bytes for example the ID for POSLLH is (0x02) and it is a message in the class NAV which is defined by (0x01) byte. Each message contains information about the time in milliseconds calculated from the beginning of every week. The time information are described in iTOW (GPS Millisecond Time of Week) with four bytes.

1. NAV-POSLLH

   Geodetic Position Solution. This message outputs the Geodetic position in the currently selected Ellipsoid. The default is the WGS84 Ellipsoid. The structre and the payload of this message are described by Table 5.2 and Table 5.3 respectively.

   Table 5.2: POSLLH message structure

Header	Class and ID	Length (in bytes)	Message Payload	Checksum
0xB5 0x62	0x01 0x02	28 = 0x1C 0x00	described below	CK_A CK_B

   The payload of POSLLH message contains the needed information about the geodetic positions and they are presented by 28 bytes in seven variables

2. NAV-SOL

   The NAV-SOL messege combines Position, velocity and time solution in ECEF (Earth Centered, Earth Fixed) . This message comes with the same class 0x01 and its ID is 0x06. It is long with a length of 52 (0x34 0x00) bytes excluding the bytes of the header, the class and the messege ID. Most of the information provided by this message are described in Table 5.4. The message also gives number of satellites available and dilution of precision

3. NAV-VELNED

   This message provides velocites, total speed (3D) and ground speed (2D) in NED coordinates. The message gives also the heading of motion in the north-east plane. each unit is structured by four bytes formats as shown in a structure example in Table 5.5

Table 5.3: POSLLH payload

Byte Offset	Number Format	Name and Description	unit	Example
0	U4	iTOW (GPS Millisecond Time of Week)	ms	-
4	I4	lon (Longitude)	deg	44492395 = (0x2C 0x7E 0x0D 0x98)
8	I4	lat (Latitude)	deg	11330145 = (0x00 0xAC 0xE2 0x61)
12	I4	height (height above elipsoid)	mm	101722 = (0x00 0x01 0x8D 0x5A)
16	I4	hMSL (height above mean sea level)	mm	56181 = (0x00 0x00 0xDB 0x75)
20	U4	hAcc (Horizontal accuracy estimate)	mm	11506 = 0x00 0x00 0x2C 0xF2
24	U4	vAcc(Vertical accuracy estimate)	mm	15435 = 0x00 0x00 0x3C 0x4B

U4 = Unsigned Long, I4= Signed Long

4. NAV-STATUS

   Information about the accuracy status of the GPS can be dominated by this message. Where is the GPS fix is determined. In Addition to some flags as accuracy indicator along with the position, time and velocity information. The GPS fix values are the same as described previously in the message NAV-SOL. So to have a 3D fix value we will read 3 as shwon in Table 5.6

   A solution is considered valid, when both PDOP and Accuracy lie below the respective limits. To qualify a position as valid the gpsFixOK flag in must be checked.

5.3.4 UBX Checksum

The checksum is calculated over the packet, starting and including the CLASS field, up until, but excluding, the Checksum Field The checksum algorithm used is the 8-Bit Fletcher Algorithm.This algorithm works as follows: Buffer N contains the data over which the checksum is to be calculated. The two

Table 5.4: SOL payload

Byte Off-set	Number Format	Name and Description	unit
0	U4	iTOW (GPS Millisecond Time of Week)	ms
4	I4	fTOW (fractional nanoseconeds remainder	ns
8	I2	week(GPS time)	-
10	U1	GPSFix	-
11	X1	flags (fix status flags)	-
12	I4	ecefX (ECEF X coordinate)	cm
16	I4	ecefY(ECEF Y coordinate)	cm
20	I4	ecefZ(ECEF Z coordinate)	cm
24	U4	pAcc(3D Position Accuracy Estimate)	cm
28	I4	ecefVX (ECEF X velocity)	cm/s
32	I4	ecefVY(ECEF Y velocity)	cm/s
36	I4	ecefVZ(ECEF Z velocity)	cm/s
40	U4	sAcc (Speed Accuracy Esti-mate)	cm/s
44	U2	pDOP - Position DOP	-
46	U1	reserved1 - Reserved	-
47	U1	numSV (Number of SVs used in Nav Solution)	-
48	U4	reserved2 - Reserved	-

U4 = Unsigned Long, I4= Signed Long

Table 5.5: VELNED message structure

Header	Class, ID	Length	time	velN	velE	velD	...
B5 62	01 12	24 00	11 C2 00 00	24 00 00 00	08 00 00 00	22 00 00 00	...
-	-	36	49681 ms	36 cm/s	8 cm/s	2 cm/s	...

...	speed	Gspeed	Heading
...	2B 00 00 00	25 00 00 00	3F E5 20 00
...	43 cm/s	37 cm/s	21.55839 deg

Table 5.6: STATUS message structure

Header	Class, ID	Length	time	gpsFix	flags	fixStat	...
B5 62	01 12	10 00	11 C2 00 00	03	05	00	...
-	-	16	49681 ms	3	15 = (00001111)	0	...

...	flags2	ttff	msss
...	00	25 00 00 00	22 01 00 00
...	0	37 s	4130 s

Chapter 6

Experimental Validation and Results

6.1 Introduction

To demonstrate real-time validation and to illustrate the distributed architecture of the proposed work, real-time implementation of Guidance, Navigation and Control (GNC) systems have been tested on a smartphone based quadrotor. All the algorithms of the GNC systems have been implemented in java using Multi-thread programming composed of smaller units of computation, known as tasks (or threads). Initialy, the estimation process were implemented as S-functions in MATLAB-simulink environment for purpose of testing the performance of sensors fusing and debugging the errors in the code before performing the flight test as well as to apply an optimization approach the estimation filters to tune precisely their gains. The performance of the attitude and position estimation of the quadrotor was evaluated indoor by a simulated and real flight tests. The attitude and position results were compared with a very precise data generated from an optitrack system. Then the experiments were expanded and moved outdoor to validate the estimation process with GPS and Barometer.

In order to evaluate the performance of autonomous flights by the quadrotor, several flight tests have been performed in different mission scenarios in indoor and outdoor environments.

6.2 Sensors Calibration

After the calibration process the results bias and scale factor for each sensitive axis are detailed in Table 6.2

The direct intigration of the accelerometer measurment before and after the calibration process are shown in figure 6.1(a). the divergence after 80 seconds of the valocity and positon from the uncalibrated measurment are (-18.23,-32.67,16.35)m/s,(-706.37,-1299.40)m respectively. 6.1(b) shows the better results of the velocity and position after calibrating the accelormenter. the divergance decresese to (-2.55,-2.35,-1.76)m/s and(-79.13,-86.7,-77.06)m respectively

Table 6.1: Accelerometer calibration results

	x	y	z
b_a (m/s^2)	-0.0242	-0.3787	-0.2344
S_a	0.0054	0.0060	0.0008

Table 6.2: Magnetometer calibration results

	x	y	z
b_m (μT)	-54.678	-47.725	-440.140
S_m	-0.0961	-0.1486	-0.1641

6.3 Autonomous flights

This section shows the performance of the developed smartphone based quadrotor in two real world application scenarios. The first flight test was coducted indoor and the position information were provided by an Optitrack system. In this experiment, the user defines the waypoints for the desired trajectory directly in the app as shown in figure 4.9. In the second experiment conducted outdoor, the user through a specific protocol, appoints a set of waypoints via Twitter account. The user can also specifies several parameters based on the mission (i.e. hovering time at the waypoint, loiter radius and maximum traveling speed). When all parameters are set and autonomous flight mode is engaged, the quadrotor starts to follow the desired trajectory autonomously and once it is achieved, the quadrotor starts descending slowly and lands. In order to obtain a reliable flying platform a manual tuning was performed for attitude and position controller gains. More precisely, attitude and position controllers gain were chosen in order to minimise oscillations and settling time. Position regulator gains are decoupled in horizontal gains, which influence x and y axis position, and vertical gain for the z axes. This distinction comes directly from the control structure itself. In the horizontal plane, the position control system computes the quaternion necessary to achieve the desired position, while the altitude controller defines the thrust request for reaching the specified vertical position.

6.3.1 Indoor flight test

In this experiment the quadrotor follows the 3D trajectory generated from the assigned waypoints. Figure 6.2 shows the vehicle's positions during this experiment proving the overall stability in flight of the vehicle with an error in position bounded in 0.2 m for the vertical axis and 0.4 m for the horizontal plane. It should be noticed

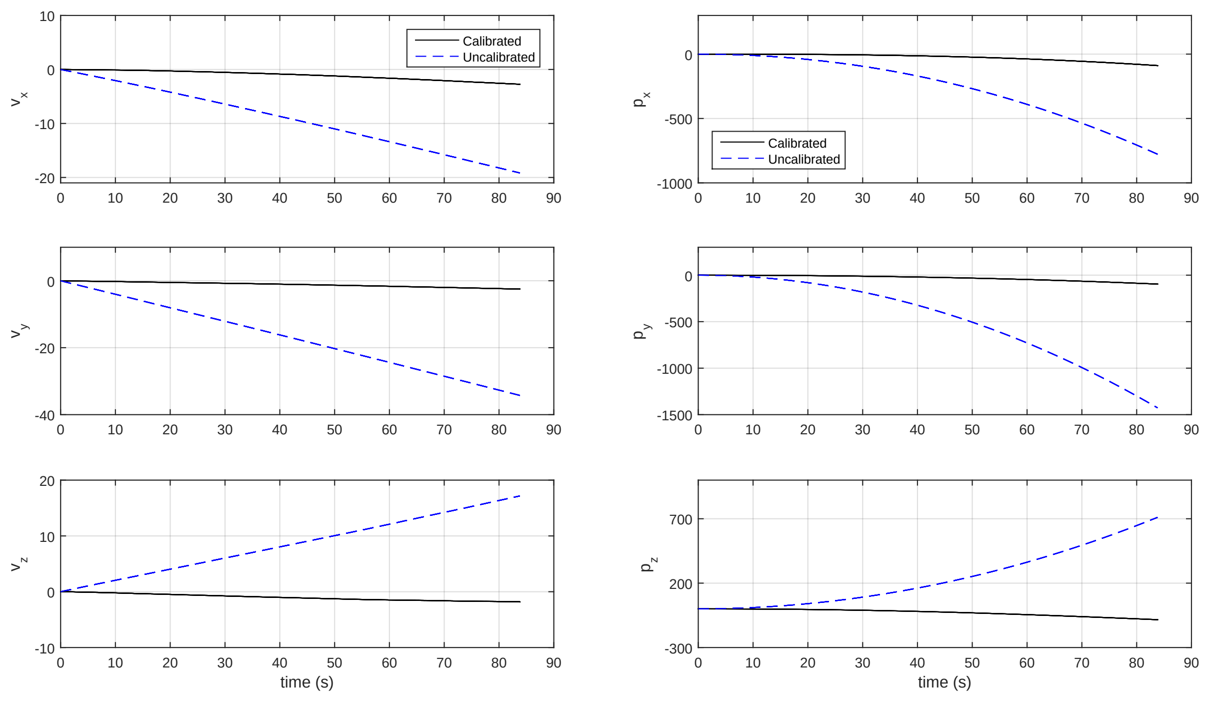

(a) (b)

Figure 6.1: Direct integration of calibrated and uncalibrated acceleration measurement for non moving IMU: (a) Velocity (m/s), (b) Position (m)

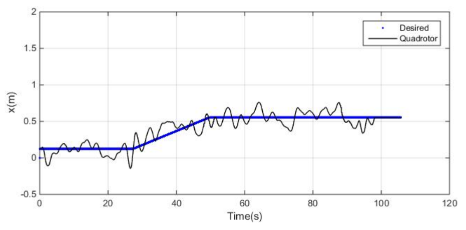
that the final variation observed in the figure for the position setpoint is due to the land procedure.

6.3.2 Outdoor mission

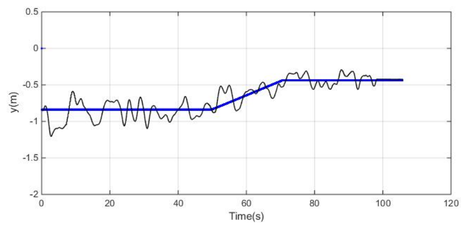
This part demonstrates the ability of the control and navigation system to track the generated trajectory and to achieve the desired waypoint navigation as well as to perform automatic take-off, hovering, and landing. In this test, a set of sequenced waypoints were defined by inserting their locations by a remote operator via Twitter account. The quadrotor should then, pass the assigned waypoints in a given sequence.

First, wee need to show the performance of the quadrotor in a hovering test. Figure 6.3 illustrates the flight trajectory at a zero position reference (i.e. $(x,\ y) = (0,\ 0)$. It can be shown from the figure that hovering control within about 1.5 m of accuracy has been achieved.

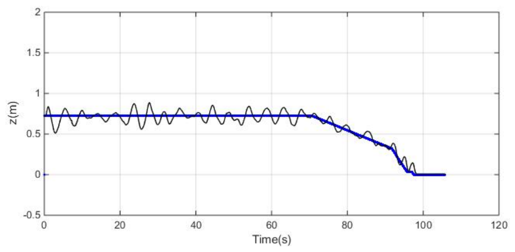
Figure 6.4 shows the position autopilot performances during an autonomous flight experiment outdoor. The experimental results, shown in the figure demonstrate good tracking of the 3D reference command. The vehicle proves quite good characteristics in terms of ability to track the predefined trajectory and effective and automatic take-off and landing.



(a)



(b)



(c)

Figure 6.2: Trajectory tracking (indoor flight test): (a) X-axis (b) Y-axis (c) Z-axis

6.4 Position and attitude estimations

This section presents the results of the proposed navigation system (see in chapter 3) obtained by implementing the nonlinear complementary filters in the android-autopilot system with the experimental data collected on-board. In order to test the attitude and position estimation performances in most flight trajectories, simulated real flight tests were performed in-door with the precence of optitrack system or out-door environment as described in chapter 4. The raw data acquired from smartphone sensors and *OptiTrack*, and the estimations provided by the algorithms described in chapter 3 and the basic sensor fusion approach provided through the `getRotationMatrix` function of *Android* were logged in order to generate the plots introduced in the next sub-sections.

6.4.1 Indoor flight tests

This experiment carried out in-door by moving by hand in the three-dimensional space the quadrotor in order to simulate its movements when flying in an in-door environment. In this section the mean square errors defined inchapter 3 is taken as figure of merit to discuss the performances of the proposed solution. In particular, a comparison is shown between the attitude estimation algorithm here described and the *Android* `getRotationMatrix` taking the *OptiTrack* data as reference. More precisely, figure 6.5(a) shows vehicle attitude estimated by the three sources and in figure 6.5(b) is displayed the error of the proposed attitude estimation algorithm with an mean square error as low as 0.1771 square degrees and the standard *Android* `getRotationMatrix` algorithm characterised by a mean square error of 16.13 square degrees with respect to *OptiTrack* estimation. From these figures it should be

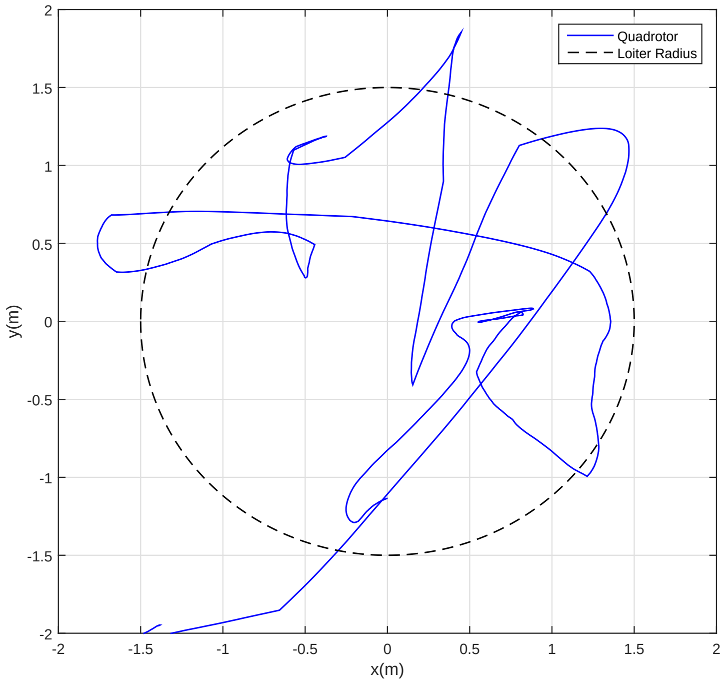


Figure 6.3: Hovering results.

noticed that the *Android* estimator is prone to high accelerations that are not noticeable in the reference source. This probably happens cause a not perfect tuning of the complementary filter on which it is based. In fact, the *Android* algorithm is not preliminarly tuned for the specific smartphone or tablet on which it is installed. Differently, the algorithm here proposed keeps very close to the *Optitrack* estimation. It is also demonstrated the reliability of the position estimation algorithm considering the reference data provided by the *OptiTrack* system. Finally, figure 6.6(a) shows very good result for what concern position estimation with the errors shown in figure 6.6(b) for the three inertial axis and a mean square error of 0.1890.

6.4.2 Outdoor flight tests

In outdoor flight tests, no truth reference signal is provided because of actual difficulties in outdoor measurements.The attitude filter is already well tuned in in-door experiments and no need to be tuned for out-door tests. However the tuning adopted in-door on the position filter, manual retuning of the filter parameters is needed in order to achieve better practical performance outdoor in the presence of GPS. The attitude estimation results, presented in figure 6.7 based on tha quadrotor movements in lateral direction while keeping the heading tho the north (i.e. yaw angle around zero). The pitch (θ) and roll (ϕ) oscillate around the mean values of 1.2^o and 1.1^o, respectively. This offset in angles is related mainly to the small inclinations of

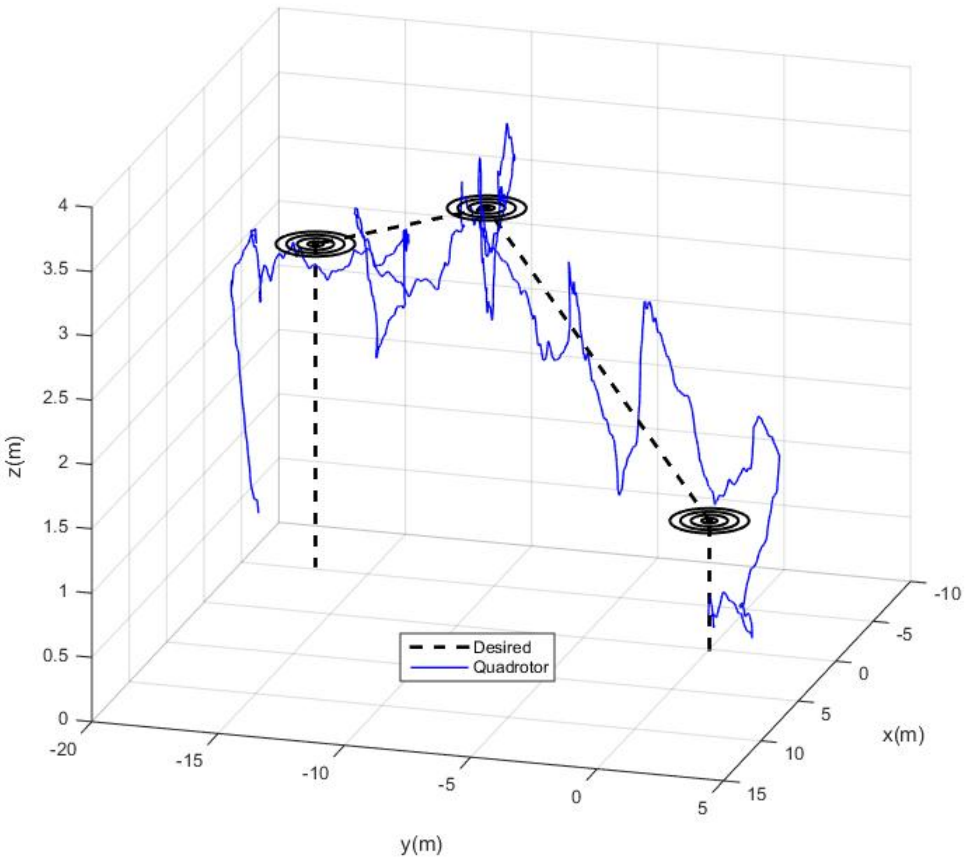
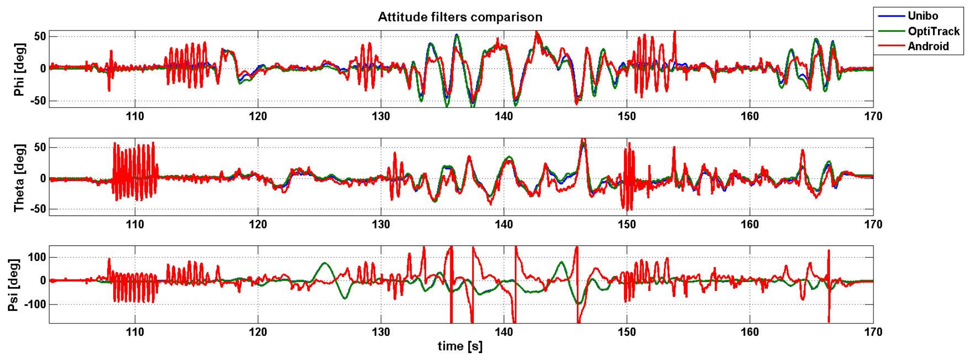


Figure 6.4: 3D desired and quadrotor trajectories in outdoor test.

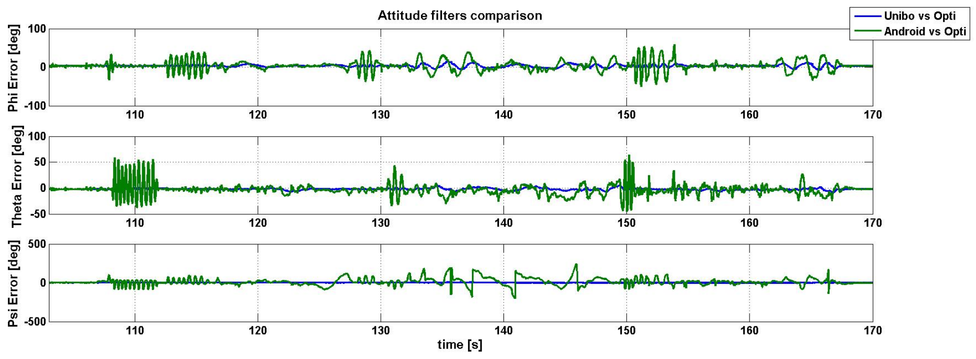
the mounted base during the installation of the hardware architecture. The angles fluctuations occur due to the quadrotor turnings during the lateral motion to achieve the desired positions. In addition to small fluctuations happen because of uncertainties and external disturbances such as interference of wind, vibration due to the rotors, asymmetry in the structure. In figure 6.7(a) larger oscillations are confirmed when the quadrotor move faster in y (East-West) direction where larger peak to peak values are verified. The heading of the quadrotor oscillates around the mean value -3.2^o, that coresopend to the soft iron magnetic effect.

The velocity and position estimation results are shown in figure 6.8 and figure 6.9 respectively. The position estimation in the proposed filter depends on the attitude kinematics, thus the the filter does not estimate explicitly the velocity and the position where the body accelerations are transformed to inertial frame and fused with the position information obtained by GPS and barometer. Figure 6.8 shows the estimate of the linear velocity in lateral direction (i.e. x-axis and y-axis), expressed in inertial frame.

The position estimation results are compared with GPS measurements for lateral position and with barometer measurements for vertical position as shown in figure



(a)



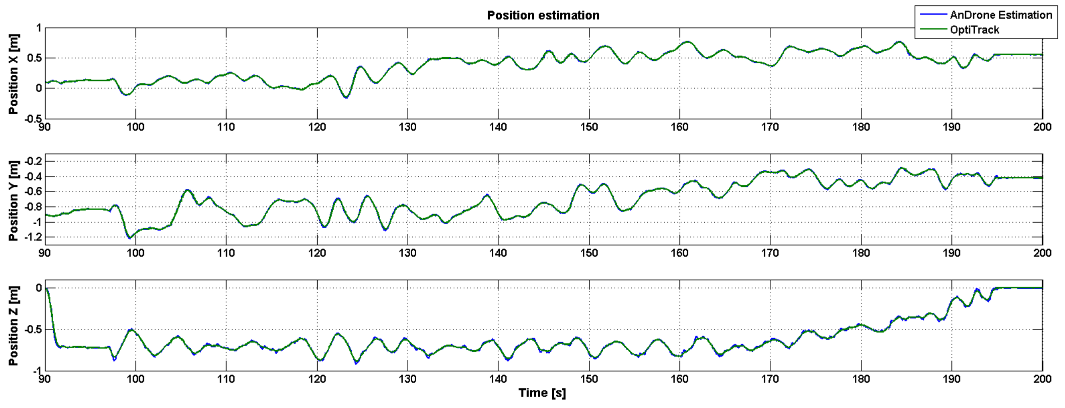
(b)

Figure 6.5: results of the experimental tests on attitude estimation: (a) comparison between proposed algorithm (UNIBO), *OptiTrack* (Optitrack) and `getRotationMatrix` (Android) attitude estimation, (b) visualisation of estimation differences for proposed algorithm vs *OptiTrack* (Unibo vs Opti) and `getRotationMatrix` vs *OptiTrack* (Android vs Opti)

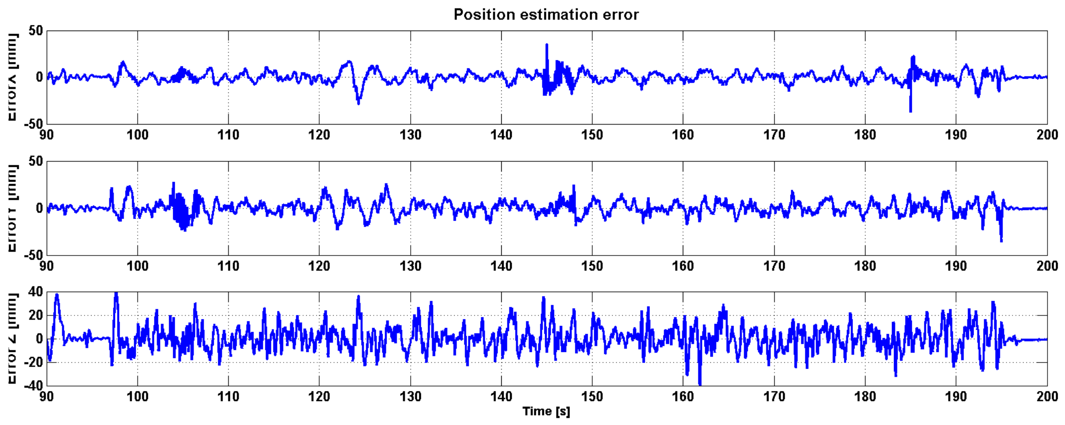
6.9. The position filter gives reliable estimation and the values always coherent with GPS measurements.

6.5 CPU analysis

In order to maximize performances, the controller gains were chosen to minimize the issues related with the fact that Android is not a RTOS [45]. With a non-RTOS, the computation time can vary and sensor data is provided at an inconsistent intervals which can adversely affect the work of the autopilot. However, the autopilot can control the aircraft properly due to the high performance capability of smartphones. The overload on CPU can influence the performance of the autopilot app. For that reason, the third party application, *CPU Monitor Advanced Lite* was used to estimate the smartphone performance while the quadrotor application was working during a real flight of th quadrotor. 6.10(a) illustrate the average and the maximum value of the CPU load, the used memory, and the network usage, while figure 6.10(b) shows



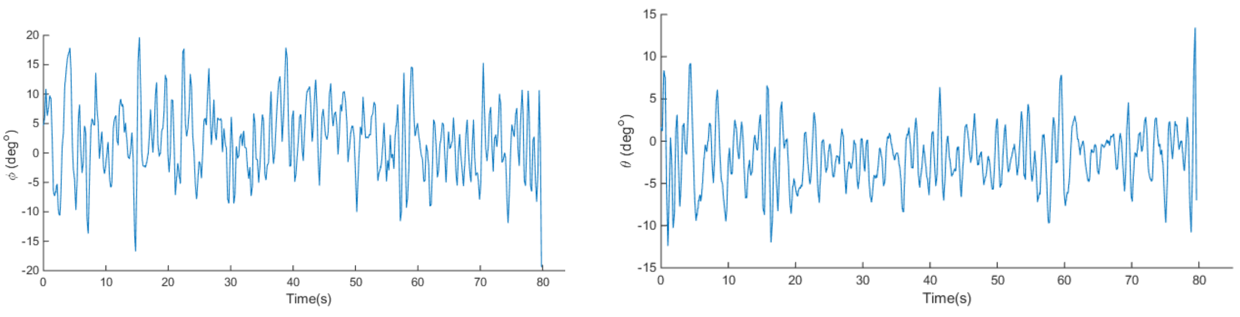
(a)



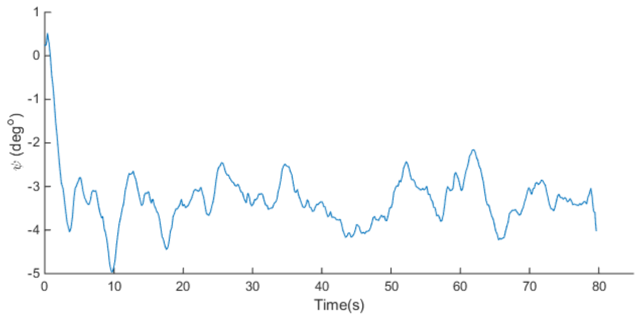
(b)

Figure 6.6: results of the experimental tests on position estimation: (a) comparison between proposed algorithm (Androne Estimation) and *OptiTrack* (Optitrack) position estimation, (b) visualisation of estimation differences for proposed algorithm vs *OptiTrack* (Unibo vs Opti) and `Android` vs *OptiTrack* (Android vs Opti)

the CPU percentage used for running the quadrotor's autopilot itself with an average of 54% and maximum value of 60% .

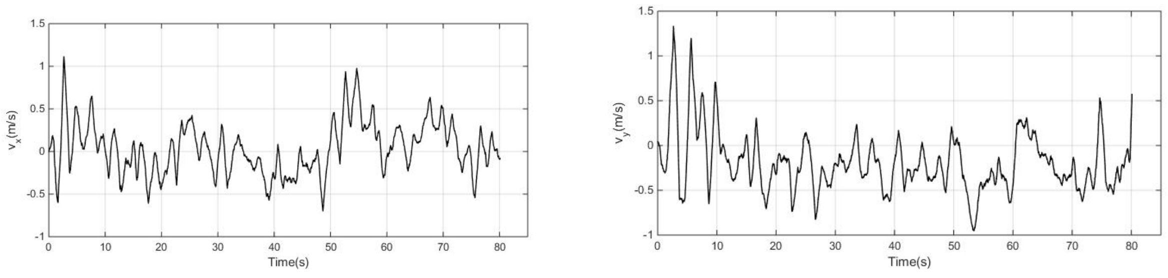


(a) (b)

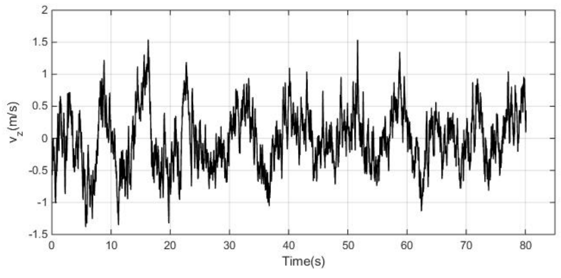


(c)

Figure 6.7: Estimated attitude angles in outdoor flight test: (a) ϕ (^{o}), (b) θ (^{o}), (c) ψ (^{o})

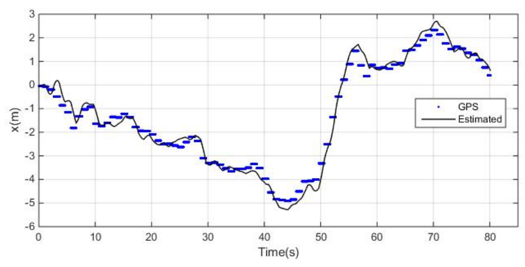


(a) (b)

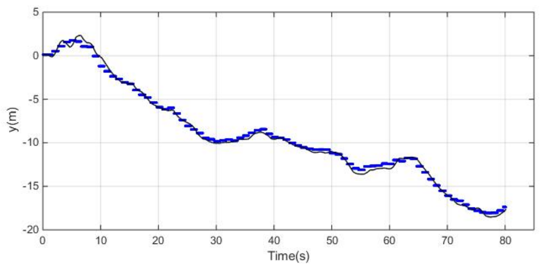


(c)

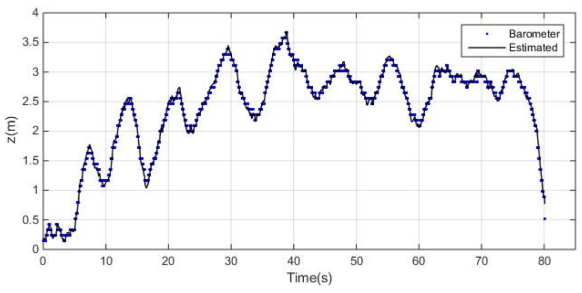
Figure 6.8: Estimated linear velocity (outdoor test): (a) v_x (m/s), (b) v_y (m/s), (c) v_y (m/s). The positive axes are directed toward North, East, and Up respectively



(a)

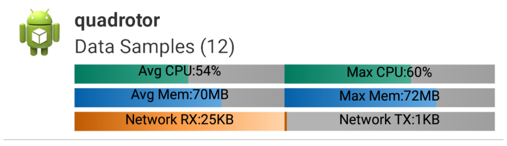


(b)



(c)

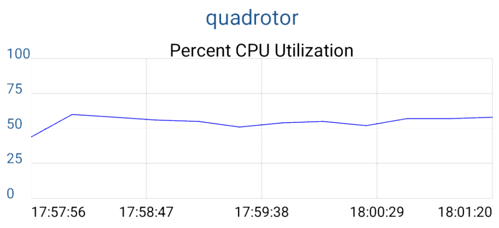
Figure 6.9: Estimated attitude angles in outdoor flight test: (a) x (North) (m), (b) y (East) (m), (c) z (Down) (m). The positive axes are directed toward North, East, and Up respectively



quadrotor
Data Samples (12)

Avg CPU:54% | Max CPU:60%
Avg Mem:70MB | Max Mem:72MB
Network RX:25KB | Network TX:1KB

(a)



(b)

Figure 6.10: Smartphone performance estimated by the app *CPU Monitor Advanced Lite*: (a) General performance of the quadrotor application, (b) CPU utilization of the quadrotor application

Chapter 7

Conclusions and recommendations

This work presented nonlinear state estimation and control of a UAV quadrotor with autonomous flight capabilities that can be used in several applications. This research dealtwith both theoretical and practical issues to explore the viability of using smartphones as flight controller for quadrotor UAVss. More precisely, it presented an effective and reliable approach to integrate noisy measurements from multiple sensors onboard. This approach provides high rate state estimates in real time allows for autonomous flight. Using quaternion, the attitude filter compensates for rate gyro bias and it achieves stability for trajectories in most configurations. The position filter estimates velocity and position in inertial earth frame and compensates for accelerometer bias. A complete navigation system was produced by integrating both the attitude and the position filters. The integration of the filtering approach based primarily on the ease of design and computational load. Furthermore, the structure of the filtering design allow for straightforward implementation without a need of high performance signal processing. The proposed filters in this work can be tuned totally independent of each other and the adopted gains are computed offline using an auxiliary design system to achieve robustness in most flight conditions. Furthermore, this work proposed a nonlinear controller for stability and trajectory tracking that is practical for real-time implementation, and also demonstrated the ability of a supervisory controller to provide effective waypoint navigation capabilities to the smartphone based quadrotor.

An implementation of the full framework on an experimental smartphone based quadrotor system has been presented. The implemented software and hardware qualifies the simultaneous performance of several demanding executions. Furthermore, the presented autopilot implements an innovative supervisory control mode for the vehicle based on a *Twitter* client. This client receives operator instructions from the cloud and translates them into commands for the quadrotor control system. From the experimental flight tests, the developed structure prove quite good performances in hovering and autonomous flight which leads towards developing a network of UAVs capable of achieving missions in real-world applications.

For future developments, It will be useful to expand the capabilities of the guidance system by integrating high-level systems that help in decisions making and planning. In this work, we were able to save real-time video during the flight from the onboard camera. This can lead to algorithms implementation for mapping and localization and for obstacle avoidance. Moreover, the communication capabilities in smartphones can be easily established to allow integration with other UAVs to increase guidance capabilities. Thus, it might be useful in many applications to develop a network containing different aerial and ground robots based on smartphones.

Bibliography

[1] DJI website. Accessed on the 11th of February 2015.

[2] 3dr robotics. http://www.u-blox.com/, 2015.

[3] 3dr robotics. `https://store.3drobotics.com/products/3dr-gps-ublox-with-compass`, 2015.

[4] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart. Onboard imu and monocular vision based control for mavs in unknown in- and outdoor environments. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3056–3063, May 2011.

[5] P. Aggarwal, Z. Syed, X. Niu, and N. El-Sheimy. A standard testing and calibration procedure for low cost mems inertial sensors and units. *Journal of Navigation*, 61:323–336, 4 2008.

[6] F. Aghili and A. Salerno. Driftless 3-d attitude determination and positioning of mobile robots by integration of imu with two rtk gpss. *Mechatronics, IEEE/ASME Transactions on*, 18(1):21–31, Feb 2013.

[7] L. Aldrovandi, M. Hayajneh, M. Melega, M. Furci, R. Naldi, and L. Marconi. A smartphone based quadrotor: Attitude and position estimation. In *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pages 1251–1259, June 2015.

[8] A Bachrach, A de Winter, Ruijie He, G. Hemann, S. Prentice, and N. Roy. Range - robust autonomous navigation in gps-denied environments. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1096–1097, May 2010.

[9] Ytai Ben-Tsvi. Ioio documentation. GitHub website. Accessed on the 23nd of December 2014.

[10] August Bjalemark and Hannes Bergkvist. Quadcopter control using android based sensing. *Recent Researches in Electrical Engineering*, 2014.

[11] Luis Rodolfo García Carrillo, Alejandro Enrique Dzul López, Rogelio Lozano, and Claude Pégard. *Quad rotorcraft control: vision-based hovering and navigation*. Springer Science & Business Media, 2012.

[12] Pedro Castillo, Rogelio Lozano, and Alejandro Dzul. Stabilization of a mini rotorcraft with four rotors. *IEEE control systems magazine*, 25(6):45–55, 2005.

[13] Roger A. Chadwick. The impacts of multiple robots and display views: An urban search and rescue simulation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 49(3):387–391, 2005.

[14] Seunghwan Choi, Kijung Kim, Yunki Kim, and Jangmyung Lee. Outdoor precision position estimation system using multi-gps receivers. In *Intelligent Robotics and Applications*, pages 66–76. Springer, 2013.

[15] C. Coopmans, A.M. Jensen, and YangQuan Chen. Fractional-order complementary filters for small unmanned aerial system navigation. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 754–760, May 2013.

[16] Jin Qiang Cui, Shupeng Lai, Xiangxu Dong, Peidong Liu, B.M. Chen, and T.H. Lee. Autonomous navigation of uav in forest. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 726–733, May 2014.

[17] Xilun Ding and Yushu Yu. Motion planning and stabilization control of a multipropeller multifunction aerial robot. *Mechatronics, IEEE/ASME Transactions on*, 18(2):645–656, April 2013.

[18] Chuong B Do. The multivariate gaussian distribution, 2008.

[19] Dualsky website. Xm400es, 2003. Accessed on the 23$^{\text{nd}}$ of December 2014.

[20] Google. Google Nexus 5. Nexus website. Accessed on the 23$^{\text{nd}}$ of December 2014.

[21] José Fermi Guerrero-Castellanos, Heberto Madrigal-Sastre, Sylvain Durand, Lizeth Torres, and German Ardul Munoz-Hernández. A robust nonlinear observer for real-time attitude estimation using low-cost mems inertial sensors. *Sensors*, 13(11):15138–15158, 2013.

[22] S. Gupte, P.IT. Mohandas, and J.M. Conrad. A survey of quadrotor unmanned aerial vehicles. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–6, March 2012.

[23] Sinpyo Hong, Ju Yong Choi, Chang Sup Kim, Man Hyung Lee, and J.L. Speyer. Estimation of errors in ins with multiple gps antennas. In *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, volume 1, pages 410–415 vol.1, 2001.

[24] Minh-Duc Hua. Attitude estimation for accelerated vehicles using gps/ins measurements. *Control Engineering Practice*, 18(7):723 – 732, 2010. Special Issue on Aerial Robotics.

[25] Myungsoo Jun, S.I. Roumeliotis, and G. Sukhatme. State estimation of an autonomous helicopter using kalman filtering. In *Intelligent Robots and Systems, 1999. IROS '99. Proceedings. 1999 IEEE/RSJ International Conference on*, volume 3, pages 1346–1353 vol.3, 1999.

[26] Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378, 2012.

[27] BC Konvalin. Compensating for tilt, hard-iron and soft-iron effects. *Sens. Mag*, pages 1–11, 2009.

[28] Landing Products. Advanced precision composites propellers website, 2007. Accessed on the 23nd of December 2014.

[29] J. Lee. Global positioning/gps. In Rob KitchinNigel Thrift, editor, *International Encyclopedia of Human Geography*, pages 548 – 555. Elsevier, Oxford, 2009.

[30] Taeyoung Lee, M. Leoky, and N.H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *Decision and Control (CDC), 2010 49th IEEE Conference on*, pages 5420–5425, Dec 2010.

[31] Michael Leichtfried, Christoph Kaltenriner, Annette Mossel, and Hannes Kaufmann. Autonomous flight using a smartphone as on-board processing unit in gps-denied environments. In *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, page 341. ACM, 2013.

[32] S. Leutenegger, A. Melzer, K. Alexis, and R. Siegwart. Robust state estimation for small unmanned airplanes. In *Control Applications (CCA), 2014 IEEE Conference on*, pages 1003–1010, Oct 2014.

[33] Yibo Li and Shuxi Song. A survey of control algorithms for quadrotor unmanned helicopter. In *Advanced Computational Intelligence (ICACI), 2012 IEEE Fifth International Conference on*, pages 365–369, Oct 2012.

[34] Hyon Lim, Jaemann Park, Daewon Lee, and H.J. Kim. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. *Robotics Automation Magazine, IEEE*, 19(3):33–45, Sept 2012.

[35] Xiaojie Liu, Xiaohui Zhao, Haijun Gu, and A. Sanchez. Application and design of real-time control system for the quad-rotor helicopter. In *Measuring Technology and Mechatronics Automation, 2009. ICMTMA '09. International Conference on*, volume 2, pages 40–44, April 2009.

[36] G. Loianno, G. Cross, Chao Qu, Y. Mulgaonkar, J.A. Hesch, and V. Kumar. Flying smartphones: Automated flight enabled by consumer electronics. *Robotics Automation Magazine, IEEE*, 22(2):24–32, June 2015.

[37] R. Mahony, T. Hamel, and Jean-Michel Pflimlin. Nonlinear complementary filters on the special orthogonal group. *Automatic Control, IEEE Transactions on*, 53(5):1203–1218, June 2008.

[38] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2992–2997, May 2011.

[39] Y. Naidoo, R. Stopforth, and G. Bright. Development of an uav for search amp; rescue applications. In *AFRICON, 2011*, pages 1–6, Sept 2011.

[40] Yogianandh Naidoo, Riaan Stopforth, and Glen Bright. Development of an uav for search & rescue applications. In *AFRICON, 2011*, pages 1–6. IEEE, 2011.

[41] Roberto Naldi, Michele Furci, Ricardo G Sanfelice, and Lorenzo Marconi. Global trajectory tracking for underactuated vtol aerial vehicles using a cascade control paradigm. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 4212–4217. IEEE, 2013.

[42] UBX NMEA. Receiver description including protocol specification, u-blox 6 gnss receiver. *Public Release*.

[43] Kenzo Nonami, Farid Kendoul, Satoshi Suzuki, Wei Wang, and Daisuke Nakazawa. *Autonomous Flying Robots: Unmanned Aerial Vehicles and Micro Aerial Vehicles*. Springer Science & Business Media, 2010.

[44] S. Omari, Minh-Duc Hua, G. Ducard, and T. Hamel. Hardware and software architecture for nonlinear control of multirotor helicopters. *Mechatronics, IEEE/ASME Transactions on*, 18(6):1724–1736, Dec 2013.

[45] L. Perneel, H. Fayyad-Kazan, and M. Timmerman. Can android be used for real-time purposes? In *Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on*, pages 1–6, Dec 2012.

[46] Xin Qi, Shi Zhongke, and Zhu HongYu. Dual estimation of attitude and parameters considering vibration based on gps and imu. In *Control Conference (ASCC), 2013 9th Asian*, pages 1–6, June 2013.

[47] J.M. Roberts, P.I. Corke, and G. Buskey. Low-cost flight control system for a small autonomous helicopter. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 546–551 vol.1, Sept 2003.

[48] Benjamin Saage, Darius Morawiec, Matthias Nagel, Robin Vinzenz, and Tobias Wetzel. *Quadroid website*. `http://www.quadroid.io/?lang=en`, 2014. Accessed on the 28$^{\text{nd}}$ of December 2014.

[49] Jürgen Scherer, Saeed Yahyanejad, Samira Hayat, Evsen Yanmaz, Torsten Andre, Asif Khan, Vladimir Vukadinovic, Christian Bettstetter, Hermann Hellwagner, and Bernhard Rinner. An autonomous multi-uav system for search and rescue. *ARW 2015*, page 32, 2015.

[50] Yasmina Bestaoui Sebbane. *Planning and Decision Making for Aerial Robots*. Springer, 2014.

[51] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Multi-sensor fusion for robust autonmous flight in indoor and outdoor environments with a rotor craft mav. In *2014 International Conference on Robotics and Automation (ICRA14), Hongkong, China*, 2014.

[52] M. Tailanian, S. Paternain, R. Rosa, and R. Canetti. Design and implementation of sensor data fusion for an autonomous quadrotor. In *Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 2014 IEEE International*, pages 1431–1436, May 2014.

[53] L. Tamas, G. Lazea, R. Robotin, C. Marcu, S. Herle, and Z. Szekely. State estimation based on kalman filtering techniques in navigation. In *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, volume 2, pages 147–152, May 2008.

[54] T. Tomic, K. Schmid, P. Lutz, A Domel, M. Kassecker, E. Mair, IL. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *Robotics Automation Magazine, IEEE*, 19(3):46–56, Sept 2012.

[55] T. Tomic, K. Schmid, P. Lutz, A Domel, M. Kassecker, E. Mair, IL. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *Robotics Automation Magazine, IEEE*, 19(3):46–56, Sept 2012.

[56] R.G. Valenti, I. Dryanovski, C. Jaramillo, D. Perea Strom, and Jizhong Xiao. Autonomous quadrotor flight using onboard rgb-d visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5233–5238, May 2014.

[57] Rudolph Van Der Merwe and Eric A Wan. Sigma-point kalman filters for integrated navigation. In *Proceedings of the 60th Annual Meeting of the Institute of Navigation (ION)*, pages 641–654, 2004.

[58] Renato Ventura and Pedro U Lima. Search and rescue robots: The civil protection teams of the future. In *Emerging Security Technologies (EST), 2012 Third International Conference on*, pages 12–19. IEEE, 2012.

[59] B. Vik and T.I. Fossen. A nonlinear observer for gps and ins integration. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 3, pages 2956–2961 vol.3, 2001.

[60] S. Waharte and N. Trigoni. Supporting search and rescue operations with uavs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147, Sept 2010.

[61] Xu Wanli, Bao Shuo, and Liu Zhun. The state estimation of uav based on ukf. In *Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on*, pages 402–405, Sept 2014.

[62] Shaowu Yang, S.A. Scherer, and A. Zell. Visual slam for autonomous mavs with dual cameras. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5227–5232, May 2014.

List of abbreviations

The following list gives a short description of the Acronyms used throughout this thesis

Abbreviation	**Description**
API	Application Programming Interface
ARVA	Appareil de Recherche de Victimes en Avalanche
BLDC	BrushLess Direct Current
CoM	Center of Mass
CPU	Central Processing Unit
DCM	Direction Cosine Matrix
ECF	Explicit Complementary Filter
EKF	Extended Kalman Filter
ESC	Electronic Speed Control
GCS	Ground Control Station
GPS	Global Positioning System
GPU	Graphics Processing Unit
GNC	Guidance, Navigation and Control
GNSS	Global Navigation Satellite Systems
IMU	Inertial Measurement Unit
Li-Po	Lithium Polymer
LSM	Least Square Method
LAN	Local Area Network
LOS	Line of the Sight
MEMS	Micro-Electro-Mechanical System
NED	North-East-Down
PID	Proportional-Integral-Derivative
PSD	Power Spectral Density
PWM	Pulse-Width Modulation
RGB-D	Red, Green, Blue and Depth
ROS	Robot Operating System
RTOS	Real-Time Operating System
SAR	Search And Rescue
SPS	Standard Positioning Service

UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UDP	User Datagram Protocol
UKF	Unscented Kalman Filter