Politecnico di Torino
Dipartimento di Ingegneria Meccanica e Aerospaziale

Corso di Dottorato in Meccatronica
SDSS: ING-INF/01-Elettronica

PhD Thesis

Use of Unmanned Aerial Systems in Civil Applications

Candidate:
Antonio Toma

Advisor:
Marcello Chiaberge

XXVII ciclo

Nam et plerique nobilium Græcorum et Favorinus philosophus, memoriarum veterum exsequentissimus, affirmatissime scripserunt simulacrum columbæe ligno ab Archyta ratione quadam disciplinaque mechanica factum volasse; ita erat scilicet libramentis suspensum et aura spiritus inclusa atque occulta concitum.

Non solo numerosi autori greci di gran nome ma anche il filosofo Favorino, scrupolosissimo indagatore di antichità, hanno testimoniato con assoluta sicurezza che Archita fabbricò, in base a certi principi di ingegneria, un oggetto di legno in forma di colomba, e questa colomba volò; è evidente che essa era accuratamente equilbrata mediante contrappesi e celava al suo interno dei fiotti d'aria che le conferivano il moto.

— Gellio X 12 9-10 (Traduzione Italiana G. Bernardi Perini)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

API Application Programming Interface

ASL Autonomous Systems Lab

CMOS Complementary Metal-Oxide Semiconductor

COA Certificate of Authorization

DIMES Descent Image Motion Estimation System

DOP Dilution of Precision

ECEF Earth-Centered, Earth-Fixed

EDL Entry Descent Landing

EKF Extended Kalman Filter

ENAC Ente Nazionale Aviazione Civile

ETHZ Eidgenössische Technische Hochschule Zürich

FAA Federal Aviation Administration

F4SC FlyForSmartCity

3G 3rd Generation

4G 4th Generation

GCS Ground Control Station

GDOP Geometric Dilution of Precision

GPS Global Positioning System

GUI Graphical User Interface

HDOP Horizontal Dilution of Precision

HTML5 HyperText Markup Language 5

IMU Inertial Measurement Unit

JPL Jet Propulsion Laboratory

JSON JavaScript Object Notation

LAN Local Area Network

LIM Laboratorio Interdisciplinare di Meccatronica

LiPo LIthium POlymer

LTE Long Term Evolution

MAV Micro Aerial Vehicle

MER Mars Exploration Rovers

MK	MikroKopter
MP	MicroPilot
MSF	Multi Sensor Fusion
NASA	National Aeronautics and Space Administration
NRS	Network Robot System
NSD	Noise Spectral Density
ODOMI	Open Data Oriented MIssion Planner
PaaS	Platform-as-a-Service
PID	Proportional Integral Derivative
PoE	Power over Ethernet
PSD	Power Spectral Density
REST	REpresentational State Transfer
ROCON	RObotics in CONcert
ROS	Robotic Operating System
RPA	Remotely Piloted Aircraft
RPG	Robotics and Perception Group
RSS	Root of Sum of Squares
RSSI	Received Signal Strength Indication
RTK	Real Time Kinematics
RTOS	Real Time Operating System
SDK	Software Development Kit
SSF	Single Sensor Fusion
STEPS	Sistemi e TEcnologie per l'esPlorazione Spaziale
SVO	Semi-direct Visual Odometry
TAS-I	Thales Alenia Space-Italy
UART	Universal Asynchronous Receiver-Transmitter
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
UERE	User Equivalent Range Error
UZH	Universität Zürich
VDOP	Vertical Dilution of Precision
VO	Visual Odometry

VPN	Virtual Private Network
VTNF	Vision-based Terrain Navigation Facility
VTOL	Vertical Take Off and Landing
WGN	White Gaussian Noise

ABSTRACT

Interest in *drones* has been exponentially growing in the last ten years and these machines are often presented as the optimal solution in a huge number of civil applications (monitoring, agriculture, emergency management etc). However the promises still do not match the data coming from the consumer market, suggesting that the only big field in which the use of small unmanned aerial vehicles is actually profitable is the video-makers' one. This may be explained partly with the strong limits imposed by existing (and often "obsolete") national regulations, but also - and pheraps mainly - with the lack of real autonomy. The vast majority of vehicles on the market nowadays are infact autonomous only in the sense that they are able to follow a pre-determined list of latitude-longitude-altitude coordinates. The aim of this thesis is to demonstrate that complete autonomy for UAVs can be achieved only with a performing control, reliable and flexible planning platforms and strong perception capabilities; these topics are introduced and discussed by presenting the results of the main research activities performed by the candidate in the last three years which have resulted in 1) the design, integration and control of a test bed for validating and benchmarking visual-based algorithm for space applications; 2) the implementation of a cloud-based platform for multi-agent mission planning; 3) the on-board use of a multi-sensor fusion framework based on an Extended Kalman Filter architecture.

RINGRAZIAMENTI

This dissertation would not have been possible without the contribute and the support of many people. In particular I would firstly like to thank my supervisor, Prof. Marcello Chiaberge, for guiding me in my PhD studies and all the staff at LIM Mechatronics Lab. Thanks especially to the "UAV team" - Gianluca, Mario, Claudio - and also to Max, for his valuable help.

I would like to thank Carlo Paccagnini, who taught me a lot, Prof. Andrea Tonoli, Roberto Antonini, Mario Nigra and Marco Gaspardone, your collaboration and support was really precious.

I will forever be thankful to Prof. Davide Scaramuzza, Matthias, Flavio, Elias and all the incredible UZH's Robotics and Perception Group. Being part of your team for six months truly was one of the most important and formative experiences of my life.

I am grateful to Gabriele Ermacora, Stefano Rosa and Mario Cassaro, without you my PhD years would have been way more boring; I could not have asked for better friends when I started all this. Thanks guys!

Turin, april 2015 Antonio Toma

INTRODUCTION

Civil UAVs are considered by many analysts as one of the most promising technologies of the last ten years. Their popularity is continuously growing thanks to new available technologies, scientific research, internet communities of enthusiasts and industries. However, from a business point of view, almost the totality of revenue opportunities from the use of civil UAVs at the moment are in the photographers/video-makers market and in the toy/hi-tech gadgets one. This apparent contradiction between expected vs. actual results is likely due to two main reasons: 1) strong limits imposed by existing (and often "obsolete") regulations and 2) lack of *real* autonomy, i.e. the ability to take autonomous decisions. Until now, in fact, almost the totality of the UAV solutions on the market are autonomous in the sense that they can follow a pre-defined sequence of GPS coordinates, at given altitude, also being able sometimes to take-off and land automatically. This approach does not make the vehicles less blind than they already are and, in any way, help them in overcoming some typical problems in robotics, such as obstacless avoidance, multi-agent coordination, self-localization in unstructured environments (without any aid from global positioning systems of course) and mapping. The vehicle is not *that* autonomous after all.

This dissertation studies the issues of *control*, *planning* and *perception* for small UAVs, three fundamentals topics in the achievement of complete autonomy in civil operations. In particular:

THE FIRST CHAPTER offers an overview of the technologies involved in the described activities, briefly reviewing the four different types of autopilot boards employed;

THE SECOND CHAPTER focuses on *control* and describes the modelling and control activities of a UAV in an optically tracked facility, in the field of space research;

THE THIRD CHAPTER focuses on *planning* and describes an cloud-based infrastructure for multi-agent missions management in smart city scenarios.

THE FOURTH CHAPTER focuses on *perception* and describes the (preliminary) results obtained from the fusion of several on-board sensors using a state-of-the-art software framework;

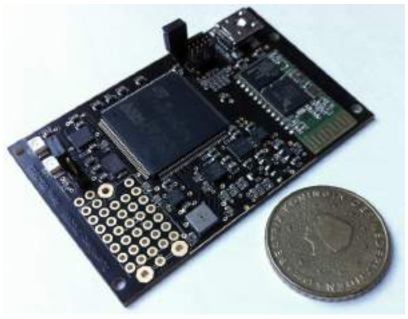
THE LAST CHAPTER closes the dissertation discussing its results and presenting the future expected activities and some concluding comments on the work.

1 THE RISE OF CIVIL *DRONES*

In the last ten years the world of civil unmanned vehicles gained an enormous momentum driven by technological development pushed by private companies, universities and research entities with heterogeneous goals and aims. The "rise of civil drones" has been mainly triggered by a number of technologies that became widely available (and less expansive) in the civil market during the first years of the last decade (see figure 1). Accurate and fast indoor tracking systems, brushless motors, lightweight transmitters/receivers for wireless communications, LIthium POlymer (LiPo) batteries and solid-state micro circuits for measuring accelerations and angular velocities, became increasingly popular and adopted in important mass markets (such as the then-nascent smartphone industry). The first hobbyists internet communities and open-source/open-hardware projects started to grow and to attract more and more developers (MikroKopter autopilot, Paparazzi project and later Ardupilot and Arducopter projects and Diydrones community, just to name a few) and, during the same years, also the first commercial products and the first companies completely focused on recreational UAVs manufacturing born (e.g. DJI and Parrot with its AR.Drone).

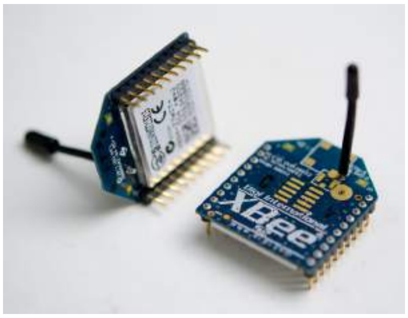
The, sometimes wild, development in the field of UAVs has created a lot of confusion about what actually a *drone* (as these vehicles are commonly called) is. Among the terms more commonly found in literature, UAV is one of the most general ones and it is also the one we will use in this document. It focuses on the absence of a human pilot on-board without further specifying whether the vehicle is fully autonomous or remotely controlled. This second possibility is on the contrary stressed by the acronym Remotely Piloted Aircraft (RPA), a term usually preferred by many national regulative authorities since it underlines the presence of a pilot, albeit not an on-board one. Micro Aerial Vehicle is another term that can be found in scientific literature and it addresses to the world of small-dimension autonomous flying vehicles. The term is particularly common in the robotics-research field in which Micro Aerial Vehicles (MAVs) use has become wide spread because of their agility and high dynamics capabilities. Finally, the term Unmanned Aerial System (UAS) emphasizes the presence of other elements (such as a control link or a Ground Control Station (GCS)) beyond the aircraft itself; Thus focusing and the whole architecture instead then on the flying agent. For the sake of clarity we will try to use only the terms UAV and UAS from here on in this report.

From the regulative point of view, at time of writing (end 2014), national and international regulations concerning the flight of UAVs are still not clear and are slowly evolving. This has somehow prevented the rise of new commercial applications, made possible in theory by the available technology, but limited by the law. While some nations showed an early interest in the adoption of permissive rules for the civil flight of UAVs (Australia, Canada with CASR regulation, 2001 and TC regulation, 2014 respectively), other nations show extreme caution and just extended pre-existing regulations regarding manned aircraft to the case of UAVs. For example, in the United States a Certificate of Authorization (COA) given by the Federal Aviation


(a) *A small IMU on its board.*

(b) *Brushless motor.*

(c) *LiPo battery.*



(d) *Two ZigBee radio-modems.*

Figure 1: The availability of new technologies in the consumer market sustained the exponential growth of inexpensive civil UAVs.

Administration (FAA) is needed for public (i.e. operated by the government of a state) flights of remotely controlled aerial vehicles. Recreational use is permitted, but commercial use is forbidden in any case, since the FAA do not grant COAs for commercial uses of UAVs (FAA regulation, 2012). In Europe, the first country to adopt specific regulations for the flight of UAVs was France, followed by the United Kingdom and Germany. In Italy the regulations became effective starting from April 2014 (ENAC regulation, 2014). The Italian regulations differentiate aerial vehicles according to their use and not to their technical differences, distinguishing between recreational and commercial use and consequently granting different rights to the vehicle owner/operator.

1.1 THE PROJECTS

This document describes the work done during my PhD years and tries to follow all the milestones which I believe have to be marked in the path towards complete autonomy of small UAVs for civil operations, i.e. *control*, *planning* and *perception*. These three major topics are discussed using the results achieved during two different research project - Sistemi e TEcnologie per l'esPlorazione Spaziale (STEPS) and FlyForSmartCity (F4SC) - and during a period as a visiting PhD student at the Universität Zürich (UZH) Robotics and Perception Group (RPG) laboratory.

STEPS – SISTEMI E TECNOLOGIE PER L'ESPLORAZIONE SPAZIALE STEPS is a research project co-financed by Piedmont Region involving universi-

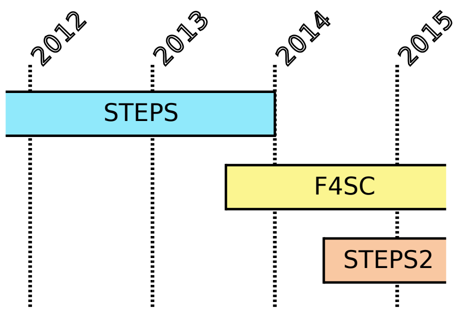


Figure 2: Projects timeline

ties, companies and research institutions of the Piedmont Aerospace District which aims at supporting the research and innovation activities in the Space Exploration domain. One of the assignment carried out by Laboratorio Interdisciplinare di Meccatronica (LIM)[1] as part of this project was to design, implement and build the Vision-based Terrain Navigation Facility (VTNF). The VTNF is a platform meant to be used as testbed for innovative vision-based technologies and solutions, suited to validate them and to perform a deep analysis of their operation in a situation as similar as possible to the operative one the technology has been studied for. The VTNF allows image analysis on pictures taken over a 1:300 scale diorama representing Martian terrain, by an ideal lander in its Entry Descent Landing (EDL) procedures on the planet. A quadrotor with a camera attached on it facing down is used to trigger some shots on the diorama surface. The image processing part is not independent by the motion of the rotorcraft, that will have to be controlled in order to reach defined waypoints in an indoor fixed flight volume. We solved this problem defining a PID (Proportional Integral Derivative) control architecture and the accessory software needed to perform the goal, and integrating all the devices (the tracking system, the quadrotor, the ground station . . .) in a functional system; defining, therefore, a fully operable platform and not only the controller's abstract layer. The work has been carried out in collaboration with Thales Alenia Space-Italy (TAS-I)[2].

FS4C – FLY4SMARTCITY F4SC is a research project started in the late 2013 as a collaboration between Telecom Italia's Joint Open Lab "CRAB"[3], Politecnico di Torino and Centro Nexa[4]. Its goal is to allow a centralized management of a fleet of UAVs distributed on the territory exploiting the advantages offered by the Cloud Robotics approach. The emerging cloud robotics paradigm considers robots as simple agents connected to a remote network infrastructure; this networked approach allows them to benefit from off-board powerful computational and storage resources, to easily access data and to build a common knowledge. FlyForSmartCity aims to demonstrate that a possible implementation of a cloud-robotics service in a smart city scenario is possible by integrating a capable, reliable and stable cloud platform, small unmanned vehicles, high-bandwidth/low-latency mobile networks and open data publicly available on the internet. To achieve the project's goal a quadrotor has been transformed in a network gateway by providing it with high bandwidth 4G/LTE (4th Generation/Long Term Evo-

1 http://www.lim.polito.it/
2 https://www.thalesgroup.com/en/worldwide/space
3 http://jol.telecomitalia.com/jolcrab/
4 nexa.polito.it

(a) *MK Flight Control* (b) *MP 2128g* (c) *PX4FMU*

Figure 3: Autopilots used in the projects

lution) connectivity. A certain number of core functions managed by the auto-pilot (GPS navigation, autonomous take off, autonomous landing . . .) have been exposed by means of specific APIs and have been made accessible from the internet on a safe Virtual Private Network (VPN) link, in order to allow low-latency control and monitoring of the agent from the cloud platform. The first results of the project have been presented in Turin, Italy, during the workshop "Droni: Prospettive di ricerca e scenari applicativi" organized by the Politecnico di Torino and Telecom Italia on June 2014.

1.2 THE VEHICLES

The activities described in this document are based on the use of autonomous *quadrotors* (or *quadcopters*); a quadrotor is a rotorcraft, more specifically a multirotor helicopter, lifted by 4 propellers attached on an equal number of motors. This vehicle architecture offer a series of advantages with respect to other possible choices (e.g. fixed wing autonomous aircrafts etc.). Their Vertical Take Off and Landing (VTOL) capabilities allow for quick take-off and landing operations in busy or space-constrained places. The possibility to hover above the target makes them well-suited for surveillance and monitoring tasks. Moreover they offer a simpler mechanics and require less maintenance with respect to a standard helicopter, while offering similar autonomy and payload. However a quadrotor is an inherently unstable system (G.M. Hoffmann et al., 2007), and for this reason it requires a fast on-board controller to guarantee its stability in standard flights (Bouabdallah and Siegwart, 2005). This controller typically runs on a dedicated electronic board, generically called auto-pilot, equipped with a number of inertial sensors (accelerometers and gyroscopes) to detect the attitude angles of the quadrotor frame and its rotational velocities; barometers and sonar sensors (for altitude measurements), GPS receivers (for autonomous waypoint navigation) and magnetometers (providing an estimate of the absolute heading of the aircraft) are other common devices, often mounted on the same board.

Various quadrotors models and autopilots boards have been used in this project (see fig. 4 on page 6 and fig. 3); Table 1 lists the various combinations tested in the different projects and specifies whether the orientation and position controllers in each project are the auto-pilot standard ones or custom versions.

MICROPILOT'S MP2128g [5] is an auto-pilot board embedding all the peripherals needed for a stable and autonomous quad-rotor flight. This autopilot is specifically addressed to professional use and its software is

5 www.micropilot.com

Table 1: The combinations of the equipment in the projects. See the corresponding chapter for further details (def: default, cus: custom).

UAV	Autopilot	Frame	Controller orientation	Controller position	Project
1	MK FC	ProS3	def	cus	STEPS (ch.2)
2	MP 2128g	ProS3	def	def	F4SC (ch.3)
3	AR.Drone	AR.Drone	def	def	F4SC (ch.3)
4	PX4FMU	AR.Drone	cus	cus	MSF (ch.4)

closed-source. In order to write customized code the user must purchase an add-on set of APIs. These functions constitute a dedicated dynamic linking library that acts as a intermediate layer between the user code and the autopilot software. Using the functions encoded in the library the developer is able to get access to several low-level parameters of the auto-pilot and can modify their values. The board has to be mounted on a mechanical quadrotor frame together with the other peripherals needed for the flight (four motors, the same number of motor controllers and a battery).

AR.DRONE [6] is a commercial ready-to-fly quadrotor solution, controllable via smartphone. It features a front HD camera and the flight stability is ensured by a mother board (running a real-time Linux-based operating system) and a navigation board interfaced with the on-board sensors (two cameras, ultrasonic range finders, gyroscopes and accelerometers). The AR.Drone is mainly conceived for gaming applications, amusement and augmented reality video games, but due to its low-cost and the availability of an official Software Development Kit (SDK), it gained a very good popularity in the academic community.

MIKROKOPTER FLIGHT CONTROL [7] is a commercial auto-pilot board. The board features an Atmel ATMEGA644 microcontroller running at 20 MHz, a 3-axis accelerometer, three gyroscopes and a barometer sensor. Since no GPS is installed, the flight control board alone cannot be used for autonomous navigation, and another board has to be purchased for this purpose. The firmware of the flight control board allows the user to take external control of the UAV (i.e., bypassing the radio controller) by means of a dedicated serial protocol on a Universal Asynchronous Receiver-Transmitter (UART) interface; this link can be used to send the *Roll*, *Pitch* and *Yaw* commands to the UAV.

PX4FMU [8] is a complete auto-pilot solution designed as part of the PX4 open-source/open-hardware project. It features a 32 bit ARM Cortex M4 Processor running the NuttX Real Time Operating System (RTOS) offering Unix/Linux-like programming environment. The presence of an on-board operating system simplifies a lot the development of custom applications on the platform for the final user when compared to firmwares written with a monolithic approach. On the communication

6 ardrone2.parrot.com
7 wiki.mikrokopter.de/en/FlightCtrl_ME_2_1
8 pixhawk.org

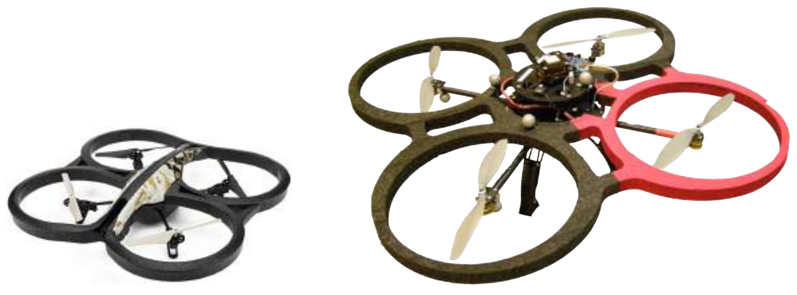

(a) *Parrot AR.Drone* **(b)** *ProS3 Octane*

Figure 4: Quadrotors used in the projects

side the PX4FMU fully support *MAVlink*[9], an header only messaging library, specifically designed for small and micro UAVs.

Two different mechanical frames have been used in the project. The *ProS3 Octane*'s frame is composed by 4 multi-directional carbon fiber beams. On the external side of each beam a brushless motor is fastened on a carbon fiber plate, linked to the beam through an aluminium support. On the other end, arms are integrated between two carbon fiber disks, upon which the electronic boards are assembled. The overall dimensions are $900 \times 900 \times 290$ mm. The *AR.Drone* mechanics, on the other side, is simpler and composed by a central plastic cross with integrated motors and motor controllers on the 4 ends. The frame is inexpensive and easily replaceable; its dimensions are $450 \times 450 \times 15$ mm.

9 http://qgroundcontrol.org/mavlink/start

2 STEPS

This chapter reports the design and the development of the Vision-based Terrain Navigation Facility (VTNF): a facility for the benchmarking of software modules and sensors to be used during planetary Entry Descent Landing (EDL). The motivation for this work lies in the growing need for autonomy in space research. On-board autonomy is in fact the key factor for the future space robotic missions, and it is particularly relevant for missions towards planets at high distances from Earth, where a robotic closed loop control in real time is not possible (e.g. the travel time of a radio-frequency signal requires from 4 to 22 minutes covering the distance between the Earth and Mars). Over the last decades, within the Viking1 and Viking2 lander missions (1976), Mars Path Finder (1997), Mars Exploration Rover (Spirit and Opportunity) (2004), Phoenix (2008), and Mars Science Laboratory (2009), the landing major axis ellipse has been reduced from 300 km to 20 km (Braun and Manning, 2006). The current aim is to further reduce this dispersion ellipse in order to achieve, within few years, ellipse landing of hundreds of meters. In this general and ambitious goal, computer vision algorithms may help obtaining information about the 3D environment of space missions to verify the correct execution of robotic commands. Unfortunately, the difficulty of gathering real data and experimenting on the field is a real limitation for these approaches, and indeed most of the computer vision research for space applications is carried out by very few well known premises - such as the NASA Jet Propulsion Laboratory (JPL). With the exception of some early works (see e.g. Cheng, Johnson, Matthies, and Wolf, 2001, Cheng, Goguen, et al., 2004), the Descent Image Motion Estimation System (DIMES) (Cheng, Johnson, and Matthies, 2005), developed by NASA, can be considered as the first historical attempt to exploit computer vision to control a spacecraft during the EDL phase. More recently, thanks to the successful application to the Mars Exploration Rovers (MER) landings, NASA invested in the last years in designing and developing vision-based solutions for EDL. The works in Huertas et al., 2006 and Cheng, 2010, describe improvements of DIMES technology for what concerns the problems of slope estimation and hazard detection on the planet surface.

The VTNF is explicitly designed in order to provide a simulation and validation environment in the study of vision-based routines and algorithms for space applications. By providing a scenario as similar as possible to the actual operative one, the VTNF allows a deep analysis of the EDL phase of a virtual Martian lander approaching Mars. In the experiments, a quadrotor with a camera attached on it facing down is used to trigger some shots on the surface of a 1 : 300 scale Mars diorama. The motion of the rotorcraft is dependent by the analysis of the pictures, since the quadrotor is controlled in order to reach defined waypoints in the indoor fixed flight volume, computed as a result from the image processing part. The choice of using an UAV as a *flying camera* follows the remarkable results reached by a number of research institutions and laboratories that have focused their work on the study of the dynamics of small aerial vehicles (e.g. multirotors and coaxial small helicopters) and their control. In particular, many of these results have been made possible by the development of special testbeds that typically provide a large flight volume in which the exact position and attitude of one or many agents is accurately measured by an optical tracking system. The *MIT Raven* (How et al., 2008) testbed is commonly considered the first described in literature, with *STARMAC* by Stanford University (G. Hoffmann et al., 2004), the *Flying Machine Arena* at Eidgenössische Technische Hochschule Zürich (ETHZ) (Lupashin et al., 2014) and UPenn's *GRASP Laboratory Test Bed* (Michael et al., 2010) being other noticeable examples of this technology. The uses of these testbeds range from the study of aggressive or coordinated multi-vehicles manoeuvres (Mellinger et al., 2012,Ritz et al., 2012), to architecture (Willmann et al., 2012), construction (Lindsey et al., 2012) and entertainment (Augugliaro et al., 2013). The VTNF has been inspired by the cited works and developed by TAS-I (project leading and computer vision algorithm) and the LIM (hardware/software integration, UAV control).

2.1 FUNCTIONAL LAYOUT

From the functional point of view, the facility is composed by three main components:

- the Tracking System;
- the Quadrotor;
- the Ground Segment.

Figure 5 depicts the main data flows exchanged between every single module.

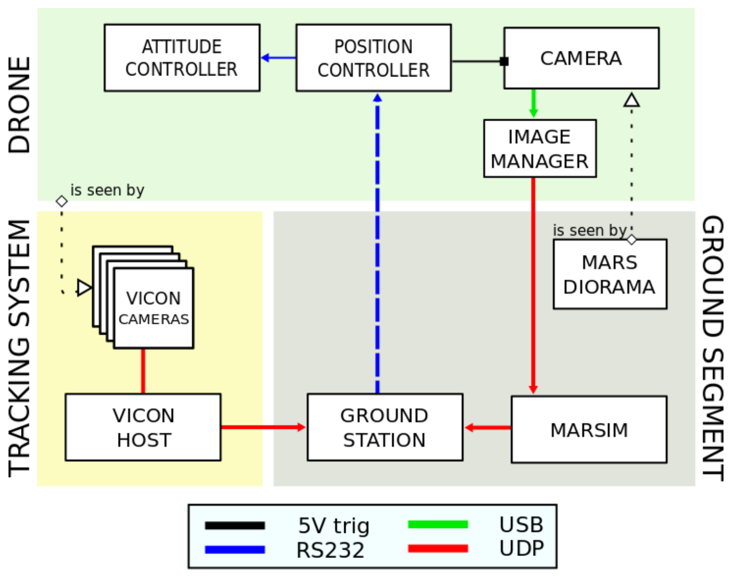


Figure 5: Functional architecture of the VTNF.

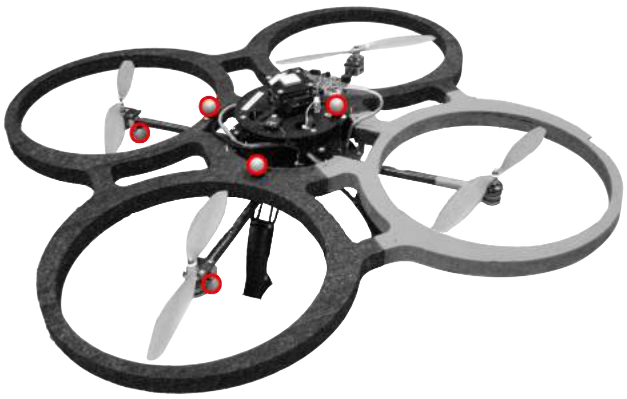


Figure 6: The IR markers attached on the vehicle.

2.1.1 The Tracking System

The purpose of the tracking system is to measure the position and the attitude angles of the quadrotor while it flies inside the tracked volume. The tracking system used in the VTNF is based on the *Vicon Motion Capture System*[1] and is composed by 13 Vicon Bonita infrared cameras, connected together in a Power over Ethernet (PoE) network hosted by a dedicated workstation (*Vicon Host* in figure 5) on which the proprietary Vicon Tracker software runs. The infrared cameras are attached on a cube-shaped aluminum structure with 9 m long sides, thus providing an available fully tracked flight volume of approximately 650 m^3, not considering the volume occupied by the diorama. They acquire 0.3 Megapixel images up to 240 Hz and can track a single marker with an accuracy of 1 mm and sub-mm precision. The markers are little spheres covered with an infrared-reflective coating, attached on the body to track (see figure 6). In order to reconstruct without ambiguity the full pose of a rigid body, at least three markers are needed.

2.1.2 The Quadrotor

Among the various solutions considered for handling the camera in the flight volume above the diorama (e.g. robotic arms, blimps, cable cameras) a quadrotor offers high maneuverability, good payload capabilities, simple mechanics and low maintenance. The model used in the VTNF is shown in Fig. 8; as it has already been anticipated in section 1.2, the UAV's frame is manufactured by ProS3, while the attitude controller, in charge of the vehicle stability, is a MK Flight Control board. The quadrotor has a custom lightweight carbon fiber frame and can bring a payload up to 1 kg heavy with a battery autonomy of about 15 minutes. In our set-up the camera (in its gimbaled or fixed version), the position control board, the radio modem and a PC104+ single board computer constitute the payload of the vehicle; its total weight amounts to 2.5 kg.

The position controller runs on an additional Arduino Mega 1280. The board runs four parallel Proportional Integral Derivative (PID) controllers, one for each of the remaining degrees of freedom (x, y, z and *Yaw* angle). This board receives on a dedicated wireless-serial link the feedback obtained

1 http://www.vicon.com/

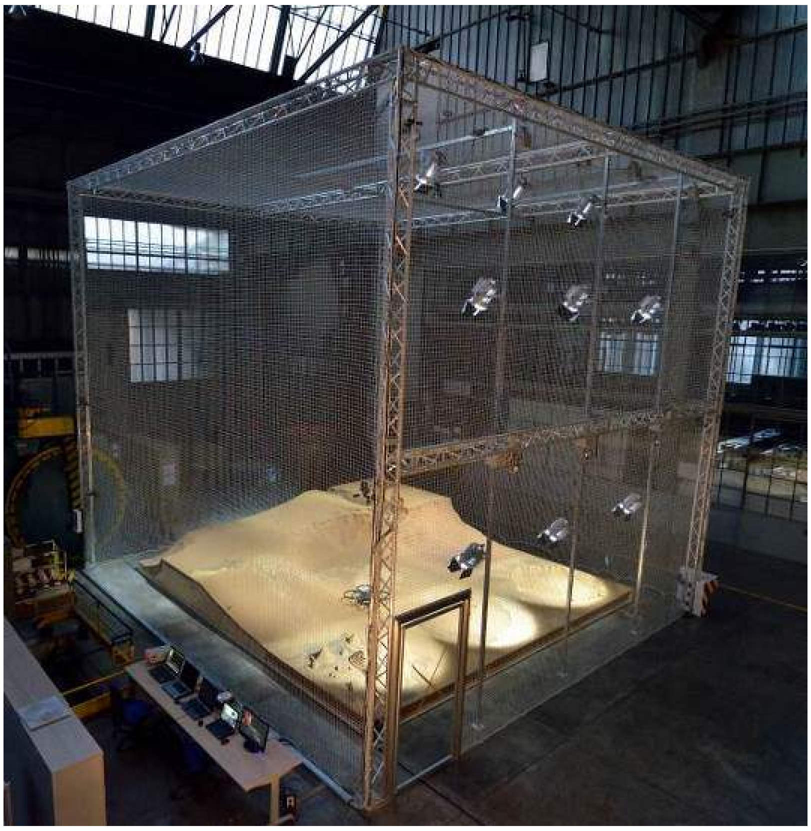


Figure 7: The Vision-based Terrain Navigation Facility.

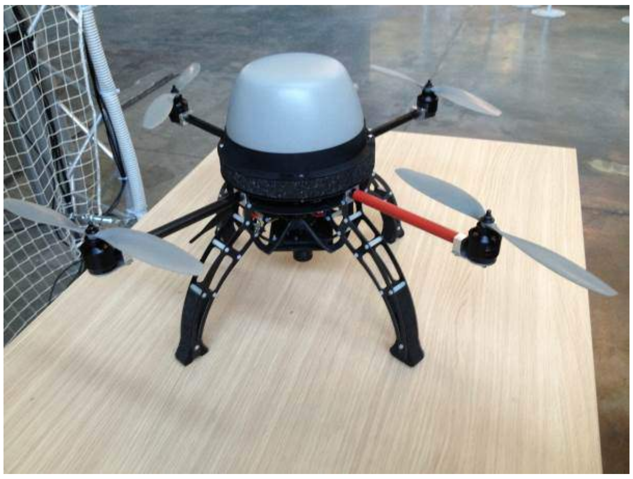


Figure 8: The quadrotor used in the VTNF.

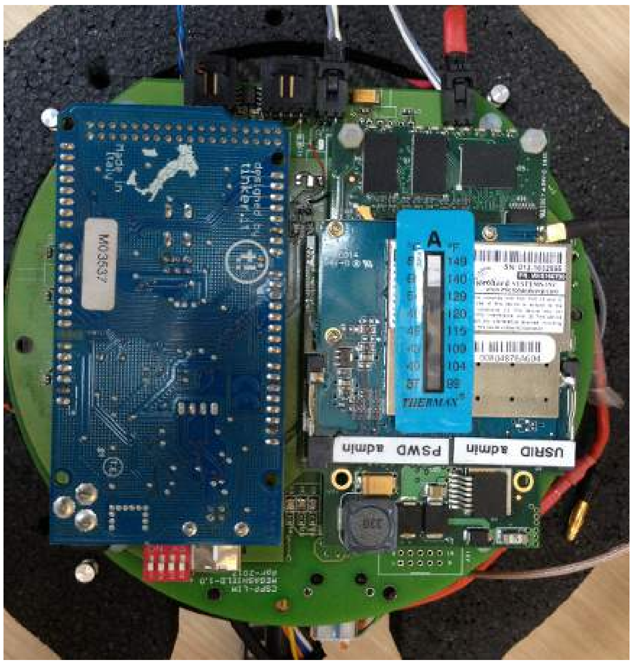


Figure 9: The Arduino board running the position controller and the on-board radiomodem attached on their carrier board.

from the User Datagram Protocol (UDP) Vicon data stream and sends the commands to the auto-pilot board. The board takes also care of triggering the camera when the MarSim requests a new picture.

A camera and a PC104+ single board computer are used to take pictures of the diorama and to deliver them to the ground segment. The computer features an Intel Atom D510 dual core 1.62 GHz CPU with 2GB RAM. It retrieves the images sent by the camera on the USB link and forwards them to the MarSim. The camera used in the project is the 1312M model[2] by Edmund Optics. It is a Complementary Metal-Oxide Semiconductor (CMOS) gray level camera (1280 $\times$ 1024 resolution, 8 bits pixel depth) with a rolling shutter. This particular model has been chosen since its characteristics are comparable to the already space qualified camera based on STAR1000 sensor. It can be attached to the UAV both in its fixed mount version and in a fully integrated gimbal solution.

A wireless link connects the UAV to the ground segment using a couple of identical radiomodems (see Fig. 9) featuring either a standard RS232 serial interface or an IEEE802.11a LAN wireless link. The first link is used to send all the time-critical data to the position controller, i.e. the current attitude of the quadrotor, the target position and the camera-trigger signal; the latter conveys via UDP to the ground station the pictures coming from the camera.

2.1.3 The Ground Segment

The Ground Segment in the VTNF is composed by two software modules (the *Ground Station* and the *MarSim* in fig. 5) and by a diorama.

2 http://tinyurl.com/naog3ll

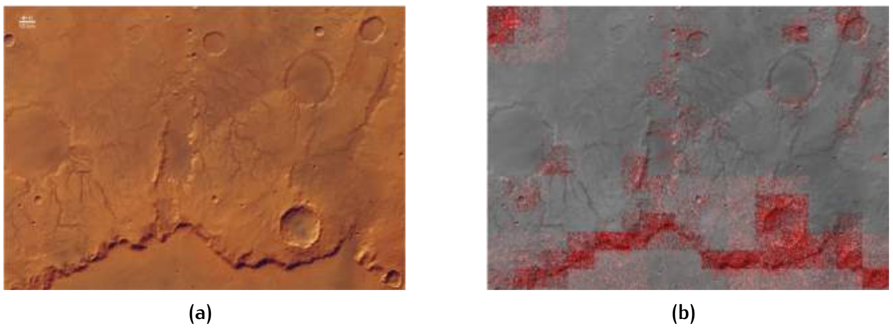


Figure 10: A real image of Martian terrain (10a) and the output of the hazard map algorithm (10b). The dangerous areas for landing have been highlighted.

The Ground Station

The Ground Station module is a simple module that accesses in real-time positions and angles of the UAV, as provided by the Vicon Host on the UDP link, and adapts them to provide a continue pose feedback to the quadrotor position controller. Moreover, this module receives the target position by the MarSim and triggers a new picture when the error is less than a tunable threshold; then it acknowledges the MarSim that it is ready to compute a new target point.

The MarSim

The MarSim is a TAS-I proprietary module. It embeds both a complete functional and dynamical model of the extra-planetary lander and the computer vision routines that are the object of the testing activities. The MarSim receives as input the pictures triggered by the UAV and outputs a succession of Martian geographical coordinates, computed and updated at runtime during the experiment. These are the result of a complex chain of operations performed to simulate the behaviour of a real lander in its EDL phase (image processing, data fusing with on-board sensor, computation of the new nominal trajectory, actuation of thrusters, integration of the dynamic until the new point of interest). The coordinates are then scaled and re-projected in the diorama reference frame and passed to the quadrotor as target waypoints. Figure 10 offers an example of a hazard map generated by the simulator in this stage, starting from a picture of a portion of Martian terrain. Further details about the MarSim can be found in Lanza et al., 2012.

The diorama (8 m $\times$ 8 m, 1.5 m of maximum relief height) accurately reproduces four peculiar geographic details of the Mars surface in $1:300$ scale:

- Nili Fossae
- Victoria Crater
- Xanthe Terra
- Dilly Crater

Nine mercury-vapor stage lamps provide accurate and homogeneous lighting. Figure 11 shows the good level of resemblance between two actual Martian surfaces, and their equivalent reproduction on the diorama.

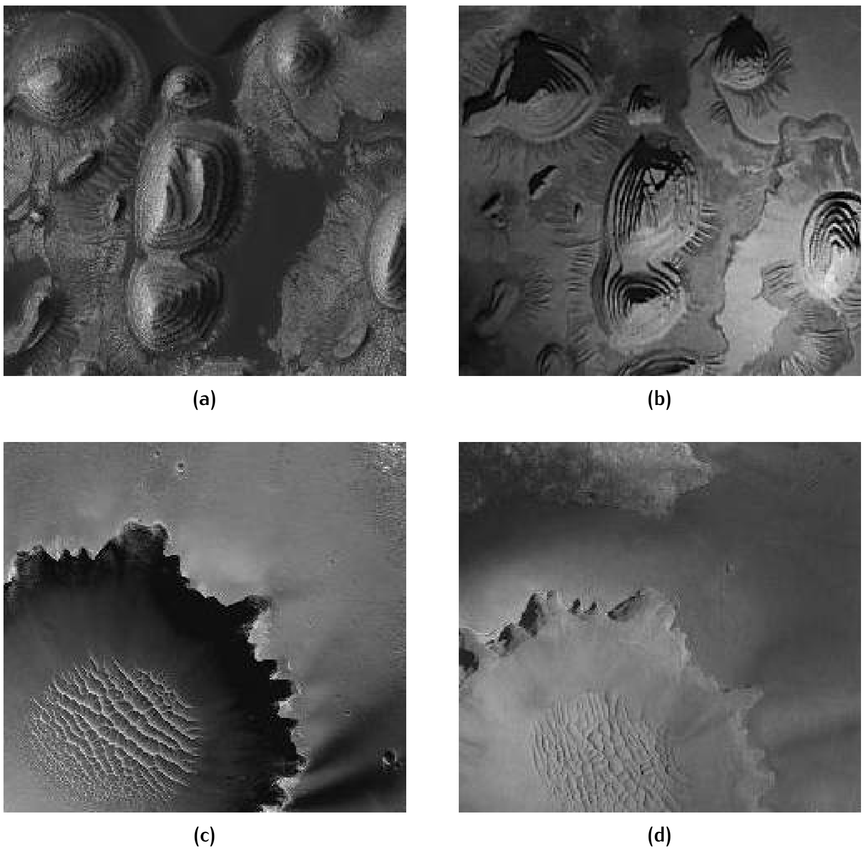


Figure 11: A portion of Western Arabia Terra (a) and Victoria Crater (c) and the corresponding representations on the diorama (b,d).

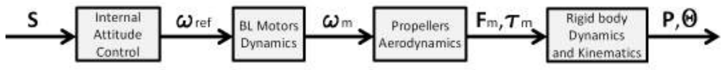


Figure 12: Quadrotor model structure.

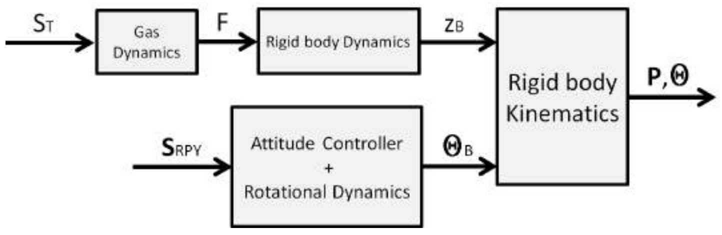


Figure 13: Revised quadrotor model structure.

2.2 SYSTEM MODELING AND CONTROL

A simple representation of the UAV model is shown in figure 12, where $\mathbf{S}$ represents the sticks references vector, ω_{ref} the angular velocity references in input to the brushless controller boards, ω_m is the vector of angular velocity of the motors, $\mathbf{F}_m$ e $\boldsymbol{\tau}_m$ are the generalized force acting on the end of each arm.

However in the current application, a high dynamic response of the quadrotor does not represent a critical requirement; in fact the UAV does not have to perform aggressive manoeuvres, nor has it to respect strict dynamical or trajectory constraints. Given the following conditions in the flight of the quadrotor:

- Small *Roll* and *Pitch* angles (± 6 °),
- *Yaw* angle fixed at $\psi = 0$ °,
- Low translational velocities,
- Hovering flight (i.e. constant thrust value),

the dynamical model of the vehicle can be completely linearized and the x, y, z and *Yaw* dynamics can be decoupled making it possible to model the vehicle as shown in picture 13.

In the next sections we will show how a very simple control architecture, featuring four parallel PID controllers (acting on the x, y, z and *Yaw* degrees of freedom of the quadrotor), is sufficient to move and stabilize the quadrotor on its target position.

2.2.1 Translational (x,y) dynamics model

The coordinate systems for the quadrotor is shown in 14 on the facing page. The earth frame is defined by the $\{x^E, y^E, z^E\}$ axis. The body reference frame is defined by a right-handed reference frame, attached to the center of mass of the quadrotor, with Z axis oriented downward and X axis opposite to the preferred forward direction, body frame axes are $\{x^B, y^B, z^B\}$. Euler ZYX angles are used to model the orientation of the quadrotor in the Earth frame. The final direction cosine matrix representing body frame attitude in the earth frame is given by 4 on the next page and it is computed as the product of a first rotation about z^E axis by the *Yaw* angle ψ, a second

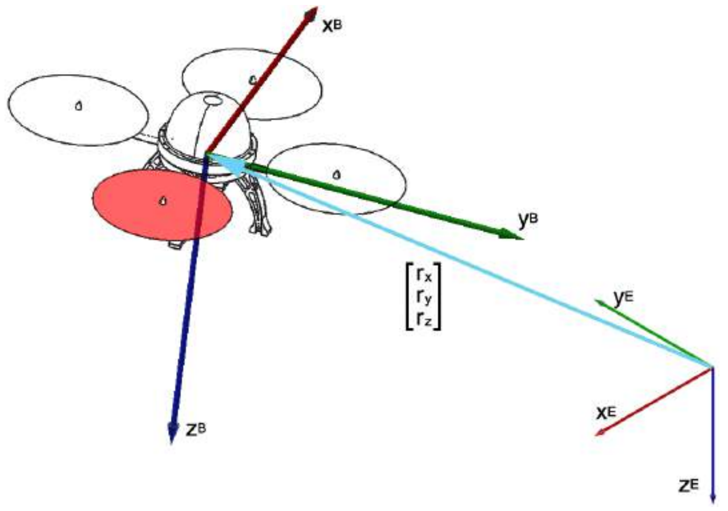


Figure 14: Body (B) and Earth (E) reference frames.

rotation about y^E axis by the *Pitch* angle ϑ and a final rotation about x^E axis by the *Roll* angle φ. The three rotations are referred to the earth frame.

$$\mathbf{R}_z(\psi) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$\mathbf{R}_y(\vartheta) = \begin{bmatrix} c_\vartheta & 0 & s_\vartheta \\ 0 & 1 & 0 \\ -s_\vartheta & 0 & c_\vartheta \end{bmatrix} \tag{2}$$

$$\mathbf{R}_x(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi & c_\varphi \end{bmatrix} \tag{3}$$

$$\mathbf{R}_{RPY} = R_z(\psi)R_y(\vartheta)R_x(\varphi) = \begin{bmatrix} c_\psi c_\vartheta & -s_\psi c_\varphi + c_\psi s_\vartheta s_\varphi & s_\psi s_\varphi + c_\psi s_\vartheta c_\varphi \\ s_\psi c_\vartheta & c_\psi c_\varphi + s_\psi s_\vartheta s_\varphi & -c_\psi s_\varphi + s_\psi s_\vartheta c_\varphi \\ -s_\vartheta & c_\vartheta s_\varphi & c_\vartheta c_\varphi \end{bmatrix} \tag{4}$$

The full non-linear dynamics of the system can be described using Euler equations:

$$m\ddot{\mathbf{r}} = \mathbf{F}^E \tag{5}$$

$$\mathbf{J}\dot{\boldsymbol{\omega}}^B + \boldsymbol{\omega}^B \times \mathbf{J}\boldsymbol{\omega}^B = \mathbf{M}^B \tag{6}$$

where $\mathbf{r}$ describes the position of the center of mass of the body, $\mathbf{F}^E$ is the resulting force applied to the body in the Earth frame, $\mathbf{J}$ represents inertia matrix of the quadrotor, $\boldsymbol{\omega}^B$ it the angular velocity seen in the body frame and $\mathbf{M}^B$ is the torque resultant acting on the body.

Let $\mathbf{g}$ denote the gravity acceleration vector $\mathbf{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ and let $\mathbf{T}^E$ be defined as the total rotors thrust as seen in the earth frame. Neglecting disturbance due to ground effect and turbulences, it is introduced a D scalar coefficient modeling the losses of the UAV as a term proportional to its center of mass velocity and acting in opposite direction. The total force $\mathbf{F}^E$ acting on the center of mass of the quadrotor can be expressed as

$$\mathbf{F}^E = -D\dot{\mathbf{r}} + m\mathbf{g} + \mathbf{T}^E \tag{7}$$

The not too strong dynamics requirmentes of the desired system allows to work with:

- small roll and pitch angles;
- near-hovering flight condition;
- limited translational velocities.

The first point allows to linearize some terms in the direction cosine matrix, the second allows to uncouple translational dynamics and rotational one, the third permits to neglect effects of drag aerodynamics force. The losses in battery power over time are also neglected so that the system model can be considered time-invariant. The linearization of the cosine matrix ($\sin\alpha \approx \alpha$ and $\cos\alpha \approx 1$) for small *roll* and *pitch* angles returns:

$$\mathbf{R}^l{}_{RPY} = \begin{bmatrix} c_\psi & -s_\psi & s_\psi\varphi + c_\psi\vartheta \\ s_\psi & c_\psi & -c_\psi\varphi + s_\psi\vartheta \\ -\vartheta & \varphi & 1 \end{bmatrix} \tag{8}$$

Equation 5 on the preceding page can thus be expressed as:

$$m\ddot{\mathbf{r}} = m\mathbf{g} + \mathbf{R}^l{}_{RPY}\mathbf{T}^B \tag{9}$$

here $\mathbf{T} = [0 \quad 0 \quad T]^T$ is represented in the body frame: The single acceleration components on the three axis are thus given by:

$$\begin{aligned} \begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix} &= \frac{1}{m} \begin{bmatrix} c_\psi & -s_\psi & s_\psi\varphi + c_\psi\vartheta \\ s_\psi & c_\psi & -c_\psi\varphi + s_\psi\vartheta \\ -\vartheta & \varphi & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \\ &= \frac{1}{m} \begin{bmatrix} Ts_\psi\varphi + Tc_\psi\vartheta \\ -Tc_\psi\varphi + Ts_\psi\vartheta \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \end{aligned} \tag{10}$$

This equation still couples roll and pitch attitude angles with the thrust force and presents some trigonometric terms related to the rotation of the local frame around Z^E axis. However some simplification can still be done. The terms that couple angles and throttle can be neglected linearizing them near the hovering condition. In this condition the total thrust approximately counteracts gravity: $T = \overline{T} \approx -mg$ and 10 can be written as follows:

$$\begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix} = \frac{\overline{T}}{m} \begin{bmatrix} s_\psi & c_\psi & 0 \\ -c_\psi & s_\psi & 0 \\ 0 & 0 & \frac{1}{\overline{T}} \end{bmatrix} \begin{bmatrix} \varphi \\ \vartheta \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{11}$$

In addition, if ψ angle is zero (i.e. if the vehicle keeps its heading constant) the model can be totally linearized and 10 becomes:

$$\begin{bmatrix} \ddot{r}_x \\ \ddot{r}_y \\ \ddot{r}_z \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 0 & \overline{T} & 0 \\ -\overline{T} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \varphi \\ \vartheta \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \overline{T}\vartheta \\ -\overline{T}\varphi \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{12}$$

The following simple representation of its translational dynamics can be derived (Castillo et al., 2004; Waslander et al., 2005):

$$\begin{bmatrix} \ddot{r_x} \\ \ddot{r_y} \\ \ddot{r_z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} 0 & \overline{T} & 0 \\ -\overline{T} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \varphi \\ \vartheta \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{13}$$

With r_x, r_y, r_z being the coordinates of the center of mass of the UAV in the earth-fixed frame, m the total mass of the quadrotor, $\overline{T}$ the hovering thrust value; φ and ϑ respectively are the *Roll* and *Pitch* rotation around x_B and y_B axes.

However, (13) does not completely characterize the description of the dynamics since its angular dynamics is still not known. Unfortunately, Mikrokopter offers very poor documentation about the design of the attitude control loops design, as noted also in Sa and Corke, 2011. For this reason their dynamics have been experimentally identified and validated by means of dedicated experiments. A similar procedure has been performed for the thrust command, however its dynamic response has been approximated with a simple static gain plus saturation.

2.2.2 Gas Dynamics

In order to identify the *Gas Stick Command* vs. *Thrust* the value of the lift force expressed in Newtons has been measured for increasing gas stick values using an electronic balance. Also the angular velocity of the propellers has been measured by using a laser sensor. The results of the experiments are shown in figure 15 and highlight a linear dependence between gas stick command and lift force with a saturation at 33 N (equation 14) and a quadratic dependence between the command and the rotation of the propellers expressed in [rpm].

$$T = \begin{cases} 0.1489 \cdot GAS_{cmd} \quad [N], & \text{if } GAS_{cmd} = [1..222] \\ 33 \quad [N], & \text{if } GAS_{cmd} =]222..256] \end{cases} \tag{14}$$

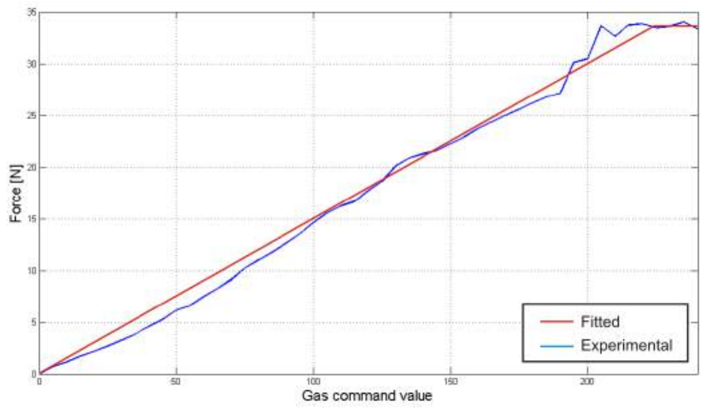
$$\omega_{rpm} = -0.06589 \cdot {GAS_{cmd}}^2 + 36.67 \cdot GAS_{cmd} + 751.2 \tag{15}$$

2.2.3 Roll and Pitch dynamics

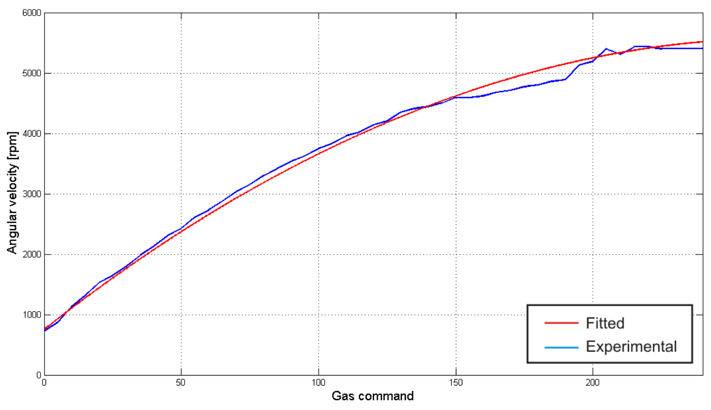
A possible way to perform the identification tests for roll and pitch dynamics is by sampling the input and output signal in the frequency domain. A swept frequency (ranging from 0.1 to 2.0 Hz) sine command waveform is used as the input signal and the gain and phase displacement of the output waveform for different frequency are measured. This allows to trace some points of the bode plot of the dynamic system that can be fitted in between to obtain an approximation of the model characteristic response. An alternative approach to model identification of MK dynamical model in time-domain can be found in Sa and Corke, 2011, where the attitude model of the vehicle is fitted using recursive least squares.

The identification process gives the experimental responses shown in figure 16 and 17; responses on the two angles are quite similar and then both have been fitted with a third order transfer function, presenting three poles in $f = 2.39$ Hz (equation 16), whose bode plot is shown in the same figures.

$$Fit_{RP}(s) = \frac{0.06299}{0.00029s^3 + 0.01333s^2 + 0.2s + 1} \tag{16}$$



(a) *Command vs. Force.*



(b) *Command vs. Rpm.*

Figure 15: Thrust characterization.

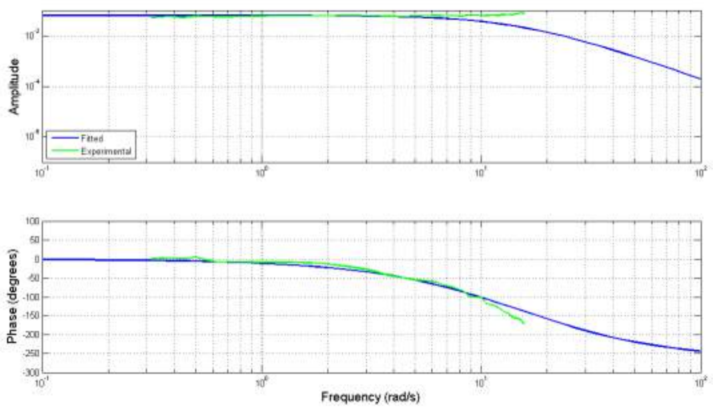


Figure 16: Experimental roll response vs. fitted one. Blue lines represent the angular frequency response of the roll controller, identified by fitting the experimental results (green lines).

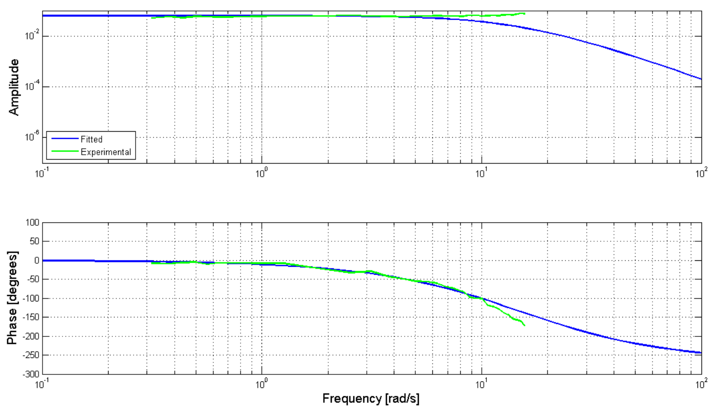


Figure 17: Experimental pitch response vs. fitted one. Blue lines represent the angular frequency response of the pitch controller, identified by fitting the experimental results (green lines).

Roll and Pitch dynamics validation

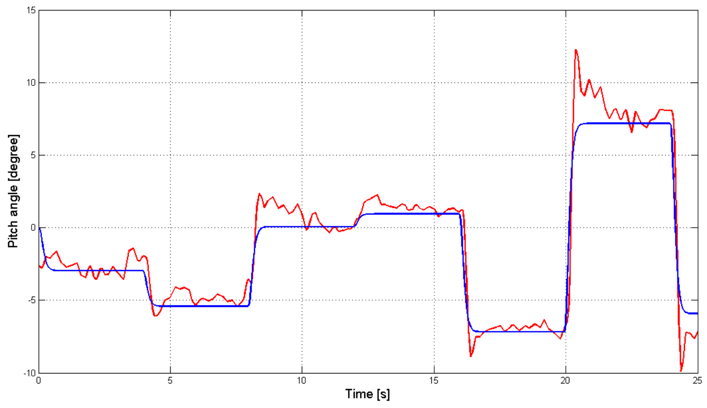
The model validation has been performed by submitting a known waveform and comparing the simulated response of the system to the real one. In our case two peculiar command waveforms have been submitted to the identified model, the simulated system response versus the real one is reported both for pitch and roll dynamics in figure 18 on the following page. In both cases the response of the fitting $\mathrm{Fit}_{RP}(s)$ function is similar to the experimental one. With the random amplitude square wave input the simulated response follows quite good the trend of the experimental one; excluding the oscillation due to disturbances, the most important difference stays in the overshoots of the experimental response that cannot be modelled by the low pass function used for the fitting. The frequency sweeping sine reference instead, gives important information about the phase fitting of the model that results very similar to the experimental one. The difference between the two models is represented by the gain, in the modelled case it is about 25% lower than the experimental curve. In conclusion the modelling function, despite its simplicity and some response imperfections, matches in a sufficiently accurate way the experimental results, and gives a good approximation of the system behaviour.

2.2.4 PID Controllers

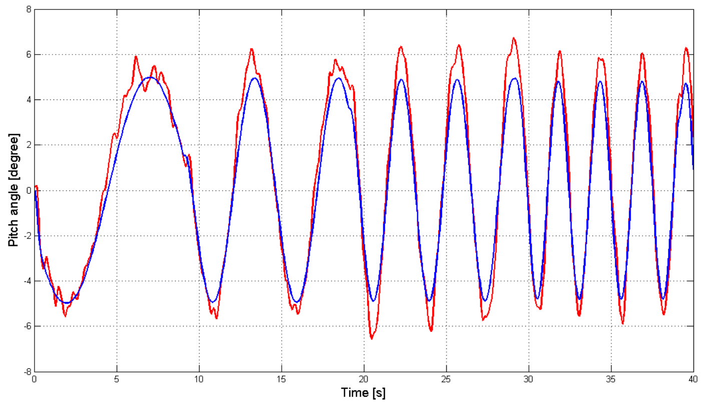
The PID controllers have been designed in order to keep the rising time in the order of 2.5 s for a 1 m step reference on x, y, z. The controller algorithm runs in one of the Arduino Mega's timer interrupt routines at 30 Hz. The digital form of the controller (18) is obtained by discretizing the standard *s-domain* PID expression (17) using *Backward Euler* method.

$$R(s) = K_P + \frac{K_P}{T_I s} + \frac{K_P T_D s}{1 + s\frac{T_D}{N}} \tag{17}$$

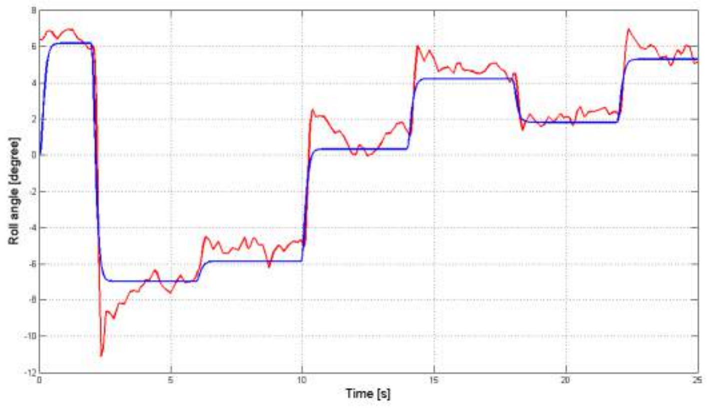
$$u(k) = p(k) + i(k) + d(k) \tag{18}$$



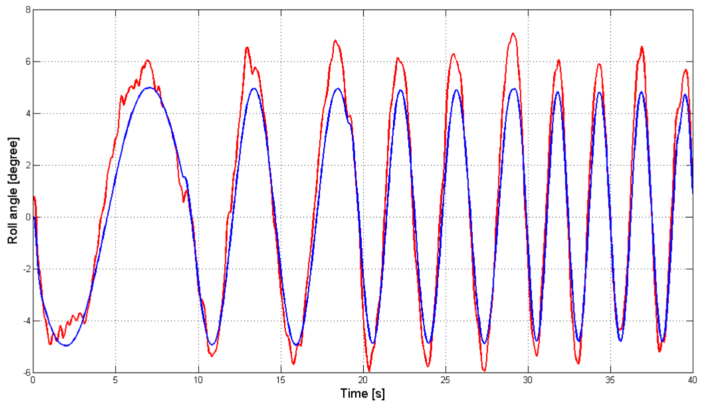
(a) *Pitch random square wave command.*



(b) *Pitch swept sine wave command.*



(c) *Roll random square wave command.*



(d) *Roll swept sine wave command.*

Figure 18: Measured and simulated response.

Table 2: PID parameters.

	K_P	K_D	K_I	N	T_S [s]
PID XY	0.091	0.012	0.12	20	0.0328
PID Z	0.055	0.01	0.04	20	0.0328

where:

$$\begin{cases} p(k) = K_p e(k) \\ i(k) = i(K-1) + \frac{K_P T_s}{T_I} \\ d(k) = \frac{T_D}{N T_s + T_D} d(k-1) + \frac{K_P T_D N}{N T_s + T_D}(e(k) - e(k-1)) \end{cases} \tag{19}$$

In equation 18 on page 19 $T_D = \frac{K_D}{K_P}$, $T_I = \frac{K_P}{K_I}$, K_P, T_s, are respectively the *Derivative Time*, *Integrative Time*, *Proportional Gain* and *Sampling Time*. N sets the location of the pole in the derivative filter. Table 2 summarizes PID parameters values.

The final digital transfer functions of the XY and Z controllers are reported in 20 and 21; please note that similarly to the dynamics response, also the controller transfer function is assumed the same both for the X and Y controller.

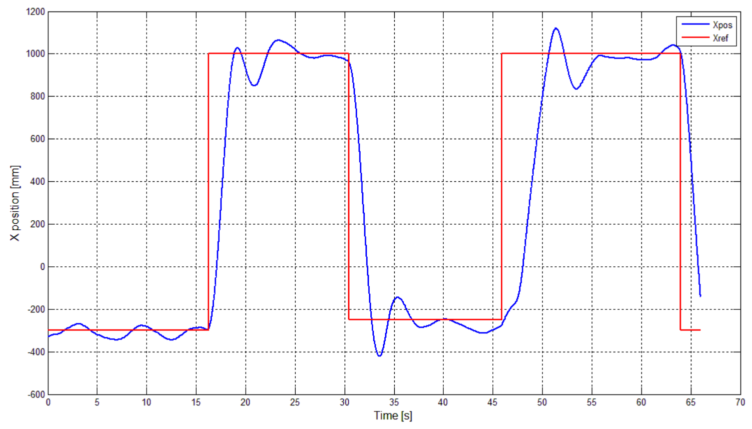
$$R_{XY}(z) = \frac{1.19z^2 - 2.352z + 1.162}{z^2 - 1.7z + 0.699} \tag{20}$$

$$R_Z(z) = \frac{0.4212z^2 - 0.8259z + 0.4046}{z^2 - 1.7z + 0.699} \tag{21}$$

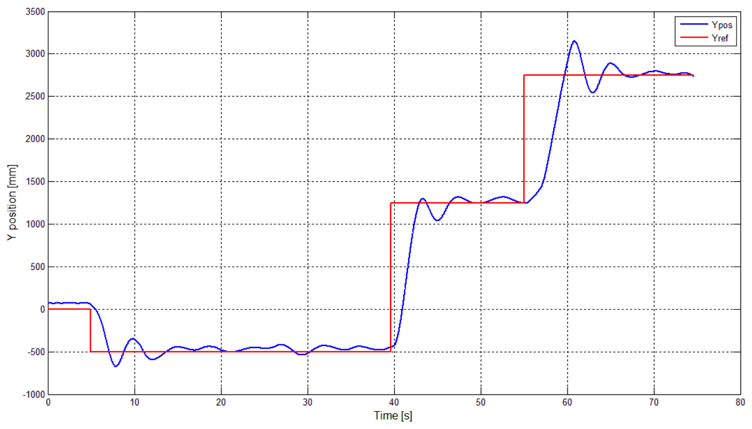
Experimental tuning of the PID parameters have been conducted directly on the plant to further adjust the flight performance of the vehicle. Figures 19, 20 and 21 show the experimental results after some test flights. The positioning error causes the UAV to hover in a circle of maximum radius 6 cm around the target point while the Z error varies within a maximum range of ± 8 cm (Figure 19c). The Yaw angle is kept within a precision of ± 3 °. Dedicated tests have shown that Vicon Tracking System provides, after a good gaber calibration and with optimal camera coverage, 1 mm accurate and 0.1 mm precise position data. The maximum latency in the camera trigger signal has been measured to be 40 ms.

2.3 CONCLUSION

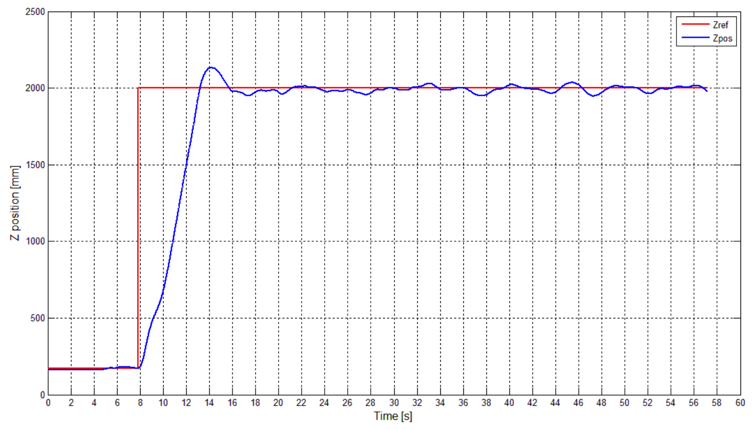
In this chapter we have presented the Visual Terrain Navigation Facility, a testbed for studying and validating vision-based routines and algorithms to be used during planetary Entry Descent and Landing (EDL) by a lander. We have described the major technical details of the architecture and the design choices in relation with its three main functional components: the Tracking System, the Ground Segment and the Quadrotor. We have finally shown how, in our operation hypothesis, a very simple PID-based control architecture successfully stabilizes the quadrotor and allows basic autonomous navigation functionalities with good performances. The VTNF is currently used in *Thales Alenia Space*'s headquarters in Turin both for the already described purposes and in technological demonstrations. Although the VTNF has been created keeping in mind the validation of specific aerospace-derived



(a) *X response.*

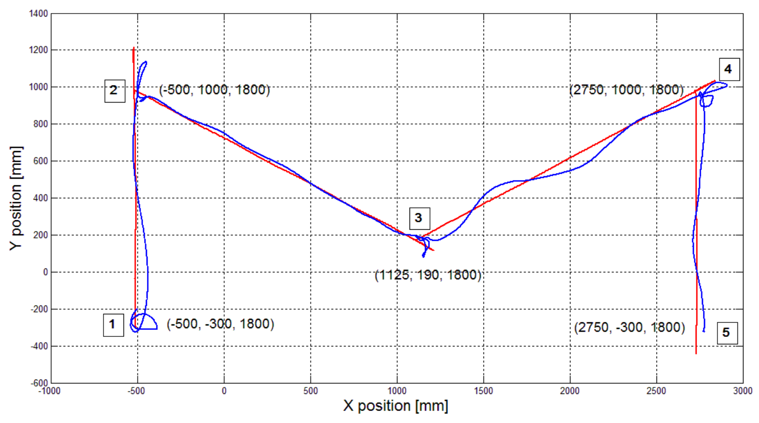


(b) *Y response.*

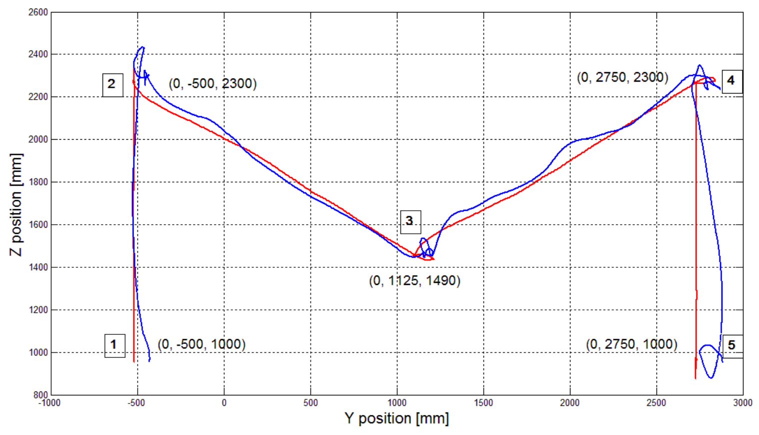


(c) *Z response.*

Figure 19: The dynamic behaviour of the vehicle on the three axes.



(a) *Flight on XY plane.*



(b) *Flight on YZ plane.*

Figure 20: The dynamic behaviour of the vehicle in two 2D flights.

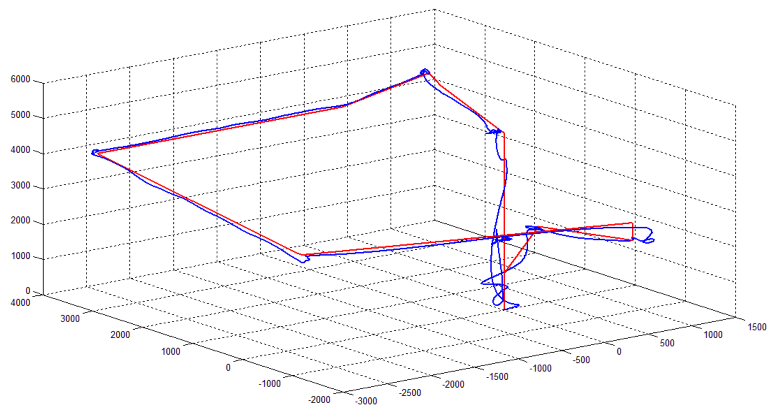


Figure 21: A complete flight in the tree dimensional volume.

image algorithms, it offers noticeable potentialities for a number of applications not strictly related with the original purposes of the project. Moreover the research activities in the facility are not over; experimentation with visual-aided auto-takeoff and auto-landing routines, visual odometry and new control strategies are currently carried out in the VTNF. An outdoor GPS-based version of the facility is also currently under development and will permit image analysis and EDL simulation on larger scale.

3 FLY4SMARTCITY

Cloud robotics is an emerging field in robotics. It embraces typical internet-based technologies (e.g. cloud computing, cloud storage, web-services, etc.) in order to allow an artificial agent to take advantage of the network's resources to off-load compute-intensive tasks. This naturally leads to a paradigm shift in which robots become *simple* agents that belong to a common cloud computing platform (Waibel et al., 2011), and represents a further step in the direction of the *Internet of Things* (Atzori et al., 2010) from the definition of a generic Network Robot Systems (NRSs) as presented in Sanfeliu et al., 2008. The cloud platform manages the robots and provides them with the possibility of sharing knowledge and performing demanding real-time data processing (such as localization, mapping, grasping, multi-input mission planning, etc.) (Chen et al., 2010, Waibel et al., 2011). Among the most noticeable works which exploit the cloud robotics approach, Quintas et al., 2011 presents a service-oriented architecture allowing distant groups of robots to share and exchange learned skills and improve cooperation with human agents. The work relies on cloud computing to provide an increased degree of scalability to the system. In Kamei et al., 2012 the requirements in typical daily supporting services have been examined through example scenarios that target senior citizens and the disabled, together with a discussion about the key research challenges offered by the cloud network-robotics approach. Similarly, in Chibani et al., 2013 the use of cloud robotics has been investigated for physical and virtual companions assisting people in their daily living (e.g., ubiquitous robots that are able to co-work alongside people). In Hunziker et al., 2013 the authors present *Rapyuta*, an open source Platform-as-a-Service (PaaS) framework for robotics applications. Rapyuta is the engine underlying *RoboEarth*, a Cloud Robotics infrastructure, which aims to create a World-Wide-Web style database for storing knowledge generated by humans and robots in a machine-readable format.

The potential of a cloud infrastructure applied to robotics is evident especially for applications that require high computational resources, quick access to vast remote knowledge bases and data repositories and that involve numerous robots - thus requiring high scalability and multi-agent management.

This is well highlighted in all the cited works, however most of these works do not provide out-of-the-lab examples of cloud-based robotics applications. We believe that the design of large-scale, cloud-based robotics applications is already possible in those environments that offer ubiquitous high-bandwidth connectivity, such as urban environments. To prove this statement we have proposed *Fly4SmartCity*, an UAV-based service meant for being used in smart-city scenarios, running on a modular and scalable cloud-robotics platform; and we have described its preliminary architecture in Ermacora et al., 2013

The service is validated using a real test case:

1. A user sends a request to the remote platform via a mobile application or a web-based Graphical User Interface (GUI),

2. The cloud platform parses the request and arranges the mission based on the request and on the available information about the scenario,
3. During the mission, authorized users can access the service at any moment to monitor in real-time the main parameters of the mission, the status of the UAV(s) and to access the video streaming coming from the vehicle.

Fly4SmartCity supports various missions requests, including monitoring, emergency-management, delivery and user-defined flight plan. This is a major difference with respect to the work described in Ermacora et al., 2013, in which just a specific emergency-management service was presented. In the mission planning phase, useful information about the operating scenario can be obtained using open-data. The Open Data Oriented MIssion Planner (ODOMI) module (explained in detail in Section 3.3) is able to access a number of open data provider on the internet and to exploit available information (e.g. position and height of obstacles, quality of the internet connection etc.) to plan, for example, a flight path that avoids connectivity losses and that provides enough signal coverage for the video streaming.

The cloud platform is based on the Robotic Operating System (ROS) (Quigley et al., 2009). ROS is an open-source, meta-operating system for robot software development and it is nowadays becoming the de facto standard for robotic software development. The platform is also designed to be resilient: an emergency flight plan (containing a reduced set of basic instructions such as "land on your current GPS position" or "come back home") is stored on the physical memory of the UAV and automatically started in case of failure. Finally the platform is able to reconfigure at any moment the mission and update the path while the UAV is in flight.

3.1 USER INTERFACE

In order to interact with the monitoring and emergency management service, two Graphical User Interfaces have been implemented. The first one is a mobile application from which the user (e.g., a citizen in danger) can request the emergency service. The second one is a web-based graphical interface for the management of the service by authorized users or operators (e.g., police officers).

The mobile application is an Android app consisting of a simple button. Once the user presses the button, the application sends a GET request, over HTTP (HyperText Transfer Protocol), to the service server, which is running in the cloud robotics platform. The request contains the GPS coordinates of the user at the time of the emergency call and a unique identification number (ID). The mobile application is shown in Figure 22.

The second user interface was written in HTML5 (HyperText Markup Language 5) and Javascript and is accessible via a web browser by authorized operators. The operator can monitor the telemetry of the UAV, whose position is displayed on a city map, and a video stream coming from the UAV to offer assistance to the person in emergency. Figure 23 shows the web-based GUI. The left panel shows a video streaming coming from the UAV on the top, and flight information (speed, altitude, battery level, etc.) on the bottom; the right panel shows all the information about the current mission. In particular, this panel shows the starting point for the UAV, the desired destination, and the path to reach the destination, overlaid on a map

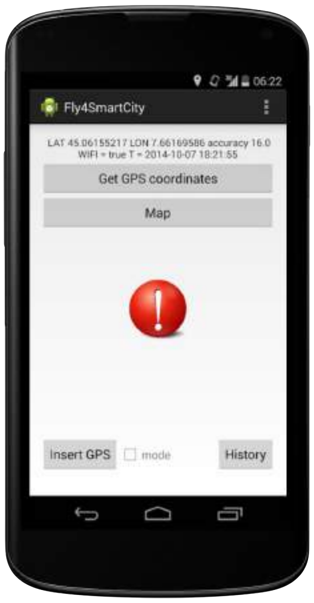

06:22
Fly4SmartCity
LAT 45.06155217 LON 7.66169586 accuracy 16.0
WIFI = true T = 2014-10-07 18:21:55
Get GPS coordinates
Map
Insert GPS
mode
History

Figure 22: The mobile application.

of the city. Controls on the top right panel allow the user to create, delete and modify missions at run-time.

3.2 THE CLOUD PLATFORM

The application is based on a cloud robotics platform, which in turn is based on ROS and has been designed to be generic and compliant with the Platform-as-a-Service (PaaS) paradigm presented in Rapyuta (Hunziker et al., 2013) and RObotics in CONcert (ROCON) (*Robotics in Concert* 2013).

The cloud robotics platform proposed in this chapter has been designed in order to abstract the hardware and software layers, to be robust in case of failures, to offload demanding computations and finally to expose simple

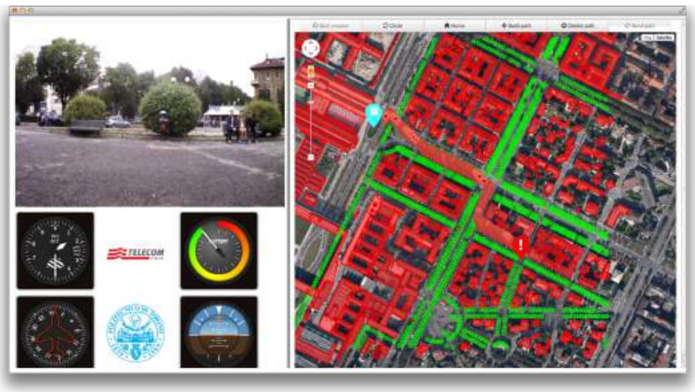


Figure 23: The web graphical user interface.

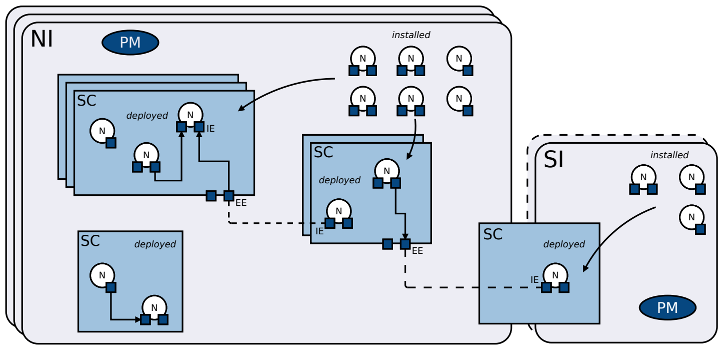


Figure 24: The platform objects and their relationships.

APIs to the final user. These APIs are built using a set of best practices and guidelines commonly known as REpresentational State Transfer (REST).

The cloud robotics platform is important to guarantee the robustness needed for long-term operativeness in a real-world application. The infrastructure knows the state of every node of the system and is in charge of distributing the computational load in a remote location which is able to provide better computational performances than the robot's onboard PC.

The basic elements of the developed cloud platform, shown in Figure 24, are:

- *Node (N)*: it represents the "building block" of a platform service. Can be *installed* if it resides in an instance, or *started* if it resides in a service container. It is *connected* if it has internal or external endpoints.
- *Internal Endpoint (IE)*: it connects a node to other nodes in the current service container, or connects a node to an external endpoint.
- *External Endpoint (EE)*: it belongs to a service container. Connects nodes belonging to different service containers.
- *Service Container (SC)*: it groups a set of nodes into a service.
- *Instance*: it is the object where the platform manager (PM) and the elements described before reside. The instance can be: *Normal* (NI) when it contains a SC and installed or started nodes; *Simple* (SI) when it does not contain any SC but only installed nodes. To start the nodes the SI calls a SC inside a NI.

Services for enabling robotics applications are built using these objects. This can be enacted by starting Nodes (N), which are installed in Instances (SI, NI), into Service Containers (SC). The platform services can be accessed by service APIs and can be built by management APIs.

The *Platform Manager* (PM), shown in Figure 25, is in charge of handling the objects described before. The PM can send and receive commands through the *Command Manager* (CM) object. It can also listen to, and create events through, *Event Manager* (EM) object. Events and commands are accessed through the *Platform API Manager* (PAM) object.

The *Event Engine* (EEn) has a set of controlled counteractions triggered when previous configured classes of events occur. This has been conceived to make the platform service robust and resilient. The counteractions can be both service commands (e.g. publish a message) and platform commands (e.g. create a SC). The first ones are accessed by Service API Manager while

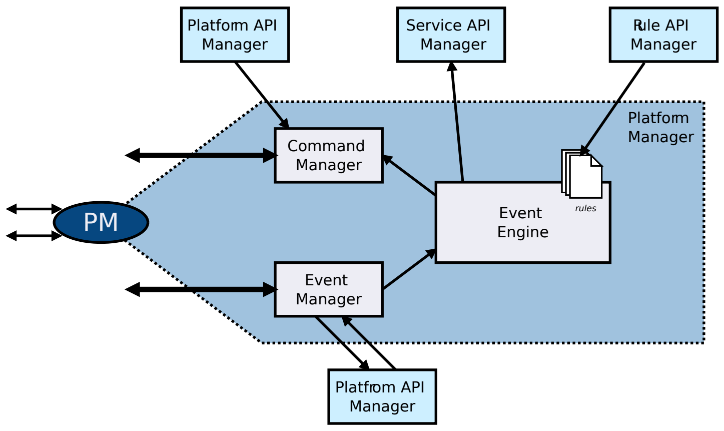


Figure 25: The Platform Manager logic architecture.

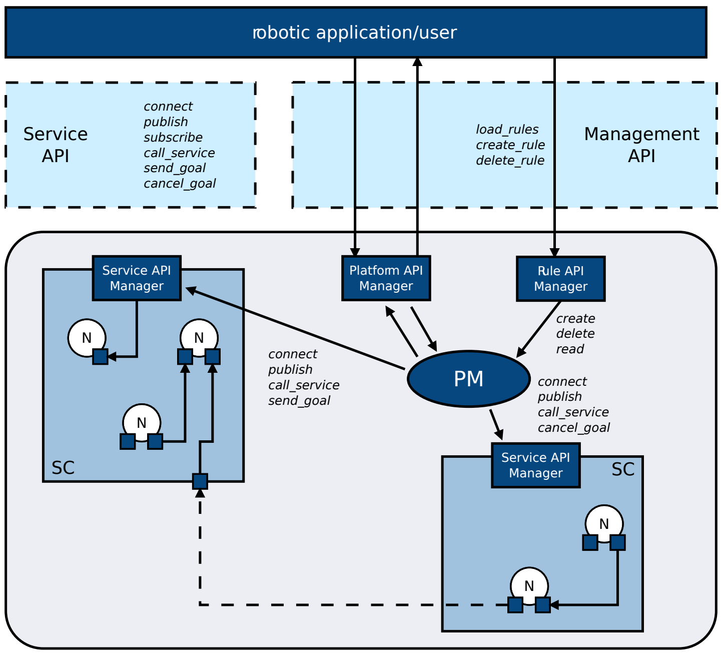


Figure 26: The APIs.

the second ones are directly accessed by PAM. The counteractions can be created, read and deleted from users and applications throughout the *Rule API Manager* (RAM). The *Service API Manager* (SAM) is a special node that needs to be started in a SC. It exposes APIs to external world for managing service commands and events.

External elements, such as robotic applications and the EE, access different kinds of APIs (Figure 26). In particular:

- *Platform API Manager:* it exposes APIs to the user to manage the platform commands and events.

- *Rule API Manager:* it exposes APIs for the management of the event engine.

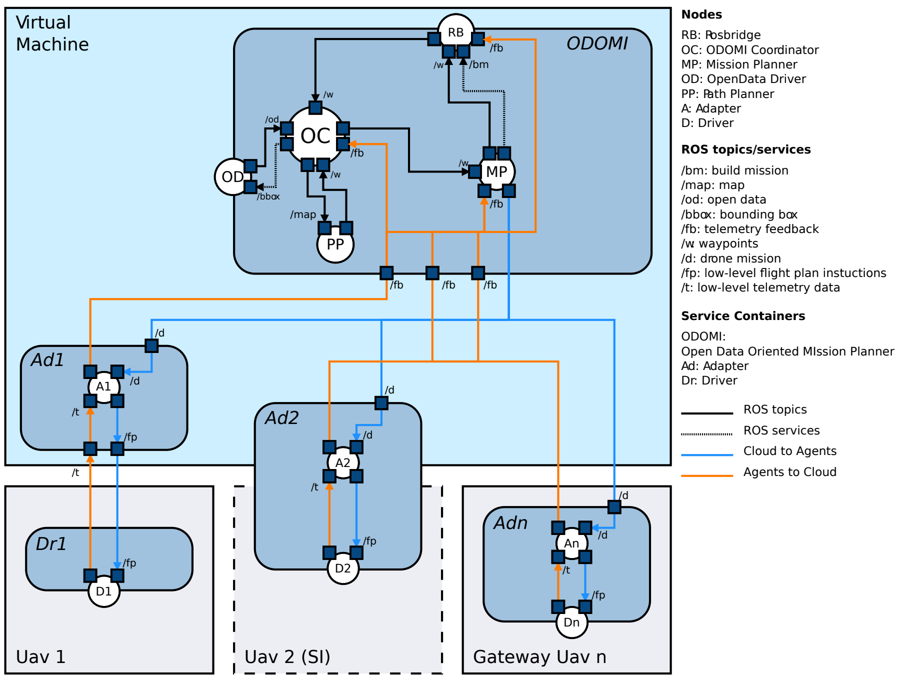


Figure 27: Implementation of our service in the cloud robotics platform. Sharp-edged rectangles represent Normal Instances, rounded rectangles represent Standard Containers, dotted-edged rectangles indicate Simple Instances. Small circles represent ROS Nodes and Endpoints are shown as small squares.

3.2.1 Implementation of the *Fly4SmartCity* service

The concepts outlined in previous chapters have been implemented in practice as follows:

- Normal Instance (NI): a real or virtual machine with high performance.
- Simple Instance (SI): a real or virtual machine with low performance.
- Service Container (SC): a ROS container identified by its ROS master.
- The multi-master Robotics In Concert technology (*Robotics in Concert* 2013) enables ROS container multiplicity.
- Node (N): a ROS node.
- Internal Endpoint (IE): ROS topic, ROS service or ROS action.
- External Endpoint (EE): ROS topic, ROS service or ROS action of a node in another Service Container.

Figure 27 shows our implementation of the cloud robotics platform. Sharp-edged rectangles represent NIs, rounded rectangles represent SCs, dotted-edged rectangles indicate SIs. Small circles represent ROS nodes and Endpoints are shown as small squares. The figure also highlights the difference occurring between Normal and Simple Instances. In the case of UAV 1, the Normal Instance contains a Service Container running the driver node, while the UAV 2 constitutes a Simple Instance, since its service container is shared with the normal instance named "Virtual Machine". In this case UAV

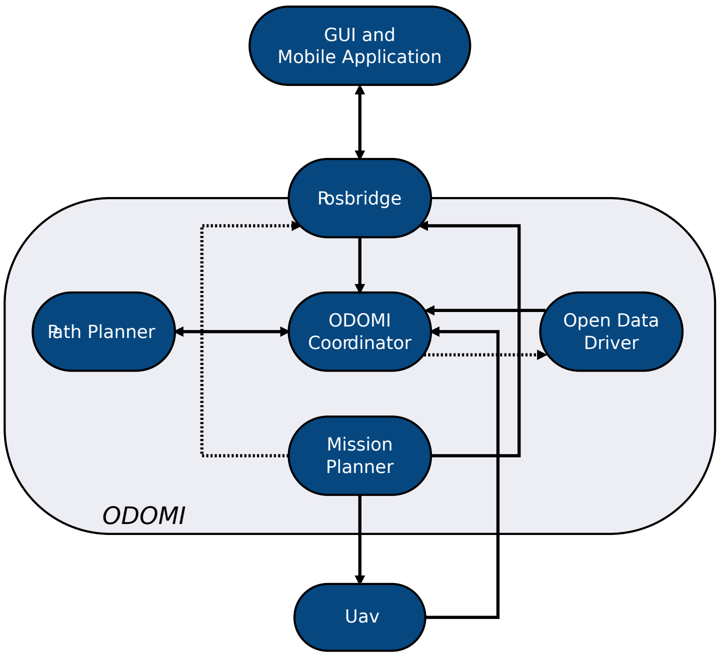


Figure 28: High-level architecture of the mission planner. Solid arrows represent communication over ROS topics, dotted arrows represent ROS service calls.

n is the Service Container (named Adn) and runs in a Gateway (ground station). Here, both the adapter and the driver nodes are placed on the same physical machine. The fourth Normal Instance shown in the figure, named "Virtual Machine", contains both the Open Data Oriented MIssion Planner (ODOMI), which will be presented in the next section, and the Adapter (Ad1 and Ad2) Service Containers. Adapter SCs translate ROS messages coming from UAVs into platform messages. ODOMI is the core Service Container of the whole platform, and it runs five nodes: the ODOMI Coordinator, the rosbridge node (the Service API manager described in previous section), the Mission Planner, the Path Planner and the Open Data Driver (described in the next section). Figure 28 offers a more functional overview of the communication between each module installed on the platform for the single-UAV case. Ensuring a secure access to the platform is fundamental. API access is strictly regulated by means of specific security keys provided to the developer after a mandatory registration phase. Moreover, the instances are connected to the platform via VPN in order not to be easily accessed by malicious users and softwares.

3.3 OPEN DATA ORIENTED MISSION PLANNER

ODOMI is the service container in charge of creating flight plans for the UAVs. The main idea is to aggregate different open data and other online sources of information in order to provide different path planning strategies for unmanned aerial vehicles. ODOMI is the only service container who has the right to START, STOP or ABORT the mission. It is composed by five nodes as shown in Figure 28:

Table 3: Open Data providers.

Provider	*Data*	*Response*	*License*
OpenStreet Maps	2D Map		Open Data Commons Open Database License (ODbL)
Open Weather Map	Temperature, wind, humidity	XML/JSON	Creative Commons CC BY-SA
Geoportale Torino	2D Map, height, pedestrian areas, rivers and waterways, treed areas, green areas	GML(XML)	Creative Commons CC-BY 2.5 IT
5T Torino	traffic, tram and train lines	XML/CSV	Creative Commons CC0 1.0 (Public Domain)

Table 4: Geo-referenced data sources.

Provider	*Data*	*Response*	*License*
Google Maps	Digital Elevation Map	XML/JSON	Google Maps API licensing
OpenSignal	Average RSSI, tower Info	XML/JSON	OpenSignal API licensing

- *Open Data Driver (ODD)*: this module retrieves available information about the mission scenario by calling the appropriate providers. It packs the retrieved data into a ROS message and sends it to the coordinator module.

- *Path Planner (PP):* this module receives the costmap generated by the coordinator and plans the path (as a simple succession of waypoints) that the UAV has to follow in order to successfully accomplish the mission. The module is described in more detail in the next Section.

- *RosBridge (RB):* it provides JavaScript Object Notation (JSON) APIs to access ROS topics and services (Crick et al., 2011). This node serves as the interface between the smartphone application and the web GUI on one hand, and the robotics platform on the other hand, via the Service API Manager that exposes the APIs for sending the goal coordinates that the UAV has to reach to the Mission Planner. The GPS coordinates are sent as a ROS message.

- *ODOMI Coordinator (OC):* this module coordinates all messages in the ODOMI service container. It is in charge of the coordination of the whole architecture and it creates the mission for the UAV according to the suitable policy.

- *Mission Planner (MP):* This module is responsible of choosing the best available UAV for the mission. It receives the set of waypoints and packs them in a custom message to be sent to the chosen UAV.

3.3.1 Open Data Driver module (ODD)

The ODD module connects and retrieves information made available on internet by several Open Data providers as shown in Table 3. Additional geo-referenced data, shown in Table 4, are used to add even more information to the Mission Planner; some of these data belong to private companies and are usually subject to restrictive license and limited access, hence they cannot be defined "open" in a strict sense (*Open Knowledge Foundation Blog. Defining Open Data* 2013), however they are typically publicly available under some constraints (e.g., maximum number of daily/monthly API calls).

Given the current position of the UAV and a goal, the Open Data Module creates a bounding box which includes these two points, plus a suitable padding in all directions, and gathers open data and other geo-referenced data as shown in Tables 3 and 4.

It should be noted that a driver must be created for accessing each different data source, as open data and other sources do not necessary comply to a standard format. As of now, in our implementation we developed the drivers for Geoportale Torino, OpenStreetMaps, 5T Torino, Google Maps and OpenSignal.

When all the data have been retrieved, they are sent to the OC, which will create a cost-map for the Path Planner module (as explained in the next Section).

3.3.2 ODOMI coordinator (OC)

This module receives the current position of the UAV and the goal from the web GUI or the mobile application. It then sends a request to the ODD module that answers with raw information, gathered from open data. The coordinator module creates a costmap using these data, according to the chosen policy of the mission. Policies define the cost to be optimized by the Path Planner module for the creation of the path. These policies can be chosen by the user according to the desired service. In general, policies that maximize flight over buildings and trees are thought for real applications in urban environments, in order to minimize the risk of hurting humans or things in case of failures. The coordinator then sends the costmap to the Path Planner module.

3.3.3 Path Planner module (PP)

The PP module exploits the data provided by the ODD module as explained in the Section 3.3.1. Once defined, the flight plan is provided to the Mission Planner that will send it to the agents in order to start the mission.

Path planning or trajectory planning is a well-known problem in mobile robotics, and it has been recently applied to UAVs. Path planning is one crucial task for UAVs that have semi-autonomous or autonomous flight capabilities. As technology and legislation move forward, the need arises for an increase in their level of autonomy. Therefore, the focus is moving from system modeling and low-level control to higher-level mission planning, supervision and collision avoidance, shifting the perspective from vehicle constraints to mission constraints (Gutiérrez et al., 2006). UAVs present some peculiar characteristics, like their flight dynamics, 6DOF movement, and high levels of uncertainty in their own state knowledge, as well as limited sensing capabilities (Goerzen et al., 2010). However, other problems have already been addressed by the robotics community, like partial knowledge of the environment. The autopilot is in charge of low-level vehicle control. Some work has been carried out in Jun and D'Andrea, 2003 for path planning in presence of uncertainties and with signal constraints in Grøtli and Johansen, 2012, Grancharova et al., 2014, Chi et al., 2012. Most of path planning approaches rely on a two-stage procedure: they first solve what is called the global path planning problem, by finding a feasible trajectory given the current pose of the agent, the destination and a map of the environment. They implement then a control strategy called local path planning to follow the found trajectory (Goerzen et al., 2010).

In our work we focus on global path planning, since our objective is the generation of a trajectory, which will be decomposed into a series of intermediate waypoints, which in turn will be fed to the UAV's autopilot. In our application, in order to simplify our case study, we also assume that the UAV flies at a fixed altitude; this is also motivated by the fact that most of the available open data are basically 2D and do not provide useful information for 3D navigation. Based on the assumption of having a fixed altitude, we simplify the problem to the 2D case and we use the implementation of the D*-Lite algorithm described in Koenig and Likhachev, 2002. D*-Lite is a fast implementation of the D* algorithm, and is based on Lifelong Planning A* (LPA*) (Koenig and Likhachev, 2001). D* is an incremental heuristic graph search algorithm which is able to deal with incomplete information and observations. D* algorithm ensures completeness and optimality properties. The Path Planner (PP) module is implemented as follows. Given a bounding box area of operation, we first build a static cost-map. Given the current position of the UAV and the desired goal, and given a planning strategy, we calculate the optimal path using D*-Lite. The Path Planner can also force the UAV to fly on areas which are considered "safer" from the citizen's point of view, like roofs, trees and water surfaces. This problem is often referred to as *risk-aware path planning*, and refers to techniques that try to minimize the risk involved with autonomous flight. Although in risk-aware path planning the "risk" often refers to the risk of hitting obstacles (De Filippis et al., 2011, Carpin S., 2014), it can also be considered as the risk of hitting people in case of hardware or software failures of the UAV. This choice is motivated by the high interest in risk minimization which is stressed in both national and international regulations (e.g., see ENAC regulation, 2014 for the Italian regulations). Currently, we implemented three strategies. All those strategies try to minimize path length.

In addition to that, the first strategy also avoids obstacles represented by buildings which are higher than the flying altitude of the UAV.

The second strategy tries instead to minimize risk of damages in case of faults. Areas which are not covered by buildings or trees are considered as risky areas (the UAV can cause more damage falling in case of hardware failures).

The third strategy tries to minimize risk and at the same time to maximize network connection, so an higher cost is given to areas with no Long Term Evolution (LTE) coverage. In our application we use LTE Received Signal Strength Indication (RSSI) heat-maps. This information is again given by the ODD Module. The cost of each cell is based on the RSSI for that particular cell; if the RSSI is below a critical threshold, the cell is marked as obstacle, since in our application we cannot afford to lose connectivity. We then extract a series of waypoints from the path using a simple procedure. First we compute the distance transform of the map. For each cell p we compute the distance of the nearest obstacle as

$$D_P(p) = \min_{q \in P}(d(p, q)),$$

where P is a grid of points representing obstacles and $d(\cdot)$ is the distance between two points. Then, for each waypoint we expand a circle around it with a radius of value $D_P(p)$, where p is the cell corresponding to the waypoint. The next point of the trajectory that lies on the circumference is taken as the next waypoint. The process is then repeated until the last point of the trajectory. With this procedure we extract a list of waypoints that are given to the UAV's autopilot.

3.4 THE AGENTS

Only quadrotors have been chosen as the agent for validating the overall system in its first stage, since they offer a series of advantages with respect to other possible architectures (e.g. fixed wing autonomous aircrafts etc). Their VTOL (Vertical Take Off and Landing) capabilities allow for quick take-off and landing operations in busy or space-constrained places. The possibility to hover above the target makes them well-suited for surveillance and monitoring tasks. Moreover they offer a simpler mechanics and require less maintenance with respect to a standard helicopter, while offering similar autonomy and payload. However a quadrotor is an inherently unstable system (G.M. Hoffmann et al., 2007), and for this reason it requires a fast on-board controller to guarantee its stability in standard flights (Bouabdallah and Siegwart, 2005). This controller typically runs on a dedicated electronic board, equipped with a number of inertial sensors (accelerometers and gyroscopes) to detect the attitude angles of the quadrotor frame and its rotational velocities; barometers and sonar sensors (for altitude measurements), GPS receivers (for autonomous waypoint navigation) and magnetometers (providing an estimate of the heading of the aircraft) are other common devices, often mounted on the same board.

For the validation of the proposed cloud robotics service both a *MicroPilot*'s *MP*2128^g and a *Parrot AR.Drone* have been used.
MP2128^g[1] is an auto-pilot board embedding all the peripherals needed for a stable and autonomous quad-rotor flight. This auto-pilot is specifically addressed to professional use and is closed-source software. In order to write customized code the user must purchase an add-on set of APIs. These functions constitute a dedicated dynamic linking library that acts as a intermediate layer between the user code and the autopilot software. Using the functions encoded in the library the developer is able to get access to several low-level parameters of the auto-pilot and can modify their values. The board in this case is mounted on a custom carbon-fiber quadrotor frame together with the other peripherals needed for the flight (four motors, the same number of motor controllers and a battery).
The AR.Drone[2] is a commercial ready-to-fly quadrotor solution, controllable via smartphone. It features a front HD camera and the flight stability is ensured by a mother board (running a real-time linux-based operating system) and a navigation board interfaced with the on-board sensors (two cameras, ultrasonic range finders, gyroscopes and accelerometers). The AR.Drone is mainly conceived for gaming applications, amusement and augmented reality video games, but due to its low-cost and the availability of an official SDK (Software Development Kit), it gained a very good popularity in the academic community.

1 http://www.micropilot.com
2 http://ardrone2.parrot.com

3.5 EXPERIMENTAL RESULTS

We tested our approach in several different simulated scenarios, as well as in the real ones. In this section we show and comment some of these results. The first two tests show the integration of different open data in ODOMI in simulated scenarios; the third test shows the integration of ODOMI, the PP module and the AR.Drone commercial quadcopter; the last test shows a live demonstration of the whole system (GUI, cloud platform) using a commercial quadcopter in a private flying area.

Simulated scenarios are based on real data gathered from open data (buildings perimeter and height, trees and waterways) and crowd-sourcing using a mobile Android app (3rd Generation (3G) and 4th Generation (4G) coverage). The only difference with real experiments is that we only generate a path but we don't actually fly a UAV. This is due to current legal restrictions, as stressed along the dissertation, and also to security reasons.

The system was implemented using C++ and Python on GNU/Linux. The tests have been carried out on a standard PC. The source code for the ODOMI mission planner is available online[3], together with the web-based GUI[4].

3.5.1 Test 1

In this test we show a simple planning strategy in a simulated scenario, in which an agent has to fly in an urban area. The Path Planner tries to minimize path length, while avoiding obstacles. Since the simulated AR.Drone quadrototor can fly using its onboard GPS receiver at a maximum altitude of only 6 m, obstacles are represented by buildings which are higher than the flying altitude of the UAV, as well as trees. Figure 29 shows the results. The starting position for the UAV and the final goal are pointed by markers. The small markers show the extracted waypoints.

3.5.2 Test 2

In this experiment we compare three different planning strategies among the possible ones, in a simulated scenario. All three strategies try to minimize path length. The first strategy avoids obstacles represented by buildings which are higher than the flying height of the UAV. The second strategy instead considers risk factors. Areas which are not covered by buildings or trees are considered as risky areas (the UAV can cause more damage falling in case of hardware failures). The second strategy tries to balance path length and to minimize travel over risky areas. The third strategy also takes into account 4G coverage. Areas with no 4G coverage are also considered "risky"; the traversal cost for these areas is treated as in strategy two. Figure 30 shows the results.

3.5.3 Test 3

We show a real use case of our system running on a real low-cost platform. The platform is an AR.Drone Parrot 2.0. The UAV is equipped with a GPS module (Parrot Flight Recorder). The GUI is running on a laptop, connected

3 https://github.com/fly4smartcity/odomi
4 https://github.com/fly4smartcity/rendezvous

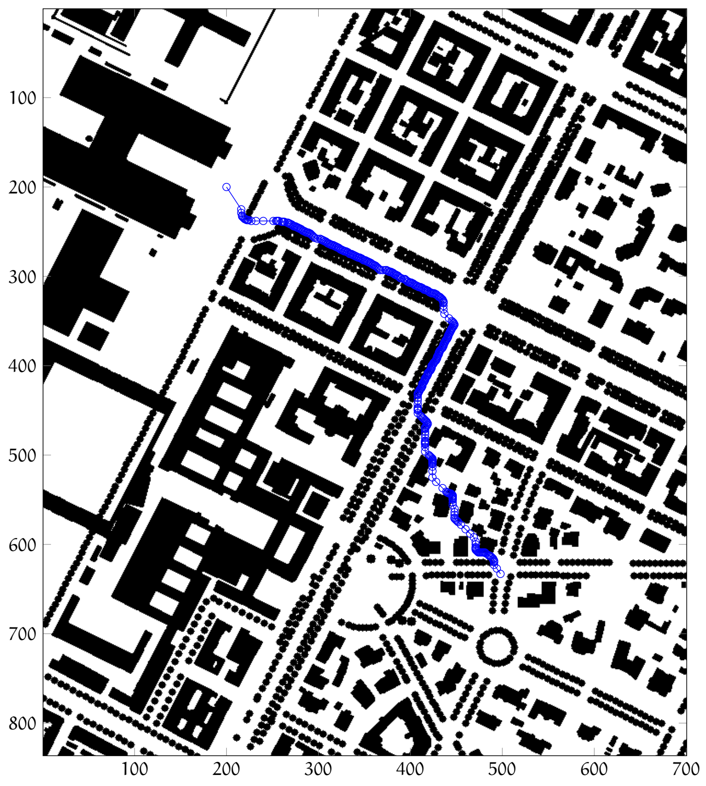


Figure 29: Simulated experiment. Obstacles from open data are shown as black cells; computed waypoints are shown in blue. The starting point is in the top-left and the goal in the bottom-right.

to the cloud platform via a 4G link. Communication between the Parrot and the laptop is done using ROS over Wi-Fi. Given the initial position of the UAV, the user selects a goal. An obstacle map is created using open data and a feasible trajectory is found. Then, waypoints are automatically sent to a guidance system.

In Figure 31 we show the results of a typical experiment. It should be noted that GPS localization of the Parrot has an average error of up to 5-10 m in an area surrounded by buildings like the one we show in this experiment. For this reason we inflated the obstacles by 5 m. Due to its poor GPS accuracy and wind dynamics, it can be seen that the Parrot is not able to follow the given trajectory accurately, but it is still able to avoid obstacles. Figure 32 shows the experiment in progress. It can be seen that the AR.Drone was able to reach the goal only within a large tolerance circle (10 m) due to the inaccuracy of GPS localization in narrow areas; for the same reasons it was not able to correctly follow the trajectory.

3.5.4 Test 4

Finally, we present the results of a live demonstration in which we tested the whole system. A commercial quadrotor was used in this case, with a MicroPilot's MP2128[9] autopilot managing the in-flight stabilization of the UAV. The live demonstration was performed in Aero Club Torino's airport in Turin, Italy. The UAV was connected both to a ground station via a standard Radio Frequency (RF) 5.8 GHz radio link and to the cloud robotics platform via a 4G LTE link. The flight was performed at line-of-sight and

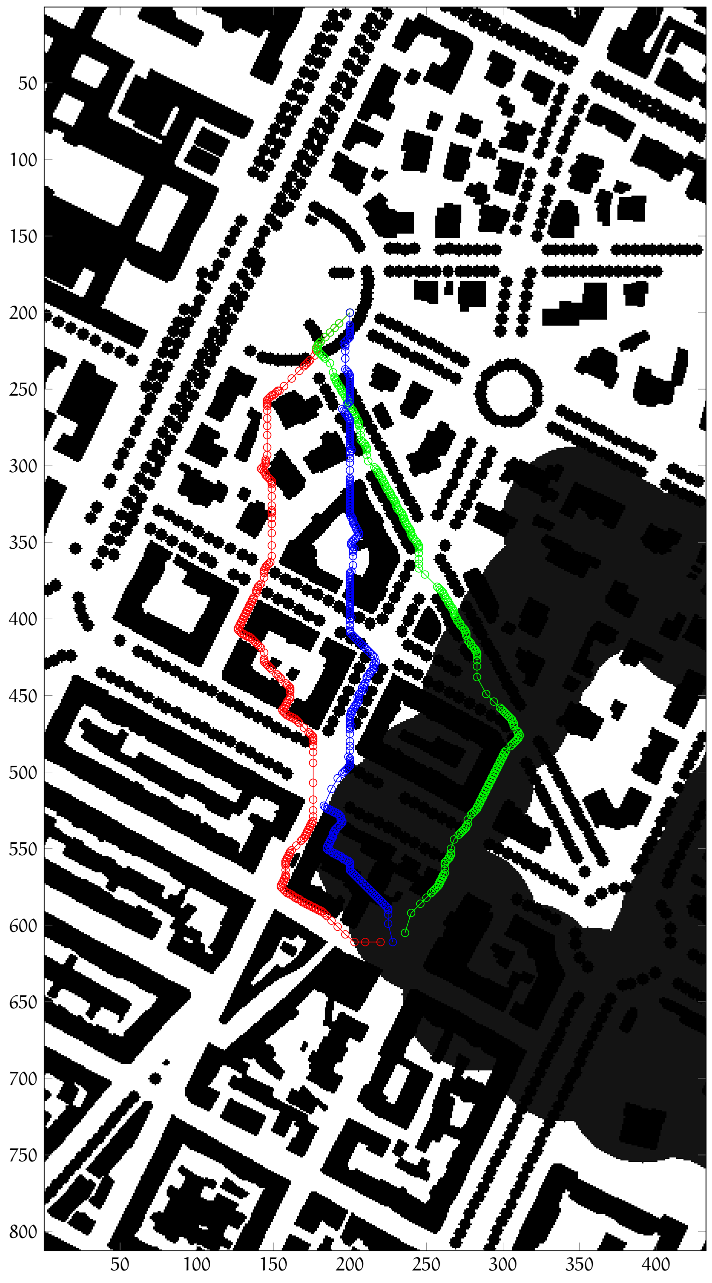


Figure 30: Comparison between three planning strategies. Black cells represent obstacles, grey cells represent areas with no 4G connectivity. Green line: optimize for shortest path, avoid obstacles, ignore connectivity; blue line: optimize for shortest path and maximize travel over roofs and trees for safety, ignore signal; red line: same as blue line, but prefer areas with 4G connectivity. The starting point is in the top-left and the goal in the bottom-right.

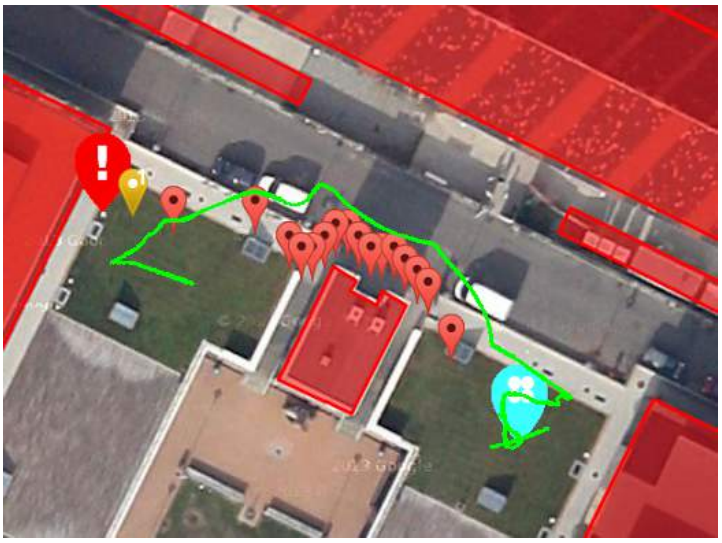


Figure 31: Real test over a short trajectory. Obstacles from open data are shown in red; starting point and final goal are shown as markers; computed waypoints are shown as red markers. Real trajectory followed by the UAV is shown in green.

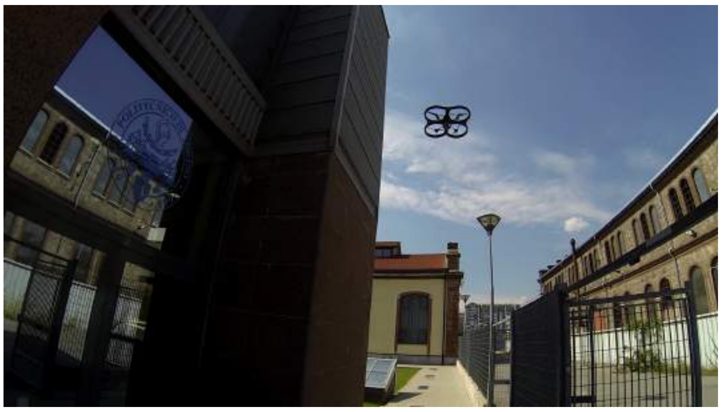


Figure 32: Real test over a short trajectory. Photo of the experiment.

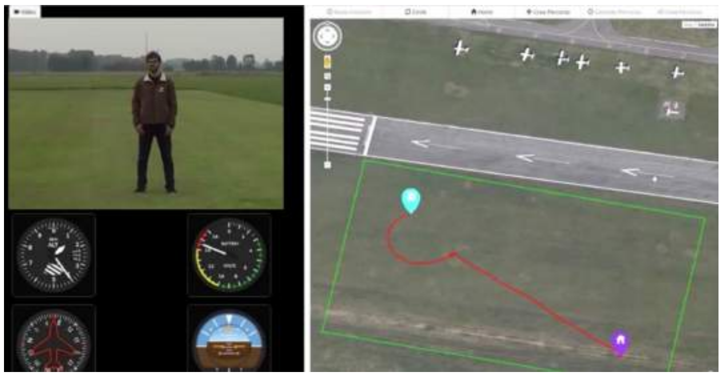


Figure 33: Video streaming and mission information during the live demonstration (first test-case).

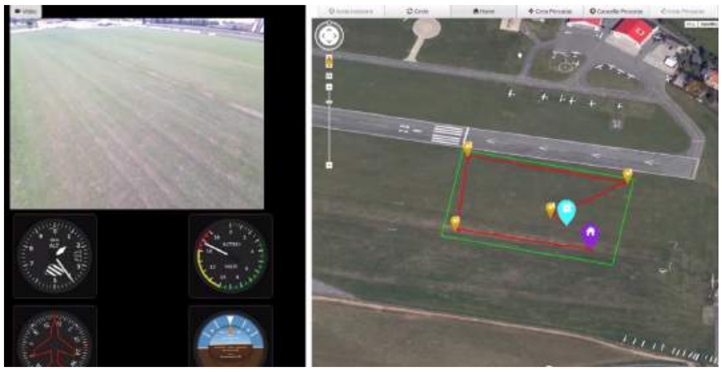


Figure 34: Video streaming and mission information during the live demonstration (second test-case).

a pilot was ready to take control of the UAV in case of trouble, as required by the current Italian legislation. In addition to that, the flight zone was segregated and the spectators were kept at a distance of 150 m.The live demonstration was divided in two test-cases.

In the first one, a user requests the emergency service using a mobile phone. The authorized user authorizes the flight using the GUI. The UAV reaches the user and performs a circle around him and sends back the video streaming and its telemetry. A screenshot of the GUI during the experiment is shown in Figure 33.

In the second case, the authorized user selects a set of waypoints using the GUI, by clicking on the map. Then, the user authorizes the mission and the UAV performs the mission and sends the video streaming and its telemetry to the GUI. A screenshot of the GUI during the experiment is shown in Figure 34. The live demo described in this section has been authorized by Ente Nazionale Aviazione Civile (ENAC), the Italian civil flight authority. Videos of the experiment are available online[5].

5 http://tinyurl.com/l5lbry4, http://tinyurl.com/qd2whr5

3.6 CONCLUSION

Fly4SmartCity is a cloud-robotics service that aims to prove that a real world application of cutting-edge technologies in robot-networking and unmanned vehicles is actually possible. The goal of *Fly4SmartCity* is to provide a service for emergency management in smart city environment; this is achieved by the strict integration of a capable, reliable and stable cloud platform; small agile autonomous agents; a high-bandwidth, low-latency mobile network; a rich set of data drawn from various providers; a well-documented and supported robotics middleware. A ROS-based cloud-robotics platform has been expressly developed to support the service and the main concepts of its implementation have been discussed. Then the service itself has been presented and the software modules have been described from a functional point of view. Several major technical issues (e.g. autonomy of the UAVs) and legal restrictions imposed by the civil flight authorities still prevent a common and practical use of the presented system, however we have proved here how information about the urban fabric, that are in large part of public domain in a smart city, offer a valuable source that it is possible to exploit in order to guarantee safer flights and operations. We showed how this information can be integrated into a simple 2D path planning module able to generate feasible flight trajectories that are also complaint with safety and signal quality constraints (e.g., fly over rivers when possible, avoid crowded areas, avoid areas with low mobile signal strength, etc.). Finally, the cloud platform has been designed not to be tied with the described service. The source code for our system is available online.

4 SENSOR FUSION

One of the most promising and active field of robotics is related to the problem of sensor fusion. Sensor fusion can be defined in general as the combination of sensory data (or data derived from sensory data) with the goal of obtaining a *better* representation of the state of a system than the one that would be possible if the data sources were used individually. This definition includes equally measurements coming from multiple sensors but also measurements produced by a single sensor at different time instants (Mitchell, 2007). One of the most relevant use-cases of sensor fusion in UAV applications is the fusion of the inertial measurements coming from the vehicle Inertial Measurement Unit (IMU) (accelerometers and gyroscopes) for obtaining a robust estimate of the three orientation degrees of freedom of the aircraft. Another possible example could be the fusion between the gyroscopes and a magnetometer sensor for getting a high-rate estimate of the global heading of the vehicle, less susceptible to magnetic error and interferences. This chapter will focus on some preliminary work aimed to introduce some sensor-fusion capabilities in the UAVs developed for both STEPS and F4SC projects. The activities here described are indeed relevant for the future developments of the two projects and have as a common goal the fusion of the data coming from a monocular-camera Visual Odometry (VO) algorithm, the GPS and the IMU, to achieve a stable and reliable *pose* (position + orientation) estimate with reduced effects of uncertain and erroneous measurements. The fusion is performed by the Multi Sensor Fusion (MSF), as described in section 4.1. The main characteristics of the employed sensors are listed in table 5, together with the advantages and disadvantages of each of them.

Table 5: Advantages and disadvantages of the sensors fused using MSF.

Sensor	$-$	$+$
Accelerometers	noisy	fast (100 to 1000 Hz)
Gyroscopes	noisy temporal drift	fast (100 to 1000 Hz)
Camera (VO)	spatial drift	high accuracy (0.01 to 0.1 m) works outdoor/indoor
GPS	low accuracy ($\pm$5 m) multi-path effects works only outdoor slow (1 to 5 Hz)	no drift

4.1 AN INTRODUCTION TO THE MSF

The Multi Sensor Fusion (MSF)[1] is a modular framework for multi-sensor fusion based on an Extended Kalman Filter and developed by the Autonomous Systems Lab (ASL) based at ETHZ. The MSF software expands the potentialities of the Single Sensor Fusion (SSF) framework[2], previously developed by the same group.

An high-level overview of the framework is given in Weiss, 2012, Weiss, Achtelik, et al., 2012 and Lynen et al., 2013.

The MSF allows to fuse information coming from a theoretically unlimited number of sensors managing at the same time:

- delayed measurements
- acquisitions at different frame-rates
- absolute and relative measurements
- on-line estimation of the calibration states of the sensors (self calibration)

Furthermore the framework is explicitly designed and optimized for being used with micro UAVs.

For these reasons in this work we have decided to integrate the ASL's MSF as part of the existing software stack, being the development of a new framework out of the scope of the projects. In particular the MSF will be used to fuse the inertial sensors (accelerometers and gyroscopes) with the GPS receiver and a vison sensor (monocular camera).

4.2 COMBINING MULTIPLE SENSORS

The MSF uses an Extended Kalman Filter to fuse the information coming from several sensors at run-time following a *loosely-coupled* approach (see figure 35); this means that the sensors are considered as independent modules exchanging information at different data rates with the module processing the fusion. On the contrary, in a tightly-coupled approach, the raw measurements coming from different sensors are not processed independently and are used directly to compute the final estimate (Corke et al., 2007). The use of a loosely-coupled approach makes it possible to consider the sensors as black-boxes and also allows the final user to integrate new sensors without any modification of the fusion part of the code (*core*).

The multi-sensor fusion algorithm is based on an EKF. The EKF is a generalization of the standard Kalman Filter for non-linear systems, based on a linearization about an estimate of the current mean and covariance. Furthermore the MSF uses an *indirect* formulation for the filter, i.e. it uses the error state vector in spite of the standard one, this is particularly convenient when using *quaternions* as attitude representation parameters (Trawny and Roumeliotis, 2005). In the MSF the states of the system are divided in two categories: *core* and *auxiliary* states. Basically the *core* states are the IMU ones:

$$x_{core}^{T} = \left[{p_{w}^{i}}^{T}, {v_{w}^{i}}^{T}, {q_{w}^{i}}^{T} \right] \tag{22}$$

1 https://github.com/ethz-asl/ethzasl_msf
2 https://github.com/ethz-asl/ethzasl_sensor_fusion

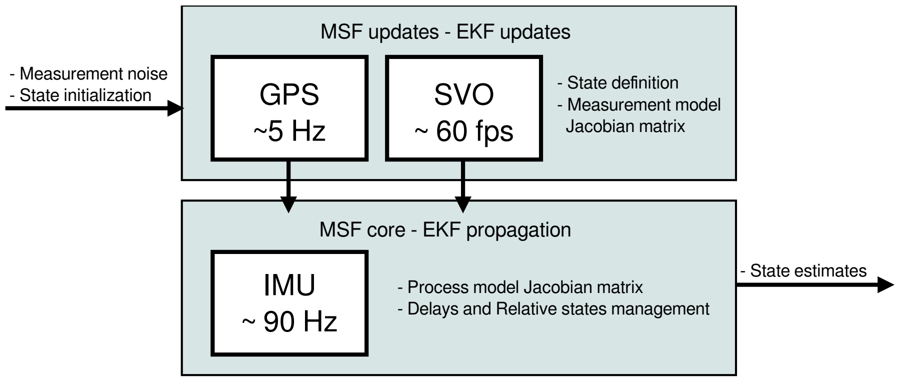

MSF updates - EKF updates
- Measurement noise
- State initialization
GPS
~5 Hz
SVO
~ 60 fps
- State definition
- Measurement model Jacobian matrix
MSF core - EKF propagation
IMU
~ 90 Hz
- Process model Jacobian matrix
- Delays and Relative states management
- State estimates

Figure 35: The loosely coupled approach allows high system modularity, with the MSF core block managing EKF states propagation and an arbitrary number of update blocks participating to the update part of the filter.

where ${p_w^i}^T$, ${v_w^i}^T$, and ${q_w^i}^T$ correspond to the relative position, velocity and orientation of the IMU w.r.t. the inertial (world) frame.

These states are fixed and are used for the *propagation* part of the Extended Kalman Filter. On the contrary the *auxiliary* states can be defined by the user depending on which sensors are currently used in the system (e.g. magnetomer, camera, GPS . . .) and are used for the filter *update* step. This design choice is justified by the fact that usually the measurements from the external sensors are 1 to 3 order of magnitude slower with respect to the IMU ones (usually in the range of 100 Hz to 10 kHz) and in systems with limited computational power performing also the update step at the same frame rate of the inertial sensors would be too demanding; furthermore this choice helps in keeping the code modular.

4.2.1 MSF setup

We introduce here the MSF setup used in the experiments with the following sensor suite:

- IMU
- Monocular camera
- GPS

where the IMU is used to estimate a *pose* (position + orientation) information for the UAV; a Visual Odometry (VO) algorithm is used in conjunction with a monocular camera in order to estimate the *pose* of the UAV and the GPS to get an *absolute position* measurement of the vehicle. The MSF is agnostic with respect to the particular VO algorithm used due to its loosely-coupled approach. Every algorithm is equally usable in the framework as long as it returns a pose information. In our experiment we have used the Semi-direct Visual Odometry (SVO) algorithm, developed by the Robotics and Perception Group at UZH and whose details are illustrated in Forster et al., 2014.

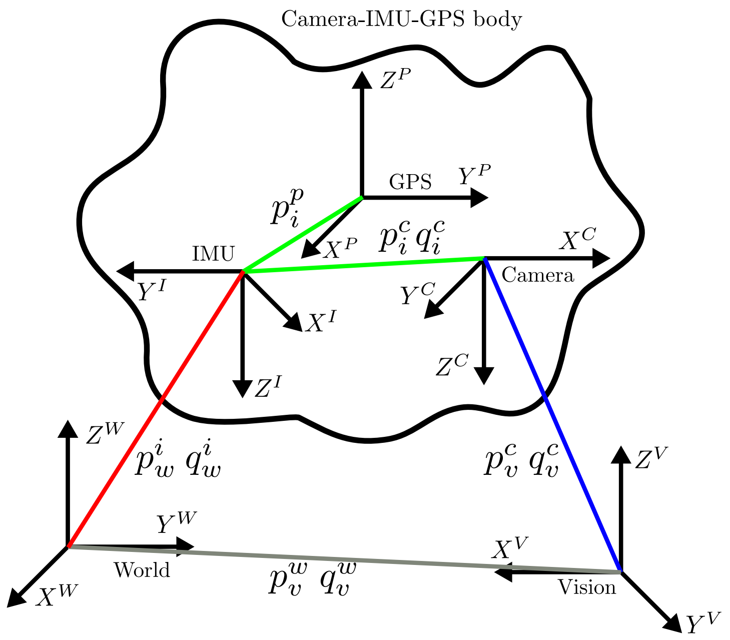


Figure 36: MSF reference frames. Green lines represent roughly constant transformations, red lines denote the variables used for robot control, blue lines show the values of the camera-vision frame transformation measured by the visual framework and grey lines highlight the translational and angular drift in the visual frame with respect to the world frame (Weiss and Siegwart, 2011).

Measurement models

In this section the generic translation from l to m frame will be denoted with p_l^m, similarly q_l^m and $C(q_l^m)$ will denote the quaternion and the rotation from the l to the m reference frame respectively.

IMU In an Inertial Measurement Unit a three axis gyroscope provides measurements of the rotational velocity and a three axis accelerometer measures linear accelerations. We consider the single-axes measurements of angular velocity and acceleration affected by two types of errors: a random additive noise n and a bias b (Trawny and Roumeliotis, 2005).

$$\omega_m = \omega + n_\omega + b_\omega \qquad a_m = a + n_a + b_a \tag{23}$$

Furthermore we model the noise as a *Gaussian White Noise* with the following characteristics

$$E[n_\omega] = E[n_a] = 0 \tag{24}$$
$$E[n_\omega(t+\tau) * n_\omega^T(t)] = R_\omega \delta(\tau) \tag{25}$$
$$E[n_a(t+\tau) * n_a^T(t)] = R_a \delta(\tau) \tag{26}$$

and the bias as a *Random Walk Process*

$$\dot{b}_\omega = n_{b_\omega} \qquad \dot{b}_a = n_{b_a} \tag{27}$$

with characterstics:

$$E[n_{b_\omega}] = E[n_{b_a}] = 0 \tag{28}$$
$$E[n_{b_\omega}(t+\tau) * n_{b_\omega}^T(t)] = R_{b_\omega}\delta(\tau) \tag{29}$$
$$E[n_{b_a}(t+\tau) * n_{b_a}^T(t)] = R_{b_a}\delta(\tau) \tag{30}$$

This model applies independently to each of the three sensors axis.

MONOCULAR VISUAL ODOMETRY SENSOR The measurement model introduced for the camera is the same described in Lynen et al., 2013. It can be divided in its position and angular parts.

$$z_{p_{VO}} = p_v^c = C^T_{(q_v^w)}(p_w^i + C^T_{(q_w^i)} p_i^c)\lambda + p_w^v + n_{p_{VO}} \tag{31}$$
$$z_{q_{VO}} = q_v^c = q_i^c \otimes q_w^i \otimes (q_w^v)^{-1} + n_{q_{VO}} \tag{32}$$

with $C^T_{(q_w^i)}$ being the IMU attitude and $C^T_{(q_v^w)}$ the rotation from the visual frame to the world one, and q_i^c, q_w^i and q_v^w the quaternions describing the rotation of the inertial to the camera frame, the world to the inertial frame and the visual to the world frame respectively (see figure 36). λ is a visual scale factor.

GPS In the following we will assume that the position measurements coming from the GPS receiver installed on the UAV is described by:

$$z_{GPS} = p_w^i + C^T_{(q_i^w)} p_i^g + n_{p_{GPS}} \tag{33}$$

similarly to the IMU case we model the noise $n_{p_{GPS}}$ as a White Gaussian Noise. This is a very simplifying hypothesis, since, as it will be seen in section 4.3.2, GPS satellite transmissions are usually affected by several different phenomena influencing the precision and the accuracy of the position information, and the White Gaussian Noise (WGN) assumption does not model important source of errors (e.g. the multipath effect). Still, this hypothesis makes the integration of a GPS sensor in an EKF a lot easier; therefore we will keep the assumption, aware of the limitations introduced. It is also noticed here that incoming GPS measurements are converted to metric Earth-Centered, Earth-Fixed (ECEF) frame (see figure 37), and the offset from the ECEF measurement to the local ECEF initialization position is subtracted before the measurement gets applied to the filter.

State definition

The prediction part of the EKF is driven by the IMU measurements. We define the following *core* states vector:

$$x_{core}^T = \left[p_w^{i\,T}, v_w^{i\,T}, q_w^{i\,T}, b_w^T, b_a^T \right] \tag{34}$$

where $p_w^{i\,T}$, $v_w^{i\,T}$, and $q_w^{i\,T}$ correspond to the relative position, velocity and orientation of the IMU w.r.t. the inertial (world) frame; b_w^T and b_a^T represent the angular velocity and acceleration bias. The addition of a camera introduces a set of *auxiliary* states:

$$x_{cam}^T = \left[\lambda, p_i^{c\,T}, q_i^{c\,T}, p_v^{w\,T}, q_v^{w\,T} \right] \tag{35}$$

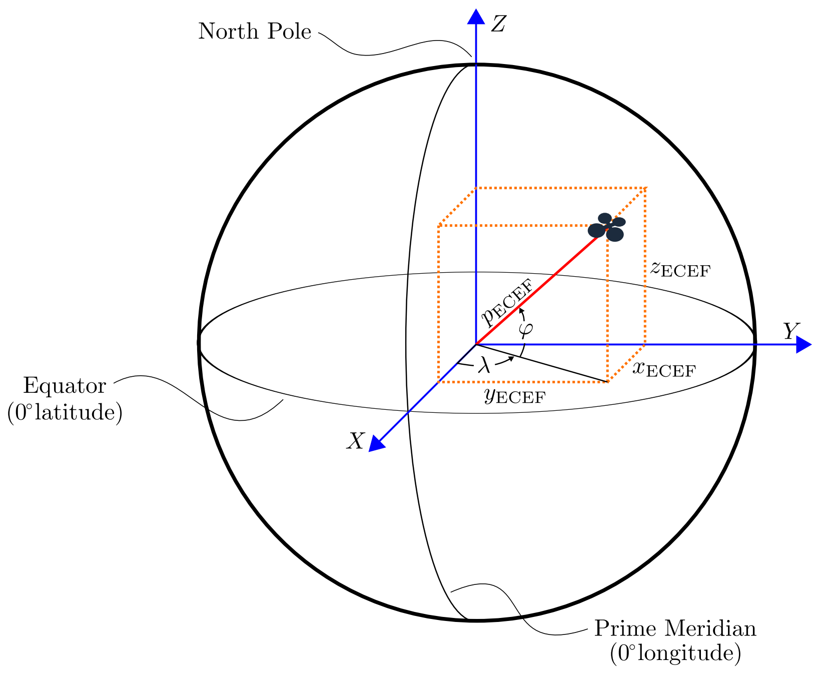


Figure 37: The diagram of Earth-Centered, Earth-Fixed coordinates, in relation to latitude (φ) and longitude (λ).

λ is the visual scale factor and it is not known a-priori when a monocular camera is used. The states ${p_i^c}^T$ and ${q_i^c}^T$ describe the transformation (translation and rotation) between the IMU and the camera frame; this kind of states are also called *extrinsic calibration state*. ${p_v^w}^T$ and ${q_v^w}^T$ represent the position and orientation of the visual frame V w.r.t. the fixed inertial one. These quantities are affected by *spatial* and not *temporal* drift, therefore their (time) derivative can be considered zero (eq. 38i, 38j).

Finally we introduce the auxiliary states added by the global position sensor (GPS):

$$x_{gps}^T = \left[{p_i^g}^T \right] \tag{36}$$

The position sensor state vector only consists in a calibration state ${p_i^g}^T$ representing the translation between the origin of the sensor and the IMU reference frame. This offset does not increase its uncertainty over time, thus we can model it as shown in equation 38k.

System model

The final resulting state vector is:

$$X = \left[{p_w^i}^T, {v_w^i}^T, {q_w^i}^T, b_w^T, b_a^T, \lambda, {p_i^c}^T, {q_i^c}^T, {p_v^w}^T, {q_v^w}^T, {p_i^g}^T \right] \tag{37}$$

while the equations describing the model of the system are:

$$
\begin{aligned}
\dot{p}_w^i &= v_w^i && \text{(38a)}\\
\dot{v}_w^i &= C_{q_w^i}^T(a_m - b_a - n_a) - g && \text{(38b)}\\
\dot{q}_w^i &= \frac{1}{2}\Omega(\omega_m - b_\omega - n_\omega)q_w^i && \text{(38c)}\\
\dot{b}_\omega &= n_{b_\omega} && \text{(38d)}\\
\dot{b}_a &= n_{b_a} && \text{(38e)}\\
\dot{\lambda} &= 0 && \text{(38f)}\\
\dot{p}_i^c &= 0 && \text{(38g)}\\
\dot{q}_i^c &= 0 && \text{(38h)}\\
\dot{p}_v^w &= 0 && \text{(38i)}\\
\dot{q}_v^w &= 0 && \text{(38j)}\\
\dot{p}_i^g &= 0. && \text{(38k)}
\end{aligned}
$$

The 32-elements state vector X can be expressed in its *indirect* 29-element $\tilde{X}$ formulation as:

$$\tilde{X} = \left[\Delta p_w^{i\,T}, \Delta v_w^{i\,T}, \delta\theta_w^{i\,T}, \Delta b_w^T, \Delta b_a^T, \Delta\lambda, \Delta p_i^{c\,T}, \delta\theta_i^{c\,T}, \Delta p_v^{w\,T}, \delta\theta_v^{w\,T}, \Delta p_i^{g\,T}\right] \tag{39}$$

The equations governing the states have now to be formulated with respect to the new state vector, however, for the sake of simplicity we will not do it here. The focus of this section is to highlight the main concepts underlying the MSF framework and not its thorough mathematical description; the interested reader can find all the details and the final derivation of the error-state (or indirect) representation in Trawny and Roumeliotis, 2005. It has to be noticed also that in an actual use of the framework the user has to define and implement only the measurement equations (and their linearization) for the auxiliary sensors that are not already integrated in the framework because the state equations of the system are already implemented in the core part of the MSF.

4.3 CHARACTERIZATION OF THE SENSORS NOISE

4.3.1 IMU

In the classical Kalman Filter formulation we assume that our sensors cannot measure directly all the physical states of the system. Only certain output variables (linear combinations of the state variables) are measurable and the measurements are affected by *additive continuous-time WGN* $n(t)$ with expected value and covariance described (in the scalar case) by:

$$E[n(t)] = 0, \qquad \text{for all } t \tag{40}$$

$$\text{cov}[n(t+\tau); n(t)] = E[n(t+\tau) * n(t)] = R\delta(\tau) \tag{41}$$

The quantity R is sometime called *signal strength* or *signal intensity* in literature and it coincides with the Power Spectral Density (PSD) of the signal which is, in the case of the WGN, a constant quantity. The PSD is a distribution describing how the total power of the signal is distributed across the various frequencies of the signal. A *discrete implementation* of the White Gaussian process described by 40 and 41 can be obtained as follows (Crassidis, 2006):

$$n_d[k] = R_d w[k] \tag{42}$$

with

$$w[k] \sim \mathcal{N}(0,1) \tag{43}$$

$$R_d = R\frac{1}{\sqrt{\Delta t}} \tag{44}$$

or equivalently

$$R_d^2 = \frac{R^2}{\Delta t} = R_d^2 f_s \tag{45}$$

where $N(\mu, \sigma)$ is a normal distribution characterized by mean μ and standard deviation σ, Δt is the sampling time and f_s is the sampling frequency. It is noted here how the discrete formulation of the covariance is equivalent to the integral of the power spectral density of the (WGN) signal up to the frequency of interest. The same white gaussian modelization of the noise described here is typically assumed by the manufacturer of the inertial sensors that usually report the value of the PSD (Or the Noise Spectral Density (NSD), $\text{NSD} = \sqrt{\text{PSD}}$) on the datasheet.
Dimensional analysis reveals its dimensional units for the continuos time. As an example, for the gyroscope measurements:

$$\left[E[n_\omega(t+\tau) * n_\omega^T(t)]\right] = \left[R_\omega^2 \delta(\tau)\right] = \frac{\text{rad}^2}{\text{s}^2} \tag{46}$$

Knowing that $[\delta(\tau)] = \frac{1}{s}$, we obtain:

$$[R_\omega] = \frac{\text{rad}^2}{\text{s}} \qquad [R_\omega] = \frac{\text{rad/s}}{\sqrt{\text{Hz}}} \tag{47}$$

and similarly for the accelerometers:

$$[R_a] = \frac{\text{m}^2}{\text{s}^3} \tag{48}$$

In general the PSD has units SU^2/Hz (where SU stands for "signal units"). The term *power* in the PSD denomination comes from the fact that, in electrical circuits, the power can be written, in terms of the voltage V applied to the circuit, as V^2/Z, being Z the impedance of the circuit. The concept can be widened to the case of transducers returning a voltage output proportional, through a scaling factor k, to the extent of the input (measured) quantity, as it is shown in eq. 49.

$$\frac{(\text{m/s}^2)^2}{\text{Hz}} \overset{k}{\propto} \frac{V^2}{\text{Hz}} \overset{1/Z}{\propto} \frac{W}{\text{Hz}} \tag{49}$$

Table 6 on the next page shows an overview of the parameters normally used to characterize the noise from statistical and spectral point of views. The parameter used by the MSF for initializing the noise variables in the EKF is the *Continuos Time Noise Spectral Density*. The framework will take care of the discretization of the parameters.

Table 6: Noise characteristics and their units (SU: Signal Units).

Parameter	unit
Standard Deviation	SU
Variance	SU^2
Noise Spectral Density	$SU/\sqrt{Hz}$
Power Spectral Density	SU^2/Hz

IMU characterization tool

In order to automatically characterize the noise parameters of the sensors, and to double-check the NSD values provided by the datasheet a Matlab script has been specifically developed. The script allows to compute an approximation of the sensor's noise characteristics in the frequency domain and in the time one and it is freely downloadable[3] and editable in the terms of the GNU general public license v3[4]. The results of the toolbox are presented and discussed in the appendix D.

4.3.2 GPS

GPS receivers work by measuring range to four or more satellites. This measurements are normally affected by a number of different error sources, therefore they are usually referred as *psuedoranges*. Among the most common causes of error in the range estimation we find (Kaplan and Hegarty, 2005):

SATELLITE EPHEMERIS ERROR errors in the set of orbital information broadcast by the GPS satellites;

SATELLITE CLOCK ERROR Error in the compensation of the drift of the GPS satellites' atomic clocks;

IONOSPHERIC DELAY Error caused by the changes of the signal speed, introduced by the transit in the ionosphere

TROPOSPHERIC DELAY Error caused by the changes of the signal speed, introduced by the transit in the troposphere;

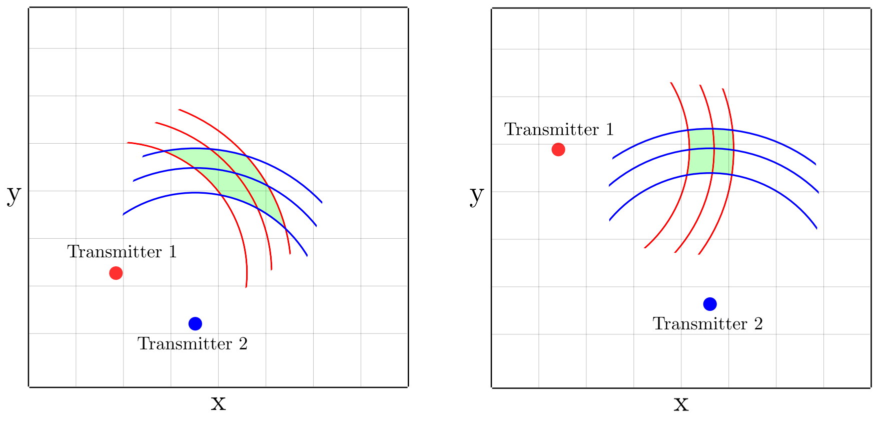
MULTIPATH The multipath error occurs when a signal reaches an antenna following two ore more paths. Often man-made structures or natural/-geographic features reflect the signal, causing the final signal received by the antenna to be a combination of the direct line-of-sight signal and some delayed reflected multipath replicas.

RECEIVER MEASUREMENT ERROR Error introduced by the internal replica of the expected signal, generated by the particular GPS receiver in use to measure the pseudorange.

These error components - treated as independent random variables - are root-sum-squared (RSS) to define a single parameter, called *User Equivalent Range Error*; the User Equivalent Range Error (UERE) conveniently express the standard deviation of the pseudorange measurements. The pseudorange error budgets for a Standard (Single Frequency) Positioning Service are shown in table 7.

Table 7: Main error sources affecting the GPS range estimates.

Error Source	σ error (m)
Satellite ephemeris error	0.8
Satellite clock error	1.1
L1 P(Y)-L1 C\A group delay	0.3
Ionospheric delay	7.0
Tropospheric delay	0.2
Multipath	0.2
Receiver measurement error (σ_{UERE})	0.1
Total (Root of Sum of Squares (RSS))	7.1



(a) Closest transmitters give higher uncertainty

(b) More spaced apart transmitters reduce the positioning uncertainty

Figure 38: Satellites-receiver geometry influences the precision of the position measurement.

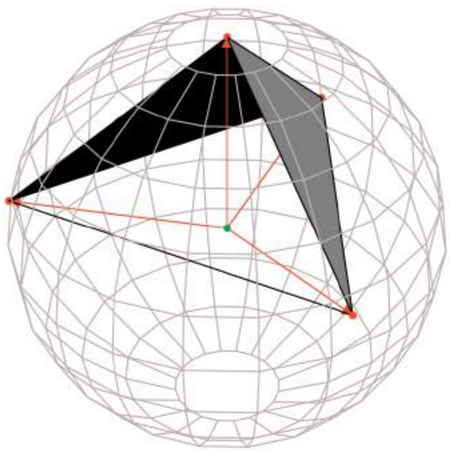


Figure 39: The volume of the tetrahedron (in the simple case of four observable satellites) formed by the tips of the receiver-satellite unit vector, is highly correlated with GDOP. Maximizing the volume tends to minimize the GDOP. Image source: Langley, 1999.

The UERE is one of the two parameters describing the precision and the accuracy of the GPS estimate of the position of the receiver. The second parameter is called Geometric Dilution of Precision (GDOP) and it is a unitless multiplier on σ_{UERE}, accounting for the effects of satellite geometry on the accuracy of position estimates (see Figure 38). Intuitively it is easy to depict the dependence of the positioning problem on the geometry of the satellites in the four-satellites example depicted in Figure 39 (four is the minimum number of satellites that have to be observed to correctly retrieve the position of the receiver antenna and the global time measure). In this case the position measured will be more accurate when the satellites are spread out in the sky, i.e. the accuracy increases with the volume of the tetrahedron connecting the four satellites and the receiver's antenna. The GDOP is a measure of the overall quality of the least-squares solution of the estimation problem that recovers position of the antenna and global time from pseudorange measurements. It is correlated with the volume of the tetrahedron in the four-satellites example; maximizing the volume tends to minimize GDOP.

The GDOP can be usually expressed as:

$$\sigma_G = \sqrt{\sigma_E^2 + \sigma_N^2 + \sigma_U^2 + \sigma_T^2} \tag{50}$$

where σ_E^2, σ_N^2, σ_U^2, σ_T^2 are respectively the variances on East, North, Up positions and Time; and σ represents the standard deviation of the pseudorange measurements, i.e. the UERE parameter previously introduced, plus the residual model error introduced in the least-squares problem formulation (Langley, 1999). It is common to refer to the specific *position*, *horizontal*,

3 https://github.com/blackistheanswer/IMU_analysis
4 http://www.gnu.org/copyleft/gpl.html

vertical and *time* components of GDOP to get better insight about the performance of the receiver.

$$\text{PDOP} = \frac{\sqrt{\sigma_E^2 + \sigma_N^2 + \sigma_U^2}}{\sigma} = \frac{\sigma_P}{\sigma} \tag{51a}$$

$$\text{HDOP} = \frac{\sqrt{\sigma_E^2 + \sigma_N^2}}{\sigma} = \frac{\sigma_H}{\sigma} \tag{51b}$$

$$\text{VDOP} = \frac{\sqrt{\sigma_U^2}}{\sigma} = \frac{\sigma_U}{\sigma} \tag{51c}$$

$$\text{TDOP} = \frac{\sqrt{\sigma_T^2}}{\sigma} = \frac{\sigma_T}{\sigma}. \tag{51d}$$

with σ_P and σ_H being the 3D position and 2D position standard deviations. Note also that the following relationships occurs between the various Dilution of Precision (DOP) values:

$$\text{GDOP}^2 = \text{PDOP}^2 + \text{TDOP}^2 \tag{52a}$$

$$\text{PDOP}^2 = \text{HDOP}^2 + \text{VDOP}^2. \tag{52b}$$

The vast majority of GPS receivers on the market provide at least the VDOP and the HDOP and update it in real-time to give the user a quality check on the accuracy of positioning information. This is also the case of the u-blox LEA-6H GPS receiver used in the sensor-fusion experiment. The Horizontal Dilution of Precision (HDOP) value in particular is directly forwarded to the Multi Sensor Fusion that uses it for the update of the covariance information of the position states; its minimum value is in the order of 2, 3 m.

4.3.3 Camera

The problem of calculating the covariance of a pose estimate obtained via visual-odometry is not trivial since the precision (and accuracy) of the pose measurement is not only dependent on the performances of the sensor used, but also on the particular VO algorithm. Weiss, 2012, pp. 73-74, cites some reference papers addressing this problem. Moreover, some VO implementations running on Robotic Operating System provide the real-time covariance of the pose estimates, allowing the user to avoid an explicit computation; unfortunately this is not the case of the Semi-direct Visual Odometry, the VO algorithm used in this work, for which a fixed value of the covariance has been considered.

4.4 STRUCTURE OF THE MSF CODE

The MSF comes with a certain number of sensors already integrated in the framework. The directory tree 1 lists the essential files in the MSF framework code, assuming the general case of a set of two different sensors, *x* and *y*, which have to be fused with the IMU readings. For the sake of clarity file names are italicized.

As already pointed out before, the core part of the algorithm is constituted mainly by the propagation equation for the IMU and by the actual EKF

```
ethzasl_msf
├── core
│   └── ...
└── updates
    ├── include
    │   └── msf_updates
    │       ├── x_sensor_handler
    │       │   ├── x_measurement.h
    │       │   └── x_sensorhandler.h
    │       │       └── implementation
    │       │           └── x_sensorhandler.hpp
    │       └── y_sensor_handler
    │           ├── y_measurement.h
    │           └── y_sensorhandler.h
    │               └── implementation
    │                   └── y_sensorhandler.hpp
    └── src
        └── x_y_module_msf
            ├── main.cpp
            ├── msf_statedef.hpp
            └── x_y_sensormanager.h
```

Directory Tree 1: MSF code tree.

code. We will not describe the implementation details for the core part of the framework since the user is not required to modify/customize this part in a standard use of the framework. On the contrary the user may have to edit the files in the updates section of the framework in order to customize it for his own sensors set-up. The structure of the directory tree of the framework highlights its modularity: every sub-folder in the *include* directory refers to a single sensor, so, in our example, we find two folders here: `x_sensor_handler` and `y_sensor_handler`. In the `src` directory instead the focus is now on the particular sensor-fusion problem hence the code here works at higher level, managing the state definition for the EKF, the variable and state initialization and some ROS-related routines. Table 8 offers an index of the framework files and the related functionalities.

4.5 PRELIMINARY RESULTS

In this section some preliminary results obtained using the MSF are presented. These results cannot be considered complete because of two main reasons. Firstly the SVO software at the moment does not provide a real-time estimate of the covariance on the pose measurements, this is a strong limitation when trying to fuse SVO with other pose sensor in a EKF framework, since we have already discussed how this kind of visual odometry measurements present a spatial drift that grows with the elapsed distance. In the tests presented here the covariance has been set to a constant value in the order of few mm^2, unfortunately this means we are ignoring the spatial drift of the SVO software (that can grow up to several meters). Secondly the absence of a ground truth reference makes not possible to compare the accuracy of the fused data w.r.t. the GPS raw measurements and/or the pure visual odometry. A possible way to measure the ground truth in this

Table 8: Description of the main files included in the MSF.

File	Description
`ethzasl_msf\include\msf_updates\x[y]_sensor_handler\`	
`x[y]_measurement.h`	provides methods for associating the sensor reading with a meaningful measure of the x[y]-type, computing the H matrix (linearized measurement model) for the measurement and applying the measure to the core part of the framework.
`x[y]_sensorhandler.h`	contains the `x[y]SensorHandler` class prototype; the methods declared here are implemented in `\implementation\x[y]_sensorhandler.hpp`.
`ethzasl_msf\include\msf_updates\x[y]_sensor_handler\implementation\`	
`x[y]_sensorhandler.hpp`	manages the ROS sensor-reading callback for the x[y] sensor
`ethzasl_msf\src\x_y_module_msf\`	
`main.cpp`	main entry point for the ROS node, constructs a `XYSensorManager` object.
`msf_statedef.hpp`	State definition of the EKF as defined for a given set of sensors.
`x_y_sensormanager.h`	cointains the `XYSensorManager` constructor, the states initialization routines and the ROS dynamic reconfiguration callback.

kind of experiments is to use a *tracking system* like the Leica Total Station[5]: a device capable of long-range continuous laser tracking of a prism attached on the body to be monitored. Keeping clear in mind these two limitations, the experiments still highlight some of the advantages implicit in the use of sensor fusion in complex perception problems.

The result of a first experiment are shown in figure 40 on the following page; it shows, qualitatively, the effect of multipath on raw GPS measurements. When the GPS receiver is close to buildings or other artificial or natural structures, the signal may be reflected before reaching the antenna, affecting the final position measurement (see 4.3.2). This effect is clear comparing the yellow trace (GPS raw measurements) with the red one (the actual path traveled); however the position estimate resulting from the fusion of the GPS and the camera is not affected by the issue (figure 40b). This experiments ends with the UAV trying to enter the building, this causes immediate loss of GPS signal, but also the vision part is affected by the transition, because of the sudden variation in the brightness of the scene. The implementation of an algorithm for adaptive exposure correction of the camera, and intelligent sensors switch-on/off routines, would allow not to lose the visual tracking of the environment and to perform autonomous outdoor/indoor (and viceversa) transitions almost seamlessly (the interested reader is referred to Shen, 2014). The simultaneous loss of information from the two sensors makes the IMU the only on-board available sensor and its direct integration causes an almost immediate drift (see 40d at second 115)

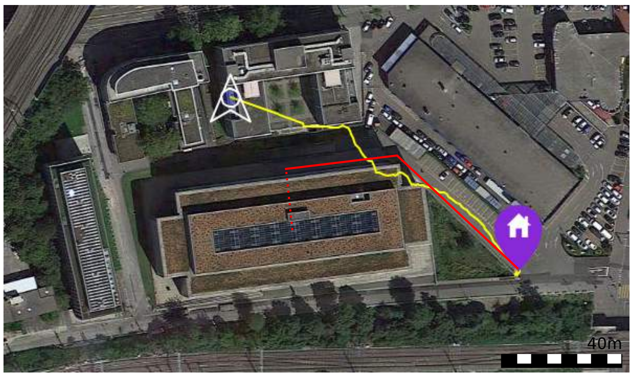
In the second experiment (figure 41 on page 59) the UAV follows a close trajectory, therefore in this case it is possible to draw some information about the extent of the spatial drift of the visual odometry. In this experiment the drift is about 6% of the total distance travelled (50 m). Similarly to the first experiment also in this test there are some external elements (the trees) disturbing the GPS signal; and exactly as in the previous case also here the fused output is immune to this problems, resulting in the much cleaner and smooth trace in figure 41b.

The hardware setup of the described test includes a PX4FMU auto-pilot (see 1.2) and its IMU (200Hz), a Matrix Vision BlueFox camera (global shutter, 752×480 pixel resolution, 60 fps), and a 3DRobotics[6] GPS module (5 Hz).

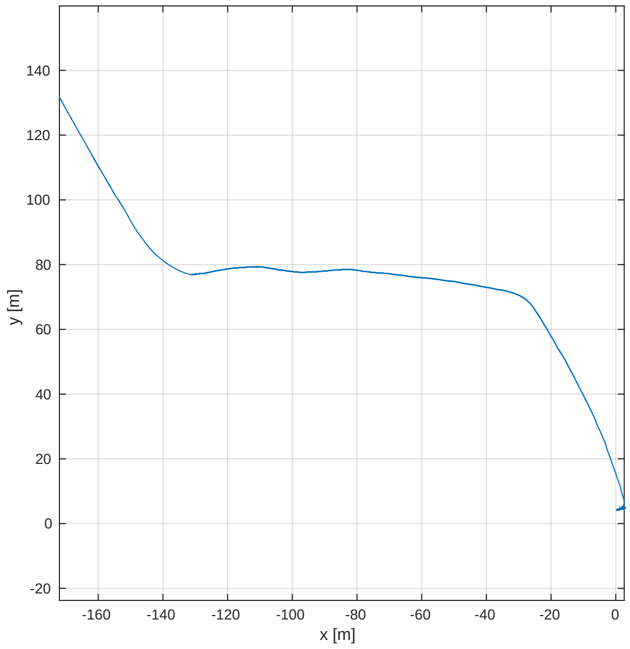
4.6 CONCLUSION

Where is the robot in this world? What is around it? How can it safely interact with its environment and how can it solve "new" emerging problems? These are four questions the research in robots perception tries to answer. However we have already mentioned in the Introduction how the majority of UAVs in the market today are "autonomous" in the sense they are able to follow pre-programmed sequences of latitude/longitude/altitude points, without any further information about their surroundings. We have shown in this chapter how a possible implementation of a sensor fusion architecture on a capable UAV can augment the available information of the state of the system, providing a result that is *better* then the measurements of the sensors taken singularly. In particular we choose to integrate on-board the Multi Sensor Fusion, an EKF-based framework developed by

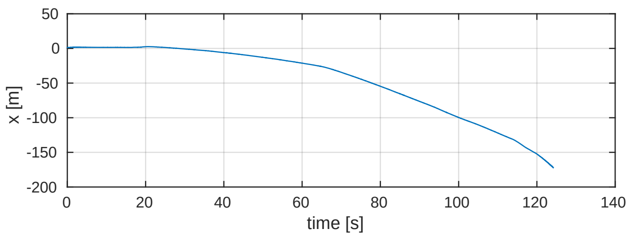
5 http://www.leica-geosystems.com/en/Total-Stations-TPS_4207.htm
6 http://3drobotics.com/



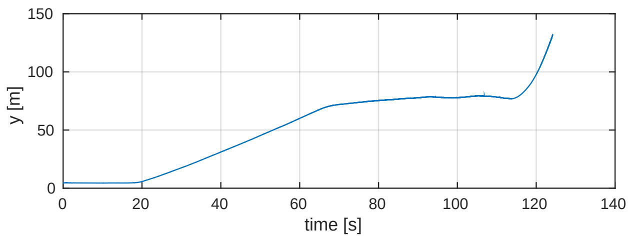
(a) GPS raw measurements



(b) Position estimate output from sensor fusion

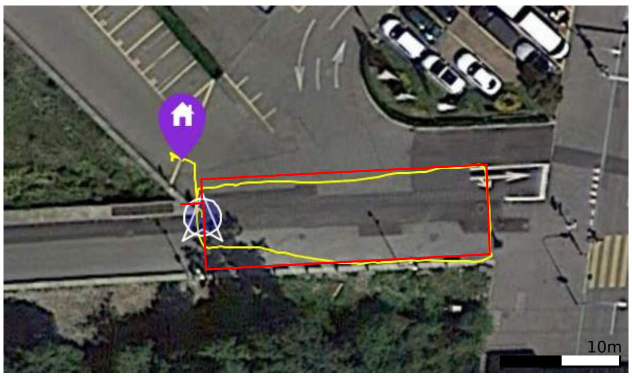


(c) X Position estimate

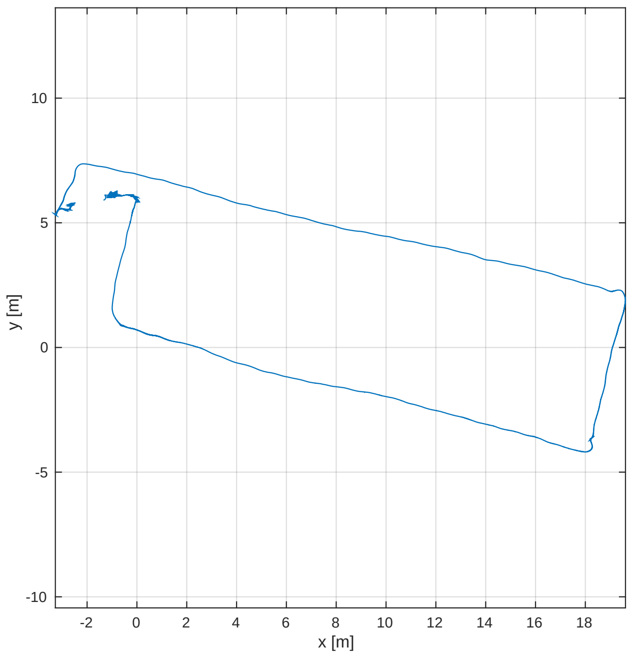


(d) Y Position estimate

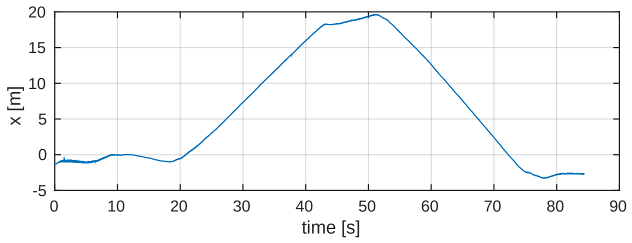
Figure 40: MSF Test 1.



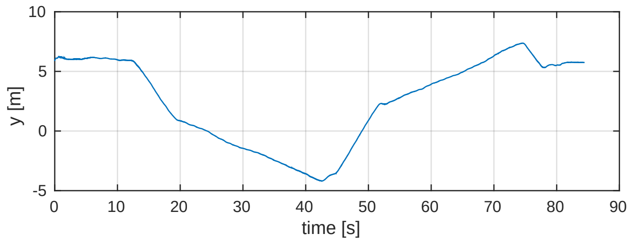
(a) GPS raw measurements



(b) Position estimate output from sensor fusion



(c) X Position estimate



(d) Y Position estimate

Figure 41: MSF Test 2.

Autonomous Systems Lab at ETHZ, to fuse the information coming from a GPS receiver, the IMU and a monocular camera. In addition to offering a series of advantages with respect to other, simpler, EKF-based softwares (such as intrinsically managing: delayed measurements, acquisitions at different frame-rates, absolute and relative measurements, on-line estimation of the calibration states of the sensors), one of the most appealing characteristics of the MSF is its modularity, i.e. the possibility to assume new sensors as black-boxes, simplifying the integration of new devices in the framework. Once introduced the system and measurement models for the sensor payload in use, we focused on the noise characterization of these sensors, since a correct model of the noise contributes is critical when using an EKF. Finally we have shown and discussed the preliminary results of two tests performed with the IMU-GPS-camera fusion running in real-time. Although these results are still preliminary and cannot be used for detailed analysis of the performance of the system, they set an important milestone for the follow-up of the activities and projects described in this dissertation, as it will be further discussed in the next chapter.

5 DISCUSSION AND CONCLUSION

In this dissertation we have presented our contributions in the field of Unmanned Aerial Systems, trying to highlight the advantages of research in *control*, *planning* and *perception* in order to achieve *actual* autonomy in UAVs missions. The importance of such an integration of these three different aspects is demonstrated by recent development in flying autonomous systems market (see figure 44 on page 63), with new products that are gradually integrating new control strategies, improved mission planner and new devices used for better in-flight perception (and not just mission-related sensor payload).

5.1 SUMMARY OF THIS DISSERATATION

In chapter 2 we have presented the design and implementation of a test facility for the benchmark of visual-based routines for aerospace operations. The focus here is on the model and the control of an autonomous quadrotor, we have approached the problem defining a simple PID control architecture completely capable of stabilizing the UAV and satisfying the system requirements.

In chapter 3 we shifted the focus on planning, proposing an original cloud-based architecture for mission management of one or more autonomous agents. The platform autonomously accesses open-data from the web and builds missions compatible with a set of given constraints. The platform has been publicly presented in June 2014 in Turin during a demonstration authorized by the Italian Civil Aviation Authority (ENAC).

In chapter 4 we presented an implementation of a sensor fusion architecture on a capable UAV. We have presented the system (based on the Multi Sensor Fusion, developed at ETHZ) and discussed in details the characterization of the sensors noise and the results of two tests performed with the IMU-GPS-camera fusion running in real-time.

5.2 RESEARCH OUTLOOK

The result obtained and documented in this dissertation offer some interesting starting point for future research activities and - in particular - for the follow-up of the already described STEPS and F4SC projects. In fact the follow-up of the STEPS project requires to build and integrate an outdoor, 1 : 1 scale, benchmarking facility. The final goals of the new project are the same as in STEPS (see 2), but working outdoor with precision and accuracy comparable to the indoor ones is challenging since it is not possible to rely on a standard GPS, because its error range would be too wide. A multi sensor fusion approach could be in this case a smart way to improve the performance of the system without the need to buy expansive device for external outdoor tracking, such as total stations, differential or Real Time Kinematics (RTK) GPS receivers. From a control point of view, moving out-

Figure 42: A possible area for the new STEPS facility in Piedmont, Italy

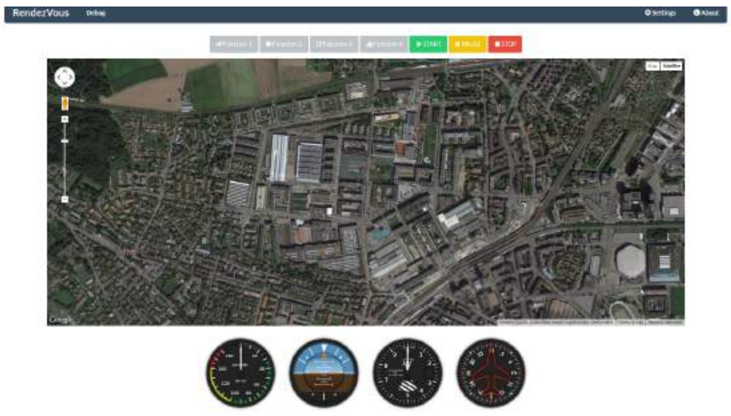


Figure 43: An improved GUI is already available and will be used for the future F4SC activities

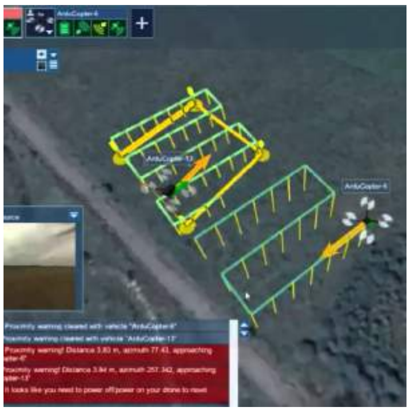
door in a 1 : 1 facility would mean adding a lot of new disturbances sources, mainly due to atmospheric agents, but also larger distances to travel; the simple point-to-point PID-based control architecture is very likely to fail in these conditions and new, improved trajectory controls should be studied in this case. A better perception represents the key-factor for the future development of the F4SC activities as well. In-flight faults due to poor GPS signal or multi-path disturbances are simply not admissible when flying in a city; and we have already shown in section 4.5 how a monocular camera helps in solving this issue, without loosing the global positioning data of the vehicle. Vision-based strategies for obstacle avoidance are also being studied. At the same time an improved open-source Graphical User Interface (figure 43) is already available on-line[1] and will let the remote user have a better interaction with the cloud platform.

1 https://github.com/fly4smartcity/rendezvous

(a) *Fotokite*
fotokite.com

(b) *Sensefly Exom*
sensefly.com

(c) *DJI Inspire*
dji.com



```
Proximity warning cleared with vehicle "ArduCopter-6"
Proximity warning cleared with vehicle "ArduCopter-13"
Proximity warning! Distance 3.83 m, azimuth 77.43, approaching
opter-6"
Proximity warning! Distance 3.84 m, azimuth 257.342, approaching
opter-13"
It looks like you need to power off/power on your drone to reset
```

(d) *U|G|CS*
ugcs.com

Figure 44: Civil UAV manufacturers are progressively investing more effort in the development of advanced control, sensors and mission planning solutions.
(a) Fotokite's solution for aerial photography exploits the dynamics properties of a tethered quadrotor to make the control of the vehicle more intuitive; (b) Sensefly Exom UAV features a gimballed sensor head and a total of five combined camera/sonar sensors for environmental awareness ; (c) the recently released DJI Inspire uses a dedicated camera for imporved stability in indoor navigation; (d) U|G|CS is a ground station for advanced planning and execution of multi-vehicles missions. (Pictures courtesy of fotokite.com, notebookitalia.it, cnet.com, ugcs.com)

A VICON PRECISION CHARACTERIZATION

As stated in chapter 2 on page 7, to satisfy STEPS project requirements the quadrotor position and attitude have to be known with high precision when the photo is triggered. Vicon Bonita camera datasheet declares a positional accuracy down to 1 mm when a 4 m $\times$ 4 m area intersects the angle of view of the camera. In order to verify both the accuracy and precision of the system two tests have been performed. In the first one, a reference object of known geometry has been used as the tracked object; in the second test, the quadrotor has been directly tracked with all its motors on, in order to check the influence of the vibrations on the system precision.

A.1 MOTORS OFF

In this test the *wand object* has been tracked in order to get the system precision for a steady target. The wand is a tool used to calibrate the Vicon system cameras and to set the origin of the global reference frame as shown in figure 45 on the following page, for this reason its geometry is known by the Vicon system.

After the calibration the wand is left in the center of the workspace at the same distance from each camera. Vicon system recognizes the wand object, hence sets there the global reference frame and starts tracking it. The wand local reference frame has been set in the center of one marker with X,Y and Z axes parallel to the global ones, therefore the expected measuring values are listed in table 9.

The results listed in table 9 confirms the Vicon performances declared in the datasheet, the system precision is lower than 1 mm while the position accuracy is in the order of 1 mm.

The figure 46 on the next page shows the three-dimensional scatter graph of the captured position data. It is possible to notice a main cloud where the great part of measures falls, and a separated cloud with few points. This cloud is due to several Z position glitches as shown in figure 47 on page 67. The main cloud extends itself mainly along the Y position axis, it means that the precision measure is affected by a larger error with respect

Table 9: Expected values, accuracy and precision of Vicon system.

	Expected values	**Accuracy**	**Precision**
X	7 mm	1.174 mm	$\pm$0.104 mm
Y	7 mm	0.844 mm	$\pm$0.143 mm
Z	53 mm	1.115 mm	$\pm$0.105 mm
X rotation	0 rad	$1.28 \cdot 10^{-3}$ rad	$\pm 0.83 \cdot 10^{-3}$ rad
Y rotation	0 rad	$1.15 \cdot 10^{-3}$ rad	$\pm 1.08 \cdot 10^{-3}$ rad
Z rotation	0 rad	$4.74 \cdot 10^{-3}$ rad	$\pm 0.90 \cdot 10^{-3}$ rad

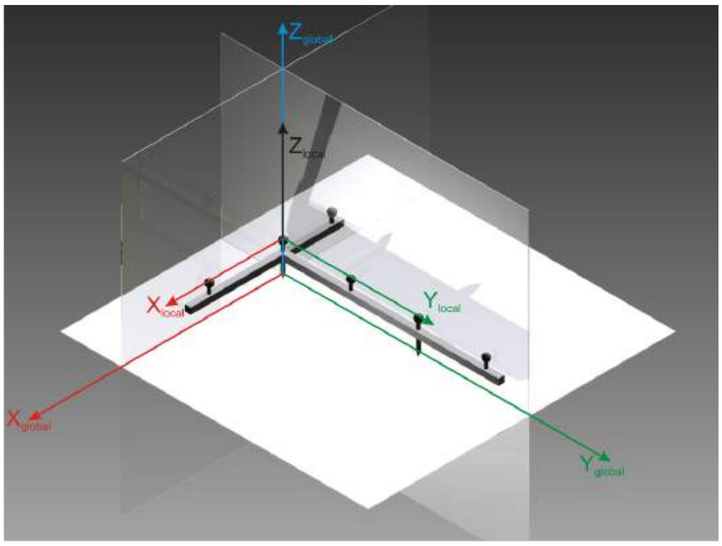


Figure 45: Isometric view of the wand.

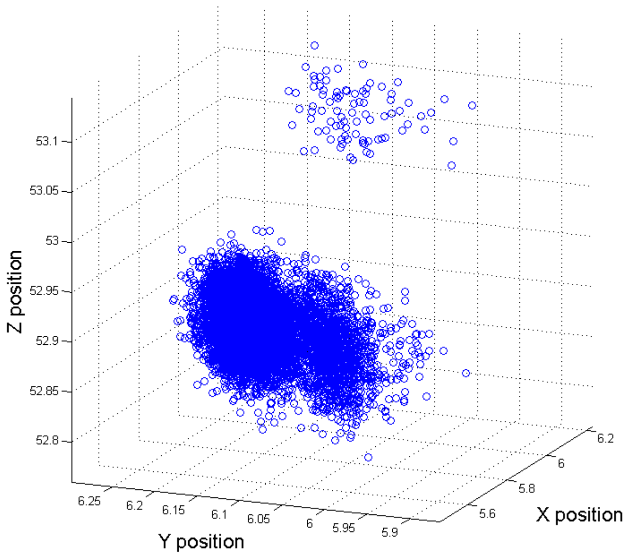
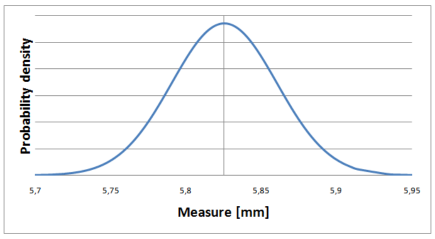
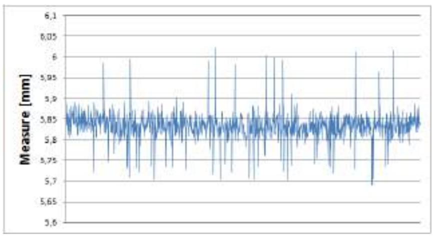


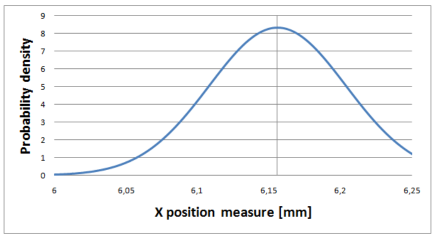
Figure 46: Three-dimensional position scatter plot [mm]



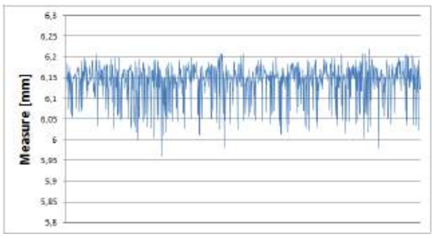
(a) *X position.*



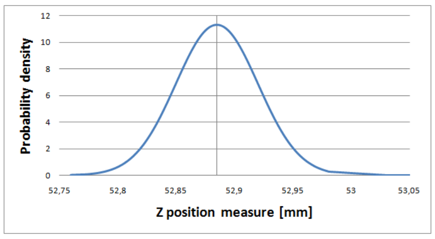
(b) *X position noise.*



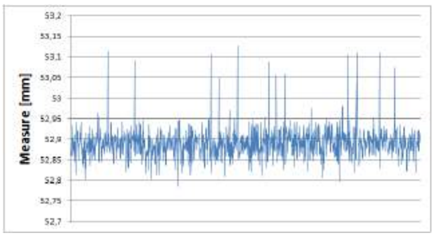
(c) *Y position.*



(d) *Y position noise.*



(e) *Z position.*



(f) *Z position noise.*

Figure 47: Gaussian distribution and noise trend for the position measurements.

to others axis, indeed it presents the higher standard deviation value and the normal distribution graph is the most dilated. The Y position measure presents several peaks and it is affected by more noise than other measures.

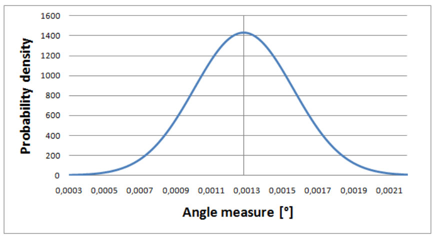
In figure 48 on the next page are showed the Gaussian distributions and the noise captures of angular measures around the axis of the global reference frame.

A.2 MOTOR ON

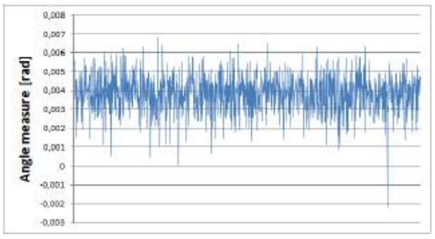
The aim of this test is to verify how the vibrations due to motors rotation influences the system precision. The quadrotor is positioned on the ground and the motors are turned on at the minimum gas value. The vibrations on the UAV induces markers displacement; as a result a higher standard deviation value with respect to the steady test is expected (see table 10 on the following page).

The results of this experiment are shown in figure 50 on page 69. In figure 49 on the following page the difference between the points clouds in both cases is shown.

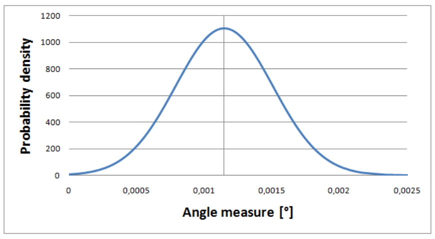
The tests confirm the good damping behaviour of the electronic platform supports: the precision value, also with motors on, stays under 1 mm.



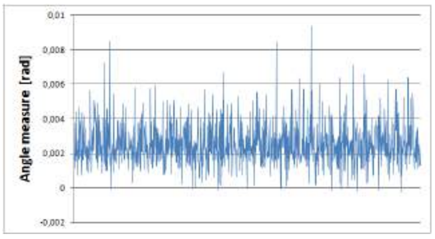
(a) *Rotation around the X axis.*



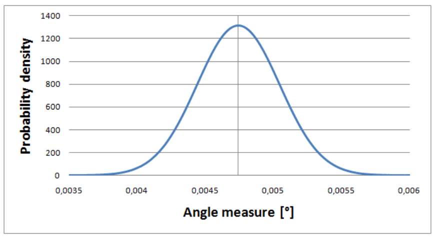
(b) *Rotation around the X axis, noise.*



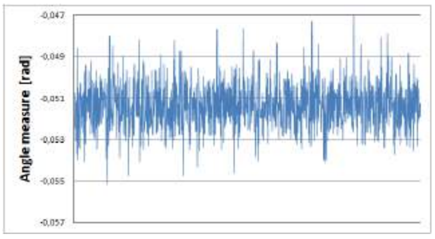
(c) *Rotation around the Y axis.*



(d) *Rotation around the Y axis, noise.*



(e) *Rotation around the Z axis.*

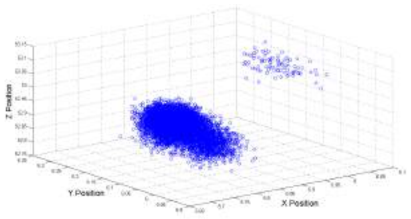


(f) *Rotation around the Z axis, noise.*

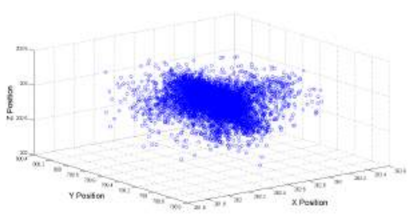
Figure 48: Gaussian distribution and noise trend for the attitude measurements.

Table 10: Accuracy and precision differences, static and dynamic case.

	Turn off	**Turn on**	**Increment**
X	±0.104 mm	±0.504 mm	±0.400 mm
Y	±0.143 mm	±0.450 mm	±0.307 mm
Z	±0.105 mm	±0.444 mm	±0.339 mm
X rotation	$\pm 0.83 \cdot 10^{-3}$ rad	$\pm 3.18 \cdot 10^{-3}$ rad	$\pm 2.35 \cdot 10^{-3}$ rad
Y rotation	$\pm 1.08 \cdot 10^{-3}$ rad	$\pm 3.90 \cdot 10^{-3}$ rad	$\pm 2.82 \cdot 10^{-3}$ rad
Z rotation	$\pm 0.90 \cdot 10^{-3}$ rad	$\pm 3.60 \cdot 10^{-3}$ rad	$\pm 2.70 \cdot 10^{-3}$ rad

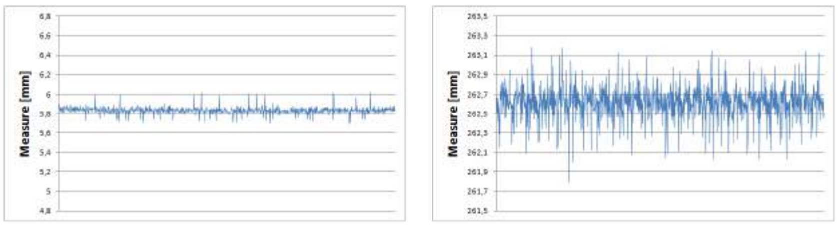


(a) *Motors off.*



(b) *Motors on.*

Figure 49: Motors off vs motors on position scatter.



(a) *Motors off.*   **(b)** *Motors on.*

Figure 50: X position measure with motors off and motors on.

B VIBRATIONAL TESTS

During the gas dynamics characterization experiments (Gas Dynamics), a specific vibrational behaviour of the quadrotor has been observed: in the hovering condition the cross beams of the quadrotor presented a considerable vibrational motion on the propellers plane. Amplitude of this motion varied in a range of about ± 5 mm. In order to better investigate the origin of this problem some tests have been performed. These tests use the modal analyser *LMS SCADAS III*[1] to acquire the signals from piezoelectric accelerometers (glued with a special wax on the motor bracket) and from a diode excited by a laser beam, periodically interrupted by the spinning propeller. Once acquired, the software allows the signals to be aggregated and post-processed in a number of different ways (Fast Fourier Transform, Frequency Domain Response, Time Domain Response).

B.1 WATERFALL PLOTS

The first set of experiments tries to reproduce a flight alike vibrational condition. Motors are controlled with a rising ramp in a range of velocities close to the hovering one, with a laser/diode couple measuring the propellers rotational speed. The results of the analysis are shown in a *waterfall plot*, presenting *vibrational frequencies* against *angular velocity* against *vibration amplitude* on the three axes. The SCADAS system triggers new captures at 25 rpm steps and at each capture the software calculates the *Fast Fourier Transform FFT* of the horizontal (H1) and vertical (V1) vibration on motor 1 (see figure 52 on the next page) and the horizontal (H3) vibration on motor 3 which is orthogonal to the first (see 51). The system has been tested both with all the four motors active and with just one.

The testbed is composed by:

- Quadrotor;
- LMS SCADAS III system;
- 3 piezoelectric accelerometers;
- Laser sensor;

B.1.1 All motors on

Each waterfall of figure 53 on page 73, shows the first three rotational orders. On waterfall related to H1 (figure 53a), a peak resonance appears on the first order when the motor rotates at 3800 rpm, this can be related with the response to propeller unbalance. In the same picture a clear quadratic trend appears on the second order, this effect is probably due to the aerodynamic effects caused by the propeller rotation on the laser bracket. The same trend can be found looking at the second order of the V1 waterfall on

1 http://tinyurl.com/kcojkwn

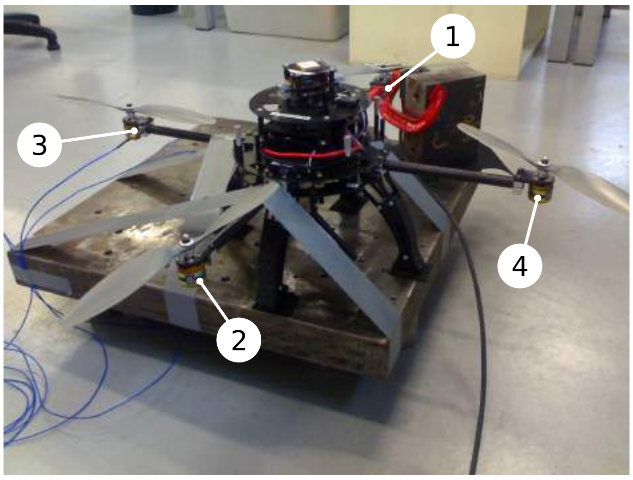


Figure 51: The quadrotor bounded on the testing ground.

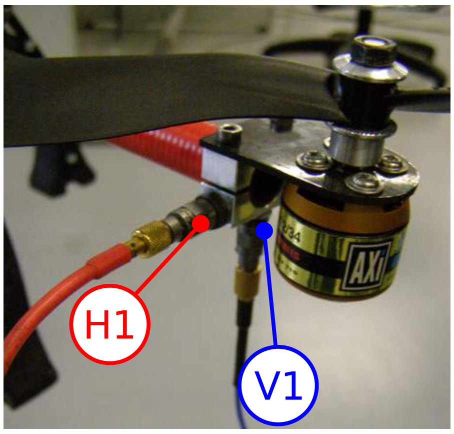
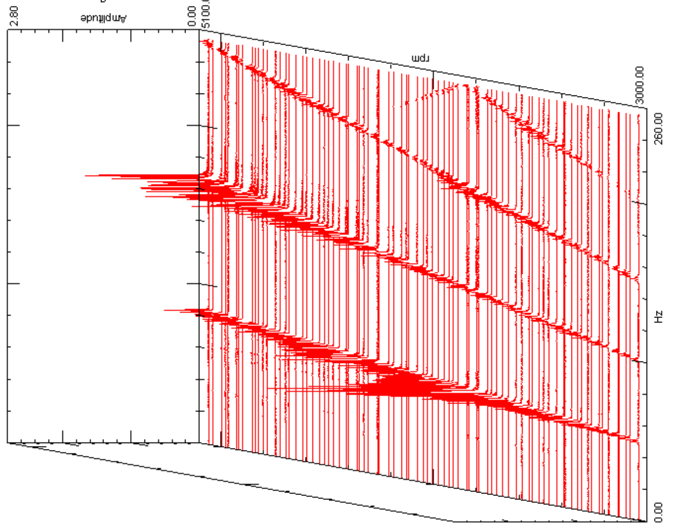


Figure 52: Accelerometers positioning on motor 1.

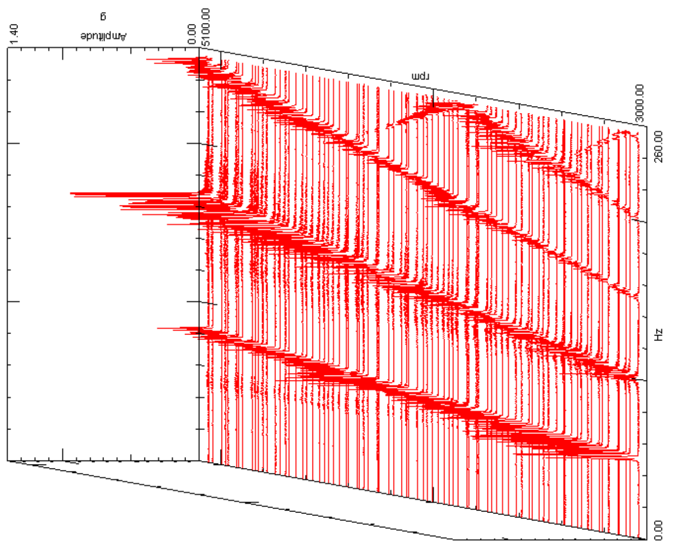
the amplitude-rpm plane (see fig. 53b). The hypothesis is confirmed by the fact that the amplitude is much lower on motor 3, that is not obstructed by the laser sensor support. The first order of H3 has a less regular vibrational response with two clear peaks at different angular velocities (figure 53c). It is possible to suppose that they correspond to two critical speed excited by the unbalance, probably also due to the interferences coming from the other motors. In order to better investigate on the frequency response of arm and to remove the coupled effects of the other motors, an experiment with only motor one turned on has been performed.

B.1.2 Motor 1 on

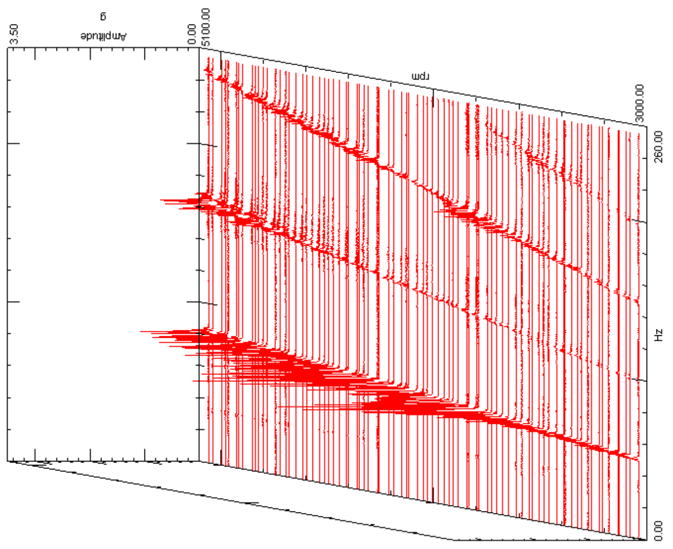
If the motors rotate at the *critical speed*, the natural frequency of the body is excited generating a resonance, at higher velocity the vibration amplitude remains constant unless others *critical speed*. The analysis of the system with a single motor on, and in particular the results shown in figure 54a, confirms the presence of a resonance peak at 3800 rpm. This peak is likely due to the rotation unbalance on the first *critical speed*. Looking then to the response on H3 (figure 54c) this behaviour is confirmed even if motor 3 is turned off; this effect is of particular interest since it proves that the vibrations



(a) $H1$.

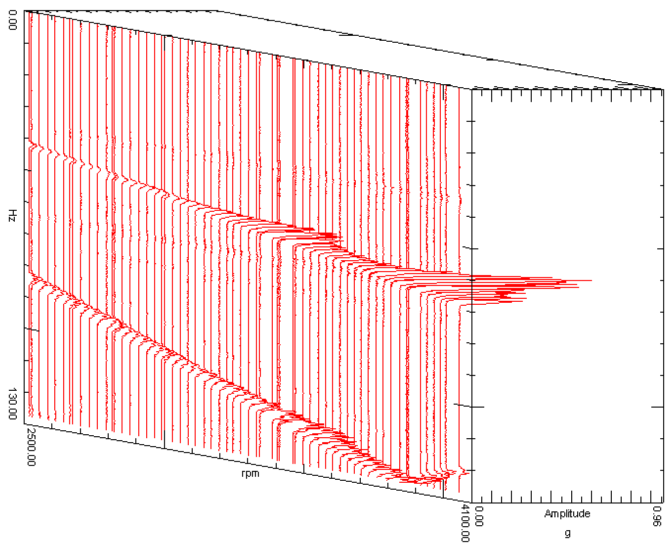


(b) $V1$.

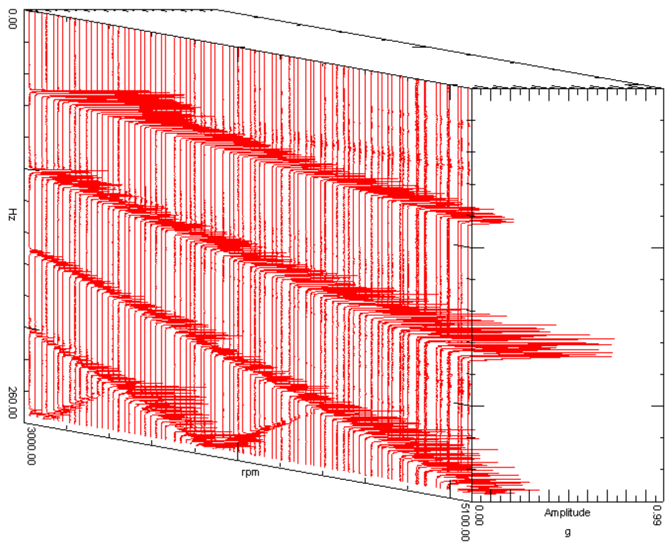


(c) $H3$.

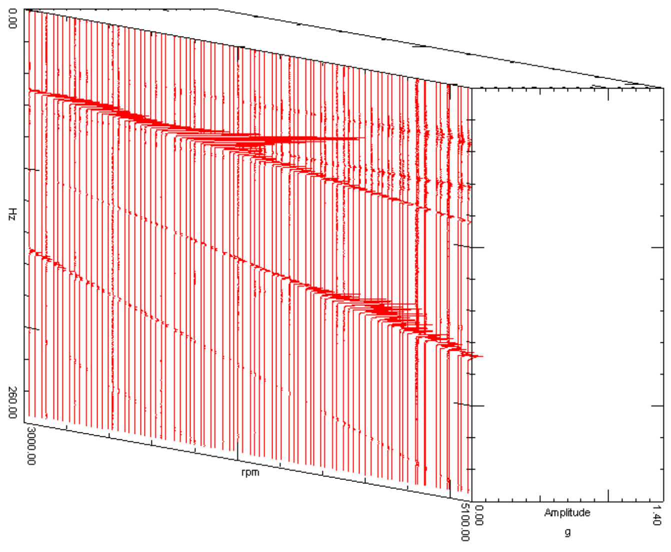
Figure 53: Whole motors ON, waterfall plots.



(a) $H1$.



(b) $V1$.



(c) $H3$.

Figure 54: Motor 1 ON, waterfall plots.

generated by the unbalance on motor 1 are transmitted through the vehicle framework and excite the natural frequency of each beam. This proves that the H3 response observed with all the motors turned on, is also affected by the unbalances of the other motors. The quadratic trend already observed before is still evident here in the second order V1 response (figure 54b), but its amplitude is slightly lower, probably because there are not other transmitted excitations from the other three motors.

B.2 BEAM MODAL ANALYSIS

In order to define the natural frequency of the arm, an impact test has been done. The test is performed only on one arm (i.e. the whole beam + motor + propeller system) of the quadrotor. The arm is tied on a seismic mass and two accelerometers are attached on the motor frame as illustrated in figure 52 on page 72. The arm is then hit by a special hammer connected to the SCADAS system. The software records the hammer input signal magnitude and the responses of the accelerometers. Because the accelerometers are positioned in both horizontal and vertical directions, given an impact direction, it is possible to measure the frequency response of the co-located sensor or of the orthogonal one.

The impact test results (see figure 55 on the following page), for both horizontal and vertical hammering, show a natural frequency at 63 Hz. When the motor rotates at 3800 rpm, corresponding to 63 Hz, the unbalance excites the beam natural frequency causing a resonance hence an high vibration on the XY plane.

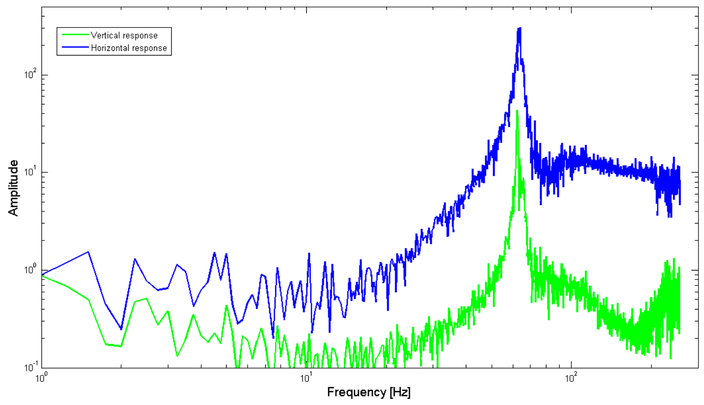
B.3 RESULTS

We start from the data on the weight of the quadrotor (1.56 kg corresponding approximately to 15.3 N) and use the combination of the thrust and rpm characteristics (see section Gas Dynamics) to compute the hovering rpm value for the vehicle, which result in about 3800 rpm, corresponding to 63 Hz.

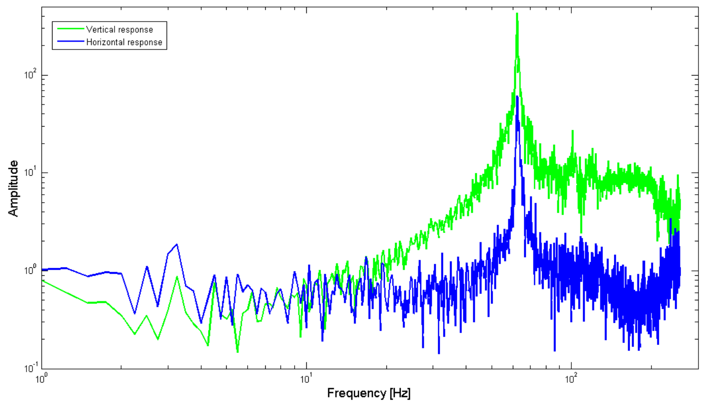
$$\mathrm{GAS}^{H}_{cmd} = \frac{T^{H}}{0.1489} = 15.3/0.1489 \approx 103 \tag{53}$$

$$\omega^{H}_{rpm} = -0.06589 \cdot {\mathrm{GAS}^{H}_{cmd}}^{2} + 36.67 \cdot \mathrm{GAS}^{H}_{cmd} + 751.2 \approx 3800 \text{ rpm} \approx 63 \text{ Hz} \tag{54}$$

It has been then confirmed that the vibration rises because of the motors unbalance, in fact unfortunately the quadrotor hovering angular velocity corresponds to a *critical speed* of the system and flying in the hovering condition excites the beams causing large vibrations. The vibrational behaviour stresses the whole system and can cause fatigue problem. Probably the stress in carbon tube is negligible but this seems not to be true for propellers, since several catastrophic failures has occurred during the tests. The results suggest that a possible solution to this problem would be to increase the beam stiffness using unidirectional carbon fiber beams instead of the multi-directional ones used until now. Further impact tests on the beam demonstrate that, in the unidirectional case, the natural frequency of the

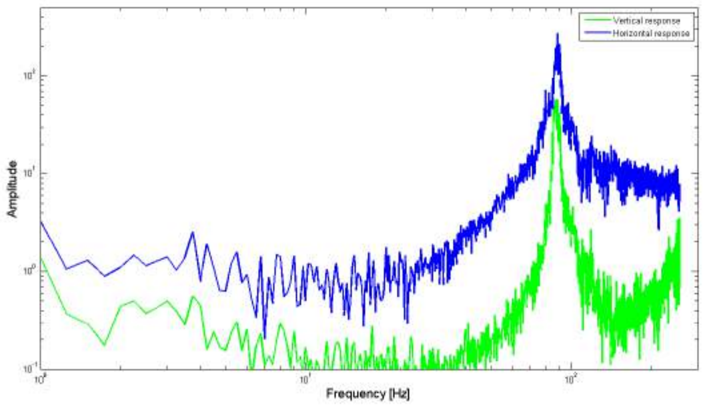


(a) *Frequency response to a horizontal hammer tap.*

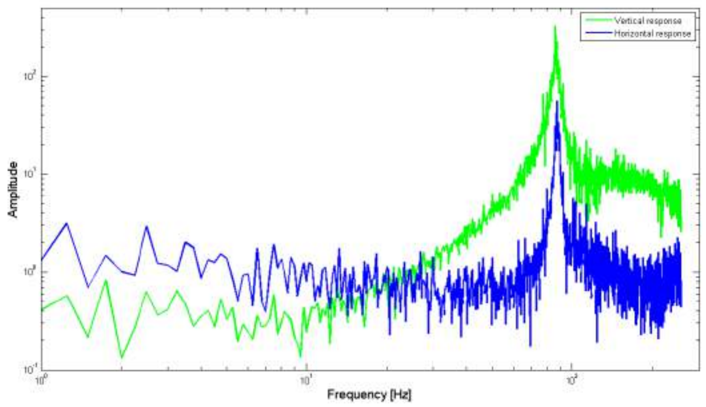


(b) *Frequency response to a vertical hammer tap.*

Figure 55: Impact test frequency responses on motor 1 using multidirectional carbon fiber beam.



(a) *Frequency response to a horizontal hammer tap.*



(b) *Frequency response to a vertical hammer tap.*

Figure 56: Impact test frequency responses on motor 1 using unidirectional carbon fiber beam.

arm shifts to higher frequencies, with a both horizontal and vertical natural frequency at 90 Hz, as shown in figure 56.

According to equation 53 on page 75 and 54 on page 75, a natural frequency of 90 Hz moves the *critical speed* from 3800 rpm to 5400 rpm, out of the hovering range. During the flight the propellers angular velocity will always be lower than 5400 rpm, this ensures a less critical behaviour.

C FLY4SMARTCITY PLATFORM MESSAGES

C.1 MISSION PLANNER MESSAGES SET

Listing 1: **Action.msg**

```
string name
int8 DEVICE = 0
int8 FEEDBACK = 1
int8 CHECK = 2
int8 UNICAST_NOTIFY = 3
int8 MULTICAST_NOTIFY = 4
int8 BROADCAST_NOTIFY = 5
# enumerate above
int8 type
# filled only in case of DeviceAction and Feedback
string action_name
# filled only in case of DeviceAction and Feedback
Parameter[] parameters
# filled only in case of Notify (any kind) or CheckNotification
string slot_name
# filled only in case of MulticastNotify or UnicastNotify
string[] receivers_name
```

Listing 2: **Bool.msg**

```
bool data
```

Listing 3: **Coordinate.msg**

```
float64 latitude
float64 longitude
float64 altitude
float64 heading
```

Listing 4: **CoordinateArray.msg**

```
Coordinate[] waypoint
```

Listing 5: **Drone.msg**

```
Header header
string name
string type_name
Coordinate home
Move[] movements
string[] slot_names
uint8 SAFE=0
uint8 NORMAL=1
uint8 AGGRESSIVE=2
uint8 travel_mode
```

Listing 6: ListString.msg

```
std_msgs/String[] list
```

Listing 7: Move.msg

```
string name
uint8 START=0
uint8 STOP=1
uint8 TAKE_OFF=2
uint8 LAND=3
uint8 GO_TO=4
uint8 HOVER=5
uint8 CIRCLE=6
uint8 HEAD_TO=7
#enumerated above
uint8 type
Action[] pre_actions
Action[] post_actions
#filled only in case of TakeOff and Circle moves
float64 altitude
#filled only in case of Goto or Circle move
Coordinate target_position
uint8 DIRECT=0
uint8 HORIZONTAL_FIRST=1
uint8 VERTICAL_FIRST=2
#enumerated above, filled only in case of GoTo move
uint8 strategy
#filled only in case of Hover or Circle moves
duration duration
#following parameters are filled only in case Circle move
float64 radius
bool clockwise
#filled only in case of HeadTo move
float64 direction
```

Listing 8: Parameter.msg

```
string key
string value
```

Listing 9: SensorPacket.msg

```
# GPS_current_position
float64 c_longit
float64 c_latit
int32 c_altit
# GPS_home_position
float64 h_longit
float64 h_latit
int32 h_altit
# GPS_target_position
float64 t_longit
float64 t_latit
int32 t_altit
# Flight_status
uint8 current_wayp
uint8 tot_wayp
int16 altimeter
```

```
uint16 fly_time
uint8 grd_speed
int16 top_speed
# Attitude
int16 compass_heading
int8 roll_ang
int8 pitch_ang
uint8 battery
```

C.2 OPEN DATA MESSAGES SET

Listing 10: Coordinate.msg

```
float64 x
float64 y
float64 z
```

Listing 11: Data.msg

```
Status status
Open_data[] data
```

Listing 12: OpenData.msg

```
int8 TYPE_STATIC=0
uint8 TYPE_DYNAMIC=1
uint8 type
string label
Parameter[] attributes
geometry_msgs/Polygon area
```

Listing 13: Parameter.msg

```
string key
string value
```

Listing 14: Polygon.msg

```
Coordinate[] vertex
```

Listing 15: Status.msg

```
int8 status_code
string reason
```

evi

D IMU CHARACTERIZATION

This appendix briefly summarizes the results obtained after the characterization of the PX4FMU IMU's sensors. The standard sensors-conditioning chain implemented in the PX4FMU is here simplified in a sampler stage followed by a first hardware low-pass filter (HW LPF), embedded in the IMU IC circuit, and a software one (SW LPF), implemented in the PH firmware. The chain is illustrated in figure 57. The experiments are performed given two different configurations of the sampling frequency F_s and the cut-off frequencies of the two hardware and software low-pass filtering stages, ${f_C}^{HW}$ and ${f_C}^{SW}$.

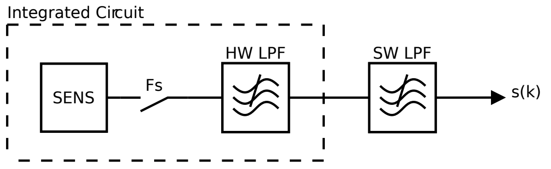


Figure 57: Conditioning chain for the PX4FMU's inertial sensors.

Figures 58, 59, 60 and 61 show the results (samples distribution, time history and noise spectral density) of the characterization experiments performed for each of the 15 seconds-long $s(k)_i$ acquisitions, using two distinct parameters sets, both for the accelerometers and the gyroscopes. While the first parameter set (set 0) is the one currently in use in the autopilot, the second one is useful to understand how different settings influences the experiments' output. In particular, in the second case the software low-pass filter has been turned off, with the cut-off frequency of the hardware one almost matching the Nyquist frequency of the signal (~100 Hz). This results in an high quantization of the gyroscopes readings (61a and 61b) and in a (predictably) flat noise-density spectrum (60c and 61c). The maximum error on the value of the noise spectral density computed by the toolbox is 52% and it may look very high; however it has to be noted that only a single value for the noise spectral densities of the three accelerometers (and similarly for the gyroscopes) is given by the datasheet. Since usually a conservative approach is used for describing this parameters, the datasheet value is likely to be the worst-case (the most "noisy") among the three values of the noise spectral densities, computed for all the axes of the two sensors. A comparison of the worst experimentally found NSD value with the datasheet one would reveal different error ranges: 17% (z-axis accelerometer NSD) and just 1% (x-axis gyroscope NSD).

The results of the experiments are summarized in tables 12 and 13, while table 11 offers a quick summary of the experiment setups and the reference to the associated figures, for a better readability.

Table 11: IMU characterization experiments - Parameters.

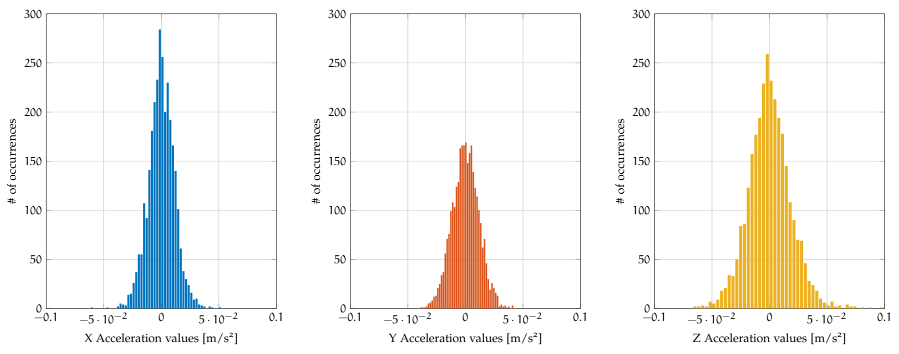
	sensors	parameters			section
		F_S	f_C^{HW}	f_C^{SW}	
set 0	accelerometers	200 Hz	42 Hz	30 Hz	58
	gyroscopes				59
set 1	accelerometers	200 Hz	10 Hz	30 Hz	60
	gyroscopes				61

Table 12: IMU characterization experiments - Results (set 0).

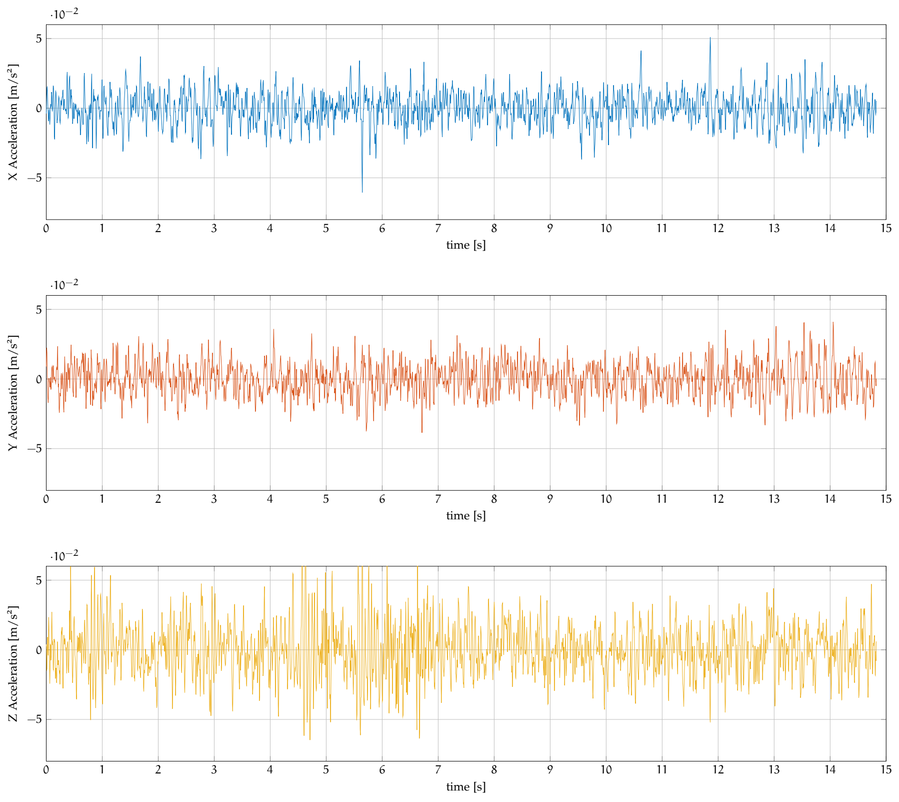
		$\bar{s}$	RMS_s	NSD_s	NSD_s (datasheet)	% error
		$[m/s^2]$	$[m/s^2]$	$\left[\frac{m/s^2}{\sqrt{Hz}}\right]$	$\left[\frac{m/s^2}{\sqrt{Hz}}\right]$	
acc.	a_x	-0.0865	0.0113	0.00186	0.00392	52%
	a_y	0.1026	0.0116	0.00189		52%
	a_z	9.8511	0.0184	0.00324		17%
		$[rad/s]$	$[rad/s]$	$\left[\frac{rad/s}{\sqrt{Hz}}\right]$	$\left[\frac{rad/s}{\sqrt{Hz}}\right]$	
gyr.	ω_x	0.000364	0.000507	0.0000864	0.0000873	1%
	ω_y	-0.000732	0.000423	0.0000718		17%
	ω_z	0.003783	0.000476	0.0000788		10%

Table 13: IMU characterization experiments - Results (set 1).

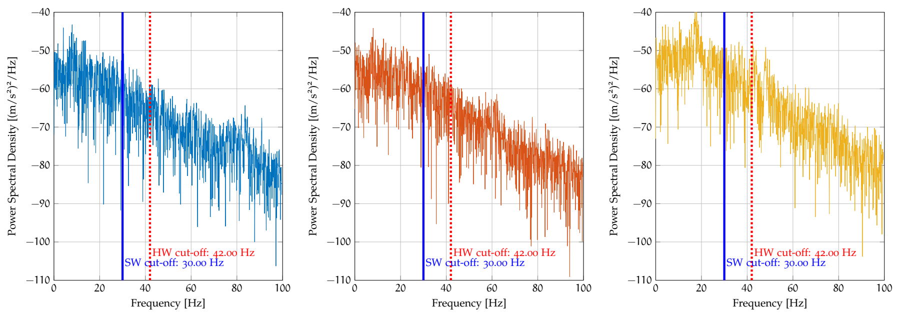
		$\bar{s}$	RMS_s	NSD_s	NSD_s (datasheet)	% error
		$[m/s^2]$	$[m/s^2]$	$\left[\frac{m/s^2}{\sqrt{Hz}}\right]$	$\left[\frac{m/s^2}{\sqrt{Hz}}\right]$	
acc.	a_x	0.0275	0.0204	0.0021	-	-
	a_y	-0.3276	0.0207	0.0020		
	a_z	10.2365	0.0298	0.0029		
		$[rad/s]$	$[rad/s]$	$\left[\frac{rad/s}{\sqrt{Hz}}\right]$	$\left[\frac{rad/s}{\sqrt{Hz}}\right]$	
gyr.	ω_x	0.008347	0.000981	0.0000962	-	-
	ω_y	0.013806	0.000854	0.0000853		
	ω_z	0.013704	0.000969	0.0000981		



(a) Accelerometers samples distribution

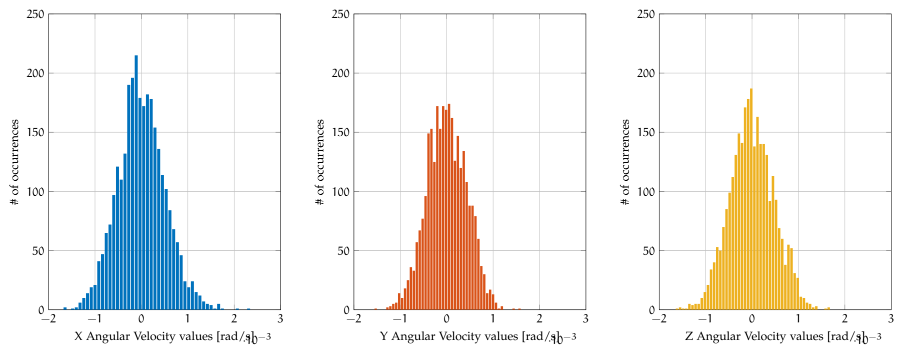


(b) Accelerometers time history

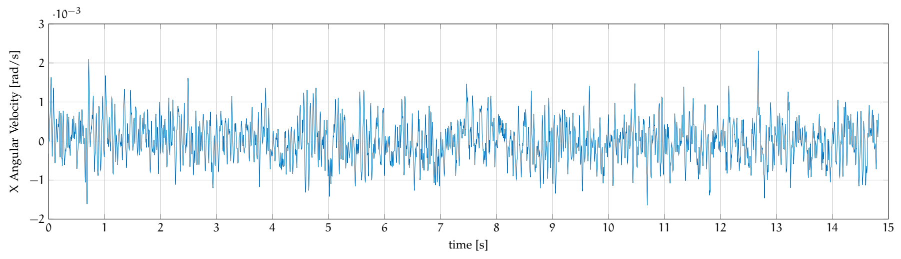
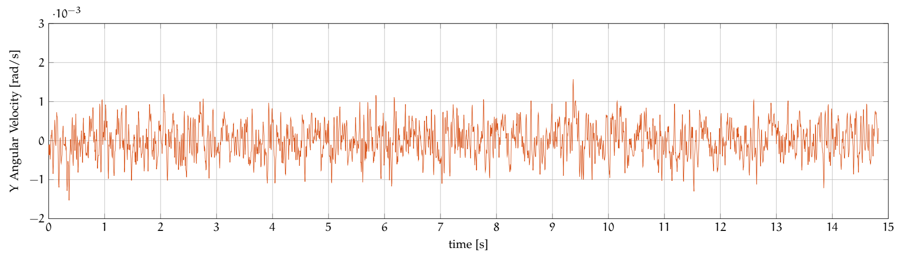
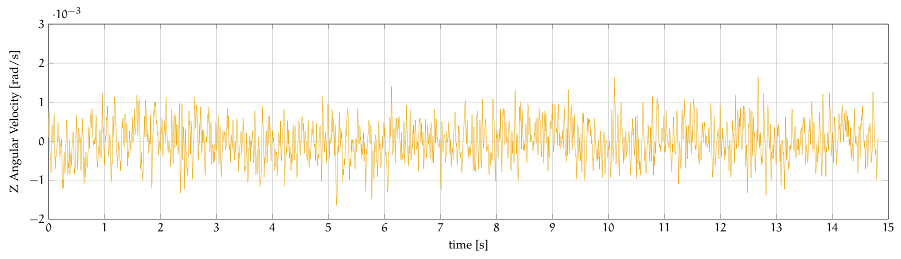


(c) Accelerometers noise spectral density

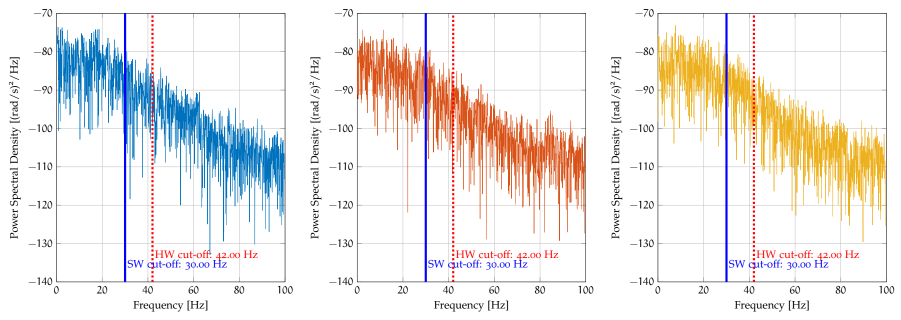
Figure 58: Measurement set 0 - Accelerometers characterization.



(a) Gyroscopes samples distribution

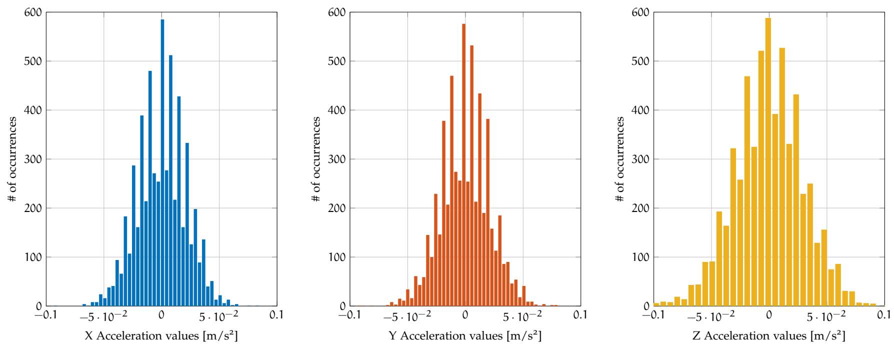




(b) Gyroscopes time history

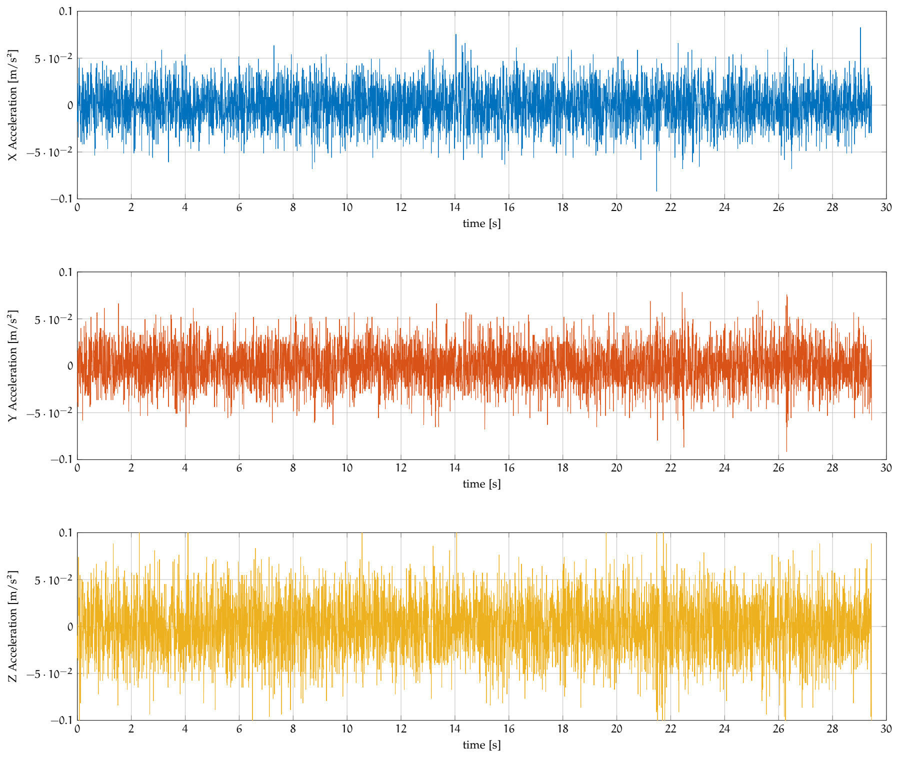


(c) Gyroscopes noise spectral density

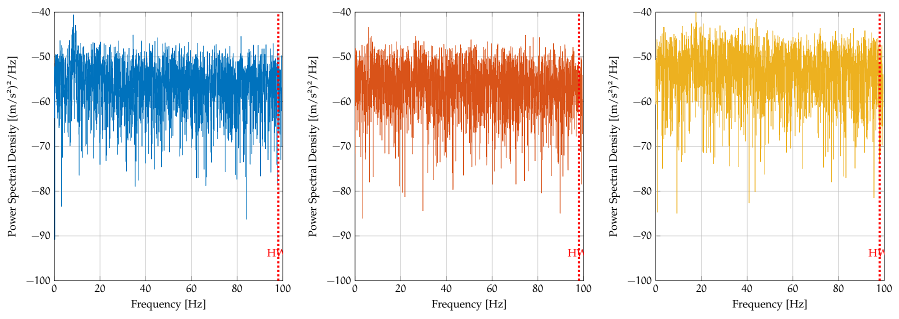
Figure 59: Measurement set 0 - Gyroscopes characterization.



(a) Accelerometers samples distribution

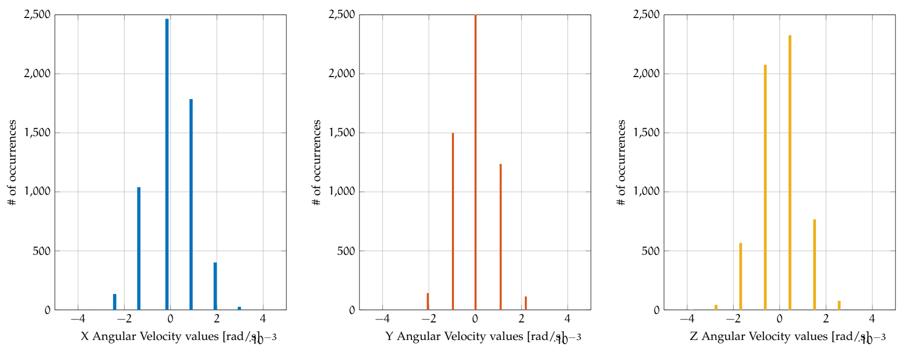


(b) Accelerometers time history

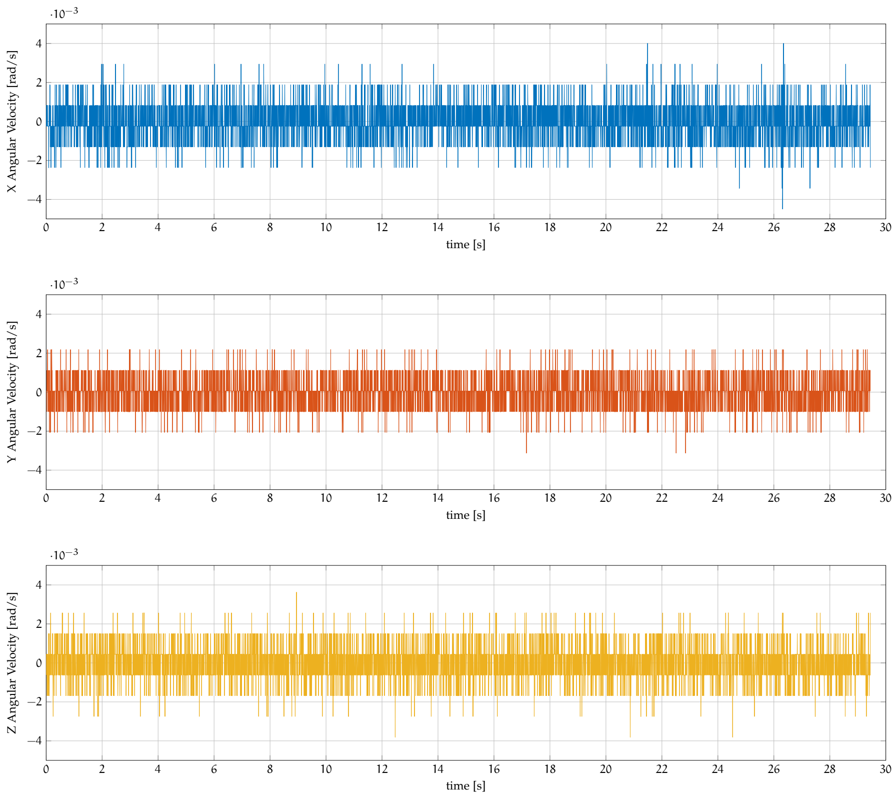


(c) Accelerometers noise spectral density

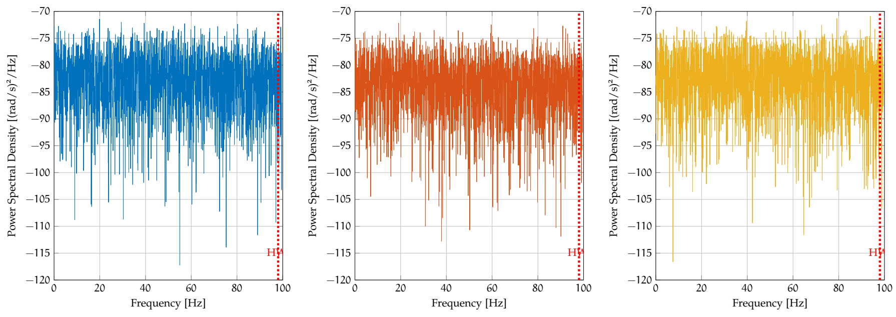
Figure 60: Measurement set 1 - Accelerometers characterization.



(a) Gyroscopes samples distribution



(b) Gyroscopes time history



(c) Gyroscopes noise spectral density

Figure 61: Measurement set 1 - Gyroscopes characterization.

BIBLIOGRAPHY

Atzori, L., A. Iera, and G. Morabito

2010 "The internet of things: A survey", *Computer networks*, 54, 15, pp. 2787-2805. (Cited on p. 25.)

Augugliaro, F., A. Schoellig, and R. D'Andrea

2013 "Dance of the flying machines", *IEEE Robotics and Automation Magazine*, 117, December. (Cited on p. 8.)

Austin, R.

2011 *Unmanned aircraft systems: UAVs design, development and deployment*, John Wiley & Sons, vol. 54.

Bouabdallah, S. and R. Siegwart

2005 "Backstepping and sliding-mode techniques applied to an indoor micro quadrotor", in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, pp. 2247-2252. (Cited on pp. 4, 35.)

Braun, R.D. and R.M. Manning

2006 "Mars exploration entry, descent and landing challenges", in *Aerospace Conference, 2006 IEEE*, IEEE, 18-pp. (Cited on p. 7.)

Carpin S., Feyzabadi S.

2014 "Risk aware path planning using hierarchical constrained Markov decision processes", in *Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering*, pp. 297-303. (Cited on p. 34.)

Castillo, P., A. Dzul, and R. Lozano

2004 "Real-Time Stabilization and Tracking of a Four-Rotor Mini Rotorcraft", *IEEE Transactions on Control Systems Technology*, 12, 4 (July 2004), pp. 510-516, ISSN: 1063-6536, DOI: 10.1109/TCST.2004.825052. (Cited on p. 16.)

Chen, Y., Z. Du, and M. García-Acosta

2010 "Robot as a service in cloud computing", in *Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on*, IEEE, pp. 151-158. (Cited on p. 25.)

Cheng, Y.

2010 "Real-time surface slope estimation by homography alignment for spacecraft safe landing", in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, IEEE, pp. 2280-2286. (Cited on p. 7.)

Cheng, Y., J. Goguen, A. Johnson, C. Leger, L. Matthies, M.S. Martin, and R. Willson

2004 "The mars exploration rovers descent image motion estimation system", *Intelligent Systems, IEEE*, 19, 3, pp. 13-21. (Cited on p. 7.)

Cheng, Y., A. Johnson, and L. Matthies

2005 "MER-DIMES: a planetary landing application of computer vision", in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, IEEE, vol. 1, pp. 806-813. (Cited on p. 7.)

Cheng, Y., A. Johnson, L. Matthies, and A. Wolf

2001 "Passive imaging based hazard avoidance for spacecraft safe landing", in *Proc. iSAIRAS*. (Cited on p. 7.)

Chi, T., Y. Ming, S. Kuo, C. Liao, et al.

2012 "Civil UAV path planning algorithm for considering connection with cellular data network", in *Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on*, IEEE, pp. 327-331. (Cited on p. 33.)

Chibani, A., Y. Amirat, S. Mohammed, E. Matson, N. Hagita, and M. Barreto

2013 "Ubiquitous robotics: Recent challenges and future trends", *Robotics and Autonomous Systems*, 61, 11, pp. 1162-1172. (Cited on p. 25.)

Corke, P., J. Lobo, and J. Dias

2007 "An introduction to inertial and visual sensing", *The International Journal of Robotics Research*, 26, 6, pp. 519-535. (Cited on p. 44.)

Crassidis, J.L.

2006 "Sigma-point Kalman filtering for integrated GPS and inertial navigation", *Aerospace and Electronic Systems, IEEE Transactions on*, 42, 2, pp. 750-756. (Cited on p. 50.)

Crick, C., G. Jay, S. Osentoski, B. Pitzer, and O.C. Jenkins

2011 "Rosbridge: ROS for non-ros users", in *Proceedings of the 15th international symposium on robotics research (ISRR)*. (Cited on p. 32.)

De Filippis, L., G. Guglieri, and F. Quagliotti

2011 "A Minimum Risk Approach for Path Planning of UAVs", *Journal of Intelligent & Robotic Systems*, 61, 1-4, pp. 203-219, ISSN: 0921-0296, DOI: 10.1007/s10846-010-9493-9. (Cited on p. 34.)

Ermacora, G., A. Toma, B. Bona, M. Chiaberge, M. Silvagni, M. Gaspardone, and R. Antonini

2013 "A cloud robotics architecture for an emergency management and monitoring service in a smart city environment", in *IROS - Cloud Robotics Workshop*, IEEE. (Cited on pp. 25, 26.)

Forster, C., M. Pizzoli, and D. Scaramuzza

2014 "SVO: Fast Semi-Direct Monocular Visual Odometry", in *Proc. IEEE Intl. Conf. on Robotics and Automation*. (Cited on p. 45.)

Goerzen, C., Z. Kong, and B. Mettler

2010 "A survey of motion planning algorithms from the perspective of autonomous UAV guidance", *Journal of Intelligent and Robotic Systems*, 57, 1-4, pp. 65-100. (Cited on pp. 33, 34.)

Grancharova, A., E. Grøtli, D. Ho, and T. Johansen

2014 "UAVs Trajectory Planning by Distributed MPC under Radio Communication Path Loss Constraints", English, *Journal of Intelligent & Robotic Systems*, pp. 1-20, ISSN: 0921-0296, DOI: 10.1007/s10846-014-0090-1, http://dx.doi.org/10.1007/s10846-014-0090-1. (Cited on p. 33.)

Grøtli, E.I. and T.A. Johansen

2012 "Path planning for UAVs under communication constraints using SPLAT! and MILP", *Journal of Intelligent & Robotic Systems*, 65, 1-4, pp. 265-282. (Cited on p. 33.)

Gutiérrez, P., A. Barrientos, J. del Cerro, and R. San Martín

2006 "Mission planning and simulation of unmanned aerial vehicles with a GIS-based framework", in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO, Paper No. AIAA*, vol. 6198. (Cited on p. 33.)

Hoffmann, G., D.G. Rajnarayan, S.L. Waslander, D. Dostal, J.S. Jang, and C.J. Tomlin

2004 "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)", *The 23rd Digital Avionics Systems Conference (IEEE Cat. No.04CH37576)*, pages, DOI: 10.1109/DASC.2004.1390847. (Cited on p. 8.)

Hoffmann, G.M., H. Huang, S.L. Waslander, and C.J. Tomlin

2007 "Quadrotor helicopter flight dynamics and control: Theory and experiment", in *Proc. of the AIAA Guidance, Navigation, and Control Conference*, vol. 2. (Cited on pp. 4, 35.)

How, J., B. Bethke, and A. Frank

2008 "Real-time indoor autonomous vehicle test environment", *Control Systems, IEEE*, 28, April, pp. 51-64. (Cited on p. 8.)

Huertas, A., Y. Cheng, and R. Madison

2006 "Passive imaging based multi-cue hazard detection for spacecraft safe landing", in *Aerospace Conference, 2006 IEEE*, IEEE, 14-pp. (Cited on p. 7.)

Hunziker, D., M. Gajamohan, M. Waibel, and R. D'Andrea

2013 "Rapyuta: The roboearth cloud engine", in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, IEEE, pp. 438-444. (Cited on pp. 25, 27.)

Jun, M. and R. D'Andrea

2003 "Path planning for unmanned aerial vehicles in uncertain and adversarial environments", in *Cooperative Control: Models, Applications and Algorithms*, Springer, pp. 95-110. (Cited on p. 33.)

Kamei, K., S. Nishio, N. Hagita, and M. Sato

2012 "Cloud networked robotics", *Network, IEEE*, 26, 3, pp. 28-34. (Cited on p. 25.)

Kaplan, E.D. and C.J. Hegarty

2005 *Understanding GPS: principles and applications*, Artech house. (Cited on p. 51.)

Koenig, S. and M. Likhachev

2001 "Incremental A*", in *NIPS*, pp. 1539-1546. (Cited on p. 34.)

2002 "D* Lite", in *AAAI/IAAI*, pp. 476-483. (Cited on p. 34.)

Langley, R.B.

1999 "Dilution of precision", *GPS world*, 10, 5, pp. 52-59. (Cited on p. 53.)

Lanza, P., N. Noceti, C. Maddaleno, A. Toma, L. Zini, and F. Odone

2012 "A vision-based navigation facility for planetary entry descent landing", in *Computer Vision–ECCV 2012. Workshops and Demonstrations*, Springer, pp. 546-555. (Cited on p. 12.)

Lindsey, Q., D. Mellinger, and V. Kumar

2012 "Construction with quadrotor teams", *Autonomous Robots*, 33, 3 (June 2012), pp. 323-336, ISSN: 0929-5593, DOI: 10.1007/s10514-012-9305-0. (Cited on p. 8.)

Lupashin, S., M. Hehn, and M.W. Mueller

2014 "A platform for aerial robotics research and demonstration: The Flying Machine Arena", *Mechatronics*, 24, 1, pp. 41-54, ISSN: 0957-4158, DOI: 10.1016/j.mechatronics.2013.11.006. (Cited on p. 8.)

Lynen, S., M.W. Achtelik, S. Weiss, M. Chli, and R. Siegwart

2013 "A robust and modular multi-sensor fusion approach applied to MAV navigation", in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, pp. 3923-3929. (Cited on pp. 44, 47.)

Mellinger, D., N. Michael, and V. Kumar

2012 "Trajectory generation and control for precise aggressive maneuvers with quadrotors", *The International Journal of Robotics Research*, 31, 5 (Jan. 2012), pp. 664-674, ISSN: 0278-3649, DOI: 10.1177/0278364911434236. (Cited on p. 8.)

Michael, N., D. Mellinger, Q. Lindsey, and V. Kumar

2010 "The GRASP Multiple Micro-UAV Testbed", *IEEE Robotics & Automation Magazine*, 17, 3 (Sept. 2010), pp. 56-65, ISSN: 1070-9932, DOI: 10.1109/MRA.2010.937855. (Cited on p. 8.)

Mitchell, H.B.

2007 *Multi-sensor data fusion: an introduction*, Springer Science & Business Media. (Cited on p. 43.)

Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng

2009 "ROS: an open-source Robot Operating System", in *ICRA workshop on open source software*, vol. 3, p. 5. (Cited on p. 26.)

Quintas, J., P. Menezes, and J. Dias

2011 "Cloud robotics: towards context aware robotic networks", in *International Conference on Robotics*, pp. 420-427. (Cited on p. 25.)

Ritz, R., M.W. Müller, M. Hehn, and R. D'Andrea

2012 "Cooperative quadrocopter ball throwing and catching", *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Oct. 2012), pp. 4972-4978, DOI: 10.1109/IROS.2012.6385963. (Cited on p. 8.)

Sa, I. and P. Corke

2011 "Estimation and control for an open-source quadcopter", *Proceedings of the Australasian Conference on Robotics and Automation 2011*. (Cited on p. 17.)

Sanfeliu, A., N. Hagita, and A. Saffiotti

2008 "Network robot systems", *Robotics and Autonomous Systems*, 56, 10, pp. 793-797. (Cited on p. 25.)

Sarris, Z.

2001 "Survey of UAV applications in civil markets", in *IEEE Mediterranean Conference on Control and Automation*, June.

Shen, S.

2014 *Autonomous navigation in complex indoor and outdoor environments with micro aerial vehicles*, PhD thesis, University of Pennsylvania. (Cited on p. 57.)

Trawny, N. and S.I. Roumeliotis

2005 "Indirect Kalman filter for 3D attitude estimation", *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, 2. (Cited on pp. 44, 46, 49.)

Waibel, M., M. Beetz, J. Civera, R. d'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J.M.M. Montiel, A. Perzylo, et al.

2011 "A World Wide Web for Robots", *IEEE Robotics & Automation Magazine*. (Cited on p. 25.)

Waslander, S.L., G.M. Hoffmann, and C.J. Tomlin

2005 "Multi-agent quadrotor testbed control design: integral sliding mode vs. reinforcement learning", *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3712-3717, DOI: 10.1109/IROS.2005.1545025. (Cited on p. 16.)

Weiss, S.

2012 *Vision based navigation for micro helicopters*, PhD thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20305, 2012. (Cited on pp. 44, 54.)

Weiss, S., M.W. Achtelik, M. Chli, and R. Siegwart

2012 "Versatile distributed pose estimation and sensor self-calibration for an autonomous MAV", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 31-38. (Cited on p. 44.)

Weiss, S. and R. Siegwart

2011 "Real-time metric state estimation for modular vision-inertial systems", in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 4531-4537. (Cited on p. 46.)

Willmann, J., F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler

2012 "Aerial robotic construction towards a new field of architectural research", *International journal of architectural computing*, 10, 3, pp. 439-460. (Cited on p. 8.)

FAA regulation

2012 *Unmanned Aircraft Systems*, http://tinyurl.com/jw42q8f. (Cited on p. 2.)

CASR regulation

2001 *Part 101 - Unmanned aircraft and rocket operations*, `http://tinyurl.com/npwnc5w`. (Cited on p. 1.)

TC regulation

2014 *Unmanned Aerial Vehicle Regulations*, `http://tinyurl.com/74egk3o`. (Cited on p. 1.)

ENAC regulation

2014 *La regolazione per la Sicurezza - Sistemi Aeromobili a Pilotaggio Remoto (SAPR)*, `http://tinyurl.com/oqaxtej`. (Cited on pp. 2, 34.)

Micropilot MP2128g homepage n.d. , `http://tinyurl.com/pue2892`.

Parrot AR.Drone homepage n.d. , `http://tinyurl.com/7dp6nnh`.

Robotics in Concert 2013 , `http://tinyurl.com/puomcre`. (Cited on pp. 27, 30.)

Open Knowledge Foundation Blog. Defining Open Data 2013 , `http://tinyurl.com/nhjq25u`. (Cited on p. 33.)

ETHZ ASL Multi Sensor Fusion Framework n.d. , `https://github.com/ethz-asl/ethzasl_msf`.

ETHZ ASL Single Sensor Fusion Framework n.d. , `https://github.com/ethz-asl/ethzasl_sensor_fusion`.