Universidad
Rey Juan Carlos

AEROSPACE ENGINEERING IN AIR NAVIGATION

Academic Course 2015/2016

Final Degree Project

Autonomus behaviour of Unmanned Air Vehicless

Author: Jorge Cano Martínez

Tutor: Jose María Cañas

Declaration of Authorship

I, Jorge Cano Martínez, declare that this end of dregree project titled, 'Autonomus behaviour of Unmanned Air Vehicless' and the work presented in it are my own. I confirm that:

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

Signed: _____

Date: _____

Abstract

“”

Jorge Cano Martínez

Contents

List of Figures

List of Tables

Abbreviations

Physical Constants

Symbols

Chapter 1

Introduction

This chapter outlines the robotics concept and a brief history, from its beginning until nowadays. Later, the development of the aerial robots and the investigation, commercial and defence use of this type of technology.

1.1 Robotics and its History

Robotics is a branch of engineering that involves the conception, design, manufacture, and operation of robots, as well as the different systems for their control and information processing. This field overlaps with, mechanical engineering, electronics, computer science, artificial intelligence, nanotechnology and lately bioengineering.

A robot is an artificial, mechanical or virtual, identity able to complete an activity autonomously, established beforehand, that substitutes humans in dangerous, difficult or special tasks. The world "robot" derive from the Czech world "robota" that means forced labour and it was firstly described in the Karel Capek novel named Rossum's Universal Robots in 1923

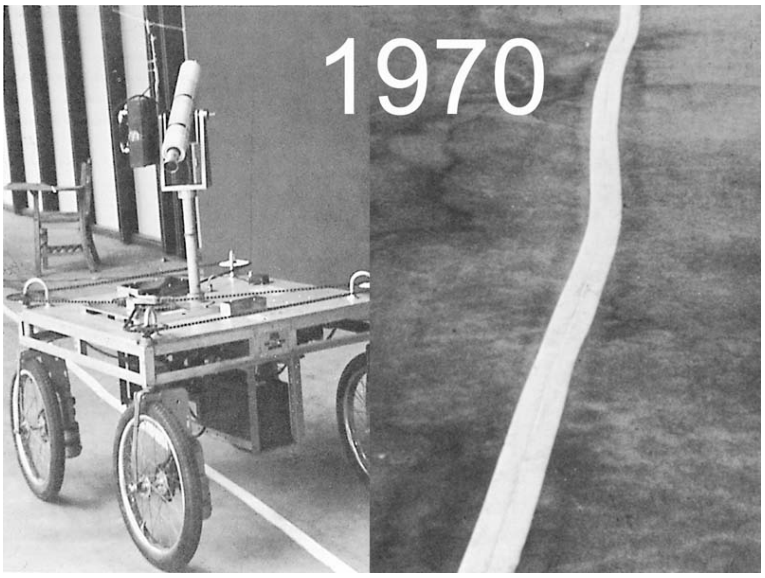
Although the term robot was coined in the 20's, these mechanical devices was already known in the world as "automatons", that are the precursors of the robots we know today. The term automaton is defined as a self-operating machine, or control mechanism designed to follow automatically a predetermined sequence of operations, or respond to predetermined instructions, that mimics the shape and the movements of an animated being

In history we can find several examples of automata, as Leonardo's robot, which was a humanoid robot designed by Leonardo Da Vinci in 1495, like many of his inventions of Leonardo, it never was constructed. The chess-playing automaton was built by engineer and mathematician Leonardo Torres Quevedo in 1912. It was presented during the Paris fair of 1914 and in 1915 the magazine Scientific American published the article "Torres and His Remarkable Automatic Device ". Using electromagnets under the chessboard, he played a certain chess play against a human opponent. With a simple algorithm evaluating the positions of the figures, not always come to mate in the minimum number of moves, but he was able to achieve victory every time without human intervention.

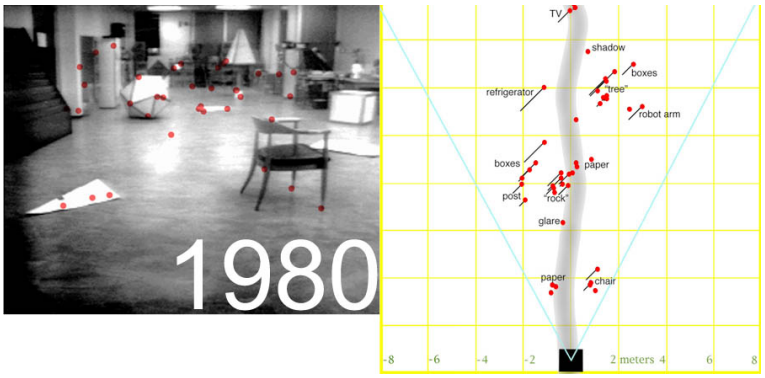
FIGURE 1.1: Torres Quevedo´s automatic device

With the advent of computers in the 70's Stanford University developed the Stanford Cart that became the first mobile robot controlled by a computer. This robot was able to follow a white line using the camera it had installed. With the data collected his camera, it was able to communicate via radio waves with a workstation, which performed calculations necessary for its movement. Another example was Shakey, it was able to reason about their own actions. Unlike other robots, they had to receive every step instructions to complete a task, Shakey could receive the task he had to perform and carry it out in smaller steps by itself.

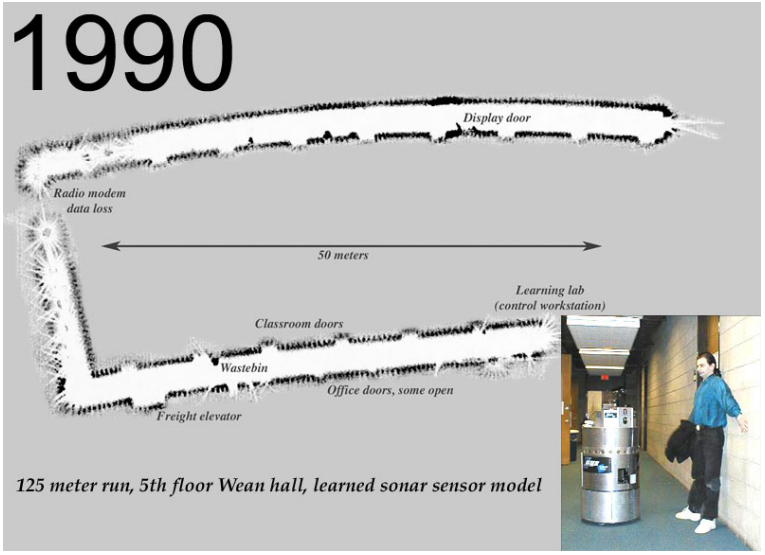
Already in the 80's, with the possibility of board computers in the robots themselves, the enhanced version Cart Stanford had two TV cameras that provided a vision stereo environment. With a limited 3D reconstruction was able to recognise objects and navigate through the environment avoiding obstacles. In the most famous experiment was performed with this robot was able to overcome various obstacles for 30 meters at 5 hours.



(a)



(b)



(c)

FIGURE 1.2: Robot history: a) Stanford Cart b) Standford Cart improvement c) Xavier Robot

In the years following the Stanford Cart, the investigation was focusing on building maps and navigate through it, as robot Xavier did. This robot was developed by Carnegie Mellow University in 1993, its purpose was to develop a mobile robot capable of operate reliable for long periods of time, learning and adapting to their environment. Since 2000 until today, robotics has starred huge breakthroughs, mainly because, the advancement of technology and the cheapening of it. Robotics has been able to conquer land, sea, and air in our planet and in other ones. They have created autonomous cars, highly sophisticated humanoids, space rovers, underwater robots and air ones.

(a) Autopia (b) Asimo (c) Opportunity

(d) Pluto (e) UAV

FIGURE 1.3: Mobile Robots

While most of these robots are mainly used for research, there are other much closer to the daily lives of people. In this section we can find some cars of the Volkswagen brand, as the Touran model already in 2007 was able to park without human intervention. Google entered the world of robotics with Google Car, driverless car capable of urban journeys. Such was the impact of this car that the state of Nevada on June 29, 2011, created a law allowing autonomous cars running on its roads. Advances in robotics technology in cars was significantly boosted by the DARPA Grand Challenge is a competition of autonomous vehicles funded by DARPA. Stanley, the most recent version, is the robotic

car from Stanford University in 2005 won the competition where cars had to go through a desert itinerary full of curves and tunnels without human intervention.

Perhaps the greatest impact of robotics on everyday life is the Roomba robot of the company iRobot. It is a robot vacuum cleaner and was the first consumer robot, was first launched to market in 2002 and now has a family of robots capable of mopping the floor, cleaning pools, polishing floors or cleaning gutters. Its last versions are probably one of the most sophisticated robots affordable by population.

(a) Stanley Robot

(b) Roomba Robot

FIGURE 1.4: Robotic automobile and Robotic vacuum cleaner

In the industrial environment robotics has reach a level of efficiency and precision never before known in the world. From automated assembly lines with robotic arms for automobile assembly, robots for packaging tasks, classifier robots and many others make up a growing robotic industrial sector. From the first automates to the last robots sent to Mars, robotics has helped man to extend and enhance human qualities. The human being has delegated robots performing the most dangerous, most repetitive or those that require a high level of accuracy to improve our current way-of-living.

(a) Assembly line

(b) Package robot

(c) Baxter Robot

FIGURE 1.5: Industrial Robots

1.1.1 Robot Components - Hardware

From an engineering point of view, a robot is a complex system equipped with sensors, actuators and processors; combined, give basic skills such as perception, action, processing and memory to interact with the environment. In a simpler way, a robot is

a computer with external peripherals that allow you to obtain information about the environment and interact with it.

- The sensors allow you to get information from the environment or itself. This capability allows a robot to work on changing scenarios by changing their behaviour depending on the environment in which it is located. There are a variety of sensors, from the most basic as sensors light, temperature or proximity to the most complex and rich as cameras, depth sensors, lasers, GPS locators, etc. in aerial robots.
- Actuators are devices that allow the robot to interact with the environment and/or perform movements. Give the robot the ability to move or change the environment in which they work. They allow the robot to perform its tasks in a real environment, such as navigation avoiding obstacles or assembling parts in an assembly. The actuators can be of various types such as motors used for moving the wheels of the robot, the motors the movement of articulations or engines spinning propellers to provide lift.
- Computers are responsible for processing the data collected by the sensors and materialising the results of the decision algorithms in actions carried out actuators. processors give the robot reasoning ability that allows the treatment of large amount of data from the sensors and the ability to execute complex algorithms in a short period of time.

1.1.2 Software

Organised information in the form of operating systems, utilities, programs, and applications that enable computers to work. The components seen in the previous point feel tangible foundations on which sits a software robot. The software is a very important element because without it, a robot would be a machine incapable of generating intelligent behaviour. Like any other development software, software for robots must meet certain requirements:

- Agility: a robot is situated in a dynamic and unpredictable scenario in constant interaction with the environment. Because of this feature the software must be flexible to provide reliable answers fast and continuously.

- Multitasking: the nature of your software robots must be multitasking, have the ability to collect sensor data, process, decide and take action simultaneously. To all this must be added the ability to react to unforeseen events, in a changing environment can interrupt a certain task.

- Distributed: sensors with increasingly more complex data processing algorithms and a large computational load, distributed computing has almost become a standard in the development of robotic applications. In addition, often coordinating multiple robots or use of other sources of data external to the robot, such as systems cameras to detect 3D robot in a controlled environment, is necessary.

- Dealing with heterogeneous hardware: often sensory and action devices are very different and each manufacturer develops its own driver. So is necessary the existence of a homogeneous interface software to unify access to sensors and actuators regardless of manufacturer or model of the device. This allows developed algorithms for a given robot, can be exported to a different one to perform a task, enabling code reuse.

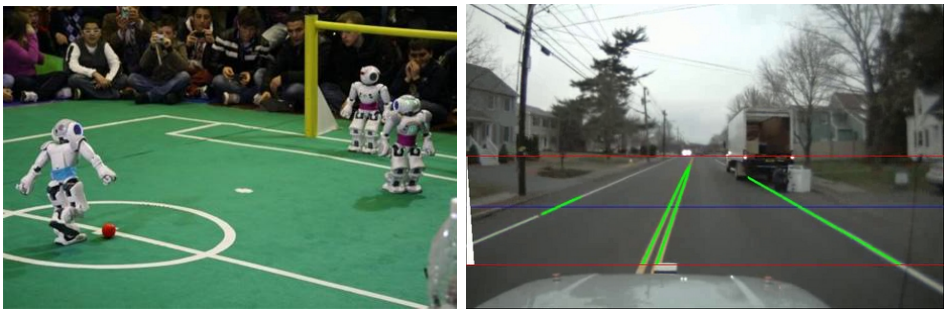
To help the developer responsible for making applications for robots, there are several software platforms (open or closed code). These platforms typically are installed on an operating system, the platform is the bridge between the robot and the developer, and take advantage of the characteristics of modern operating systems such as hardware abstraction, the multitasking capabilities and ease to create distributed applications with external libraries. Usually, if the robot model is widespread, the developer will also include simulators where you can run your code before testing in a real environment.

1.1.3 Artificial Vision

For several years more and more robots that incorporate cameras in its set of sensors. Cameras provide lot of information about the environment in which the robot is. Artificial vision or computer vision is a branch of artificial intelligence that by the use of computers is able to extract information from an image in order to understand and assimilate what is happening around them. In 1961 when Larry Roberts created a program that could see a block structure, analyse its content and replay it from another perspective, using a camera and processing the image from a computer.

Over the years, it has been shown that artificial vision is very complex, it has taken years to be resolved by a machine. However, increasingly, applications of this discipline are leaving the laboratories and are go far into the lives of people. Currently it is not difficult to find surveillance applications based on artificial vision, face recognition, industrial applications for object classification or quality controls. We can even find some applications for smartphones or at sport events.

In robotics this type of sensor has had a major impact due mainly to its low cost (compared to other sensors) and the large amount of information that can be extracted from it. It is difficult to think of a robotic project that does not contain at least one camera between its sensors. In the figure you can see the Nao robots playing a football game. To locate themselves on the field, his teammates, his opponents, the ball, field boundaries and the opposing goal, it is a technique of artificial vision for SLAM (Simultaneous Location and Mapping) uses. Also robotic cars that participated in the DARPA Urban Challenge, thanks to artificial vision algorithm, are able to detect the curves of the road and reach the goal of stay within the lane without running off it.



(a) Autopositioning based on vision (b) Curves recognition

FIGURE 1.6: Artificial vision in robotics

Fundamentals

A machine vision system is the one responsible for the formation and processing of images. For forming images could divide it into several subsystems, as the illumination subsystem, the capture and signal acquisition.

Once the image has been captured and entered into the computer we find the processing subsystem. This system is usually a computer from a digital representation of the image

and processed to extract other information of the highest level. The processing is usually organised in stages.

The first is known as reprocessing. It filters and geometric transformations are applied in order to improve the quality of the image data. The second stage, segmentation, is to isolate the image elements which, depending on the type of application, may be more interesting. For example, extract the wooded areas of an image captured by a satellite. Finally, the third stage is known as the stage of recognition or classification and involves removing features such as area or perimeter, allowing distinguish between segmented objects.

Colour Spaces

Once the video signal has passed through the acquisition subsystem, it will have generated a matrix of discrete values stored in the computer memory. Access to these data is indicating the row and column number where the row corresponds to the horizontal axis and the vertical column. Each contained in a cell in the matrix is called a pixel value. The origin of the image is typically found in the upper left corner. If the image is in black and white, the return value is the illumination, if we work in colour, will return a vector which is normally expressed in the (Red-Green-Blue) RGB system.

The human eyes perceive colour depending on the wavelength within the visible spectrum. This spectrum includes the wavelengths from 380 nm to 780 nm, thus the human brain distinguishes the different colours.

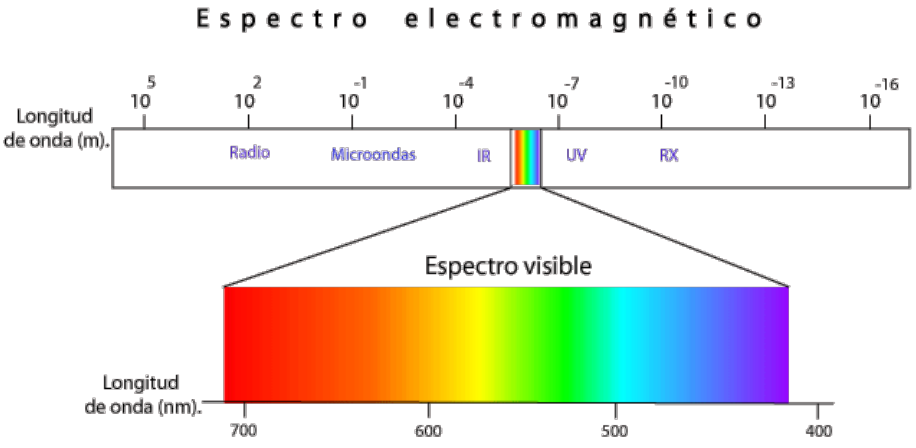


FIGURE 1.7: Visible spectrum

1.2 Aerial Robotics - Unmanned Aerial Vehicles

Aerial Robotics is the branch of robotics that deals with the study of the behaviour of autonomous unmanned aerial vehicles (UAV's). An UAV is a powered aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload. Therefore, missiles are not considered UAVs because the vehicle itself is a weapon that is not reused, though it is also unmanned and in some cases remotely guided. The flight of UAV's may operate with various degrees of autonomy: either under remote control by a human operator, or fully or intermittently autonomously, by on-board computers.

The term "*drone*", more widely used by the public, was coined in reference to the resemblance of dumb-looking navigation and loud-and-regular motor sounds of old military unmanned aircraft to the male bee. The term has encountered strong opposition from aviation professionals and government regulators.

The term *unmanned aircraft system* (UAS) was adopted by the United States Department of Defence (DoD) and the United States Federal Aviation Administration (FAA) in 2005 according to their Unmanned Aircraft System Road-map 2005–2030. The International Civil Aviation Organisation (ICAO) and the British Civil Aviation Authority (CAA) adopted this term, while the European Union's Single-European-Sky (SES) Air-Traffic-Management (ATM) Research (SESAR Joint Undertaking) road-map for 2020 also uses it. This term emphasises the importance of elements other than the aircraft. It includes elements such as ground control stations, data links and other support equipment. A similar term is an *unmanned-aircraft vehicle system* (UAVS) *remotely piloted aerial vehicle* (RPAV), *remotely piloted aircraft system* (RPAS). Many similar terms are in use.

In 1898 the US Government granted Nikola Tesla a patent for a method capable of controlling the movement mechanism of vessels or vehicles, the patent covers any type of vessel or vehicle can be driven and directed. During the following decades, the Pentagon plan to invest billions of dollars to create a new generation of unmanned vehicles for various military applications on land, sea and air. The first attempt to create a UAV was during the First World War. In 1916, the Royal Flying Corps researchers proposed

using civilians to find a way whereby they can remotely control a plane, the project was known as the "Aerial Target". In the 1930s the British Royal Navy developed a primitive Radio controlled UAVs, the Queen Bee, this drone could be reused for other flights and was able to reach 160 km / h. The Queen Bee served mainly for use as target air defences.

In the 60's and 70's, the United States conducted more than 34,000 surveillance flights using a VNAT called AQM-34 Ryan Firebee. This drone was launched from another plane where operators who were in charge of the unit were teleoperate. The United States also used another UAV called Lightning Bugs which was launched from a C-130 missions in China and Vietnam.

FIGURE 1.8: AQM-34 Ryan Firebee

In recent years, a new scientific and technological challenge in this area has been to get these vehicles completely autonomous, that is, they can fly without driving, even supervision of a person. As technology advances and costs are lowered, they are being integrated into these vehicles increasingly more advanced technologies such as information rich (cameras, lasers ...), batteries with greater autonomy, most powerful computing systems, sensors, etc.. The cheapening of UAVs technology has produce and increasing use of it in research centers throughout the world to have these vehicles and getting achievements that were unthinkable a few years ago. This fact is increasing the scope of use of UAVs, have been used only in military environments could be used for forest services, agriculture, environment, hydrology, cartography, natural disasters or geology.

However, and despite the great progress made in recent years, there are still major obstacles to overcome. Not only technological, but ethical and regulatory developments for security reasons. The aerial robotics just started what will surely go a great way.

1.2.1 Classification of UAV's

UAVs typically fall into one of six functional categories, although multi-role airframe platforms are becoming more prevalent:

- Target and decoy: providing ground and aerial gunnery a target that simulates an enemy aircraft or missile.
- Reconnaissance: providing battlefield intelligence.
- Combat: providing attack capability for high-risk missions.
- Logistics: delivering cargo.
- Research and development: improve UAV technologies.
- Civil and commercial UAVs: agriculture, aerial photography, data collection.

FIGURE 1.9: Ikhana UAV

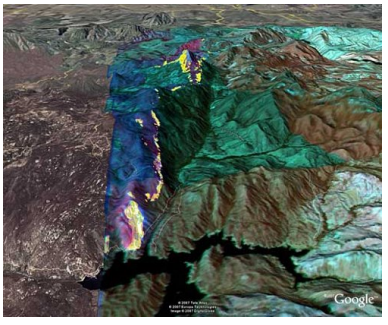


FIGURE 1.10: Image taken by one of the sensors of the investigation Ikhana UAV, belowed to the NASA, about Harris Fire in San Diego

Vehicles can be categorised in terms of range/altitude. The following has been advanced as relevant at industry events such as ParcAberporth Unmanned Systems forum:

- Hand-held 2,000 ft (600 m) altitude, about 2 km range

- Close 5,000 ft (1,500 m) altitude, up to 10 km range
- NATO type 10,000 ft (3,000 m) altitude, up to 50 km range
- Tactical 18,000 ft (5,500 m) altitude, about 160 km range
- MALE (medium altitude, long endurance) up to 30,000 ft (9,000 m) and range over 200 km
- High-Altitude Long Endurance (high altitude, long endurance - HALE) over 30,000 ft (9,100 m) and indefinite range
- Hypersonic high-speed, supersonic (Mach 1–5) or hypersonic (Mach 5+) 50,000 ft (15,200 m) or suborbital altitude, range over 200 km
- Orbital low earth orbit (Mach 25+)
- CIS Lunar Earth-Moon transfer
- Computer Assisted Carrier Guidance System (CACGS) for UAVs

FIGURE 1.11: Orbiter UAV of kind Handheld

FIGURE 1.12: Orbiter UAV land station

Other categories include:

- Hobbyist UAVs
- Midsize military and commercial drones
- Large military-specific drones
- Stealth combat drones

Classifications according to aircraft weight are quite simpler:

- Micro Air Vehicle (MAV) - the smallest UAVs that can weight less than 1g.
- Miniature UAV (also called SUAS) - approximately less than 25 kg.
- Heavier UAVs.

1.2.2 Multi-Rotor Aircraft

UAVs introduced in the previous chapters have CTOL (Conventional Take-Off and Landing) capabilities, vehicles that need a runway for take-off and landing. The UAVs type of VTOL (Vertical Take-Off and Landing) are vehicles with the ability to hover and vertical take-off and landing. These vehicles are able to maintain the hover height and direction, thus remain static at a point at a certain height. A clear example of VTOL vehicle type are helicopters, although there VTOL aircraft capabilities as the aircraft Harrier or the Boeing V-22 Osprey.

FIGURE 1.13: Boeing V-22 Osprey

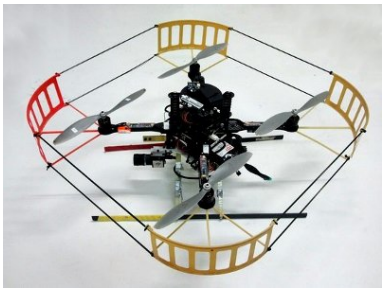


FIGURE 1.14: Pelican Quadcopter

A multirotor or multicopter is a VTOL rotorcraft with more than two rotors. An advantage of multirotor aircraft is the simpler rotor mechanics required for flight control. Unlike single- and double-rotor helicopters which use complex variable pitch rotors whose pitch varies as the blade rotates for flight stability and control, multirotors often use fixed-pitch blades; control of vehicle motion is achieved by varying the relative speed of each rotor to change the thrust and torque produced by each.

Due to their ease of both construction and control, multirotor aircraft are frequently used in radio control aircraft and UAV projects in which the names tricopter, quadcopter,

hexacopter and octocopter are frequently used to refer to 3, 4, 6 and 8 rotor helicopters, respectively.

In order to allow more power and stability at reduced weight coaxial rotors can be employed, in which each arm has two motors, running in opposite directions (one facing up and one facing down).

Flight Principles

As airplanes or helicopters, multirotors can fly due to movement of a wing (rotor blades) through the air. In this case the blades describe a circular movement unlike planes that need horizontal movement of the vehicle to generate lift. Because the air passes through a wing, a pressure differential is produced. The pressure in the upper surface (extrados) is less than the pressure at the bottom surface (intrados). This pressure difference results in the lift force.

When in an aircraft the lift is greater than the weight, it begins to fly. However, the ability to generate lift is not the only problem facing the VTOL vehicles. If we have a helicopter with a single motor, with the ability to generate sufficient lift to raise the vehicle off the ground, the vehicle would turn around the motor shaft axis in the opposite direction to the blades rotation. This is because the aerodynamic force resultant of the propeller has a horizontal component opposite the sense of rotation of the blades. This phenomenon induces a momentum parallel propeller rotor axis. For maintaining an stable flight, this momentum has to be compensated; in a conventional helicopter, this is done by adding a tail rotor, that spinning perpendicular to the main one, compensates that momentum

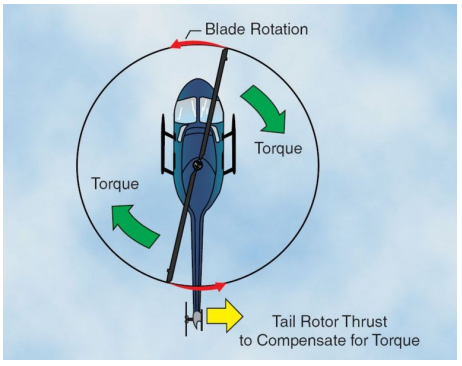
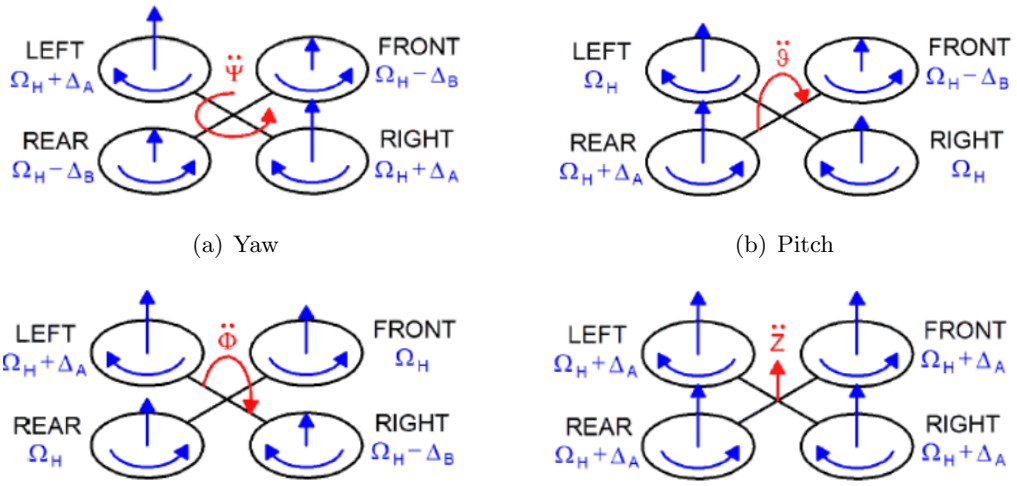


FIGURE 1.15: Torque compensation

Multirotors, take advantage of these momentum generated by its rotor to stabilise itself and manoeuvre. This vehicles typically uses pairs of rotor propellers to compensate each other momentum and varies their spinning velocity to achieve and control the momentum desired. The simplest way to explain this quality is with a quadcopter (4 rotors vehicle).



(a) Yaw (b) Pitch

(c) Roll (d) Throttle

FIGURE 1.16: Rotors configurations

With this configuration, based on the typical aeronautic body reference frame (x-front, y-right, z-down), if the rotors rotating in the counter-clockwise are fed with more power, the vehicle would turn to the right. This shift in the horizontal plane, which makes the vehicle turning on itself, is called yaw.

In the case of extend the power to the rotors one end, for example the rear, the front, the quadcopter would shift causing the vehicle turn head towards the floor because the raise of lift vector in that pair of rotor; notice that the momentum in each rotor is also increased, but as they do it the same amount and have opposite directions, the rise in yaw momentum is compensate. This movement is known as pitch, and produce the horizontal linear movement of the aircraft due to the inclination of the lift vector. The same effect is applied in different pair of rotors to produce the lateral inclination movement or roll and its corresponding lateral movement.

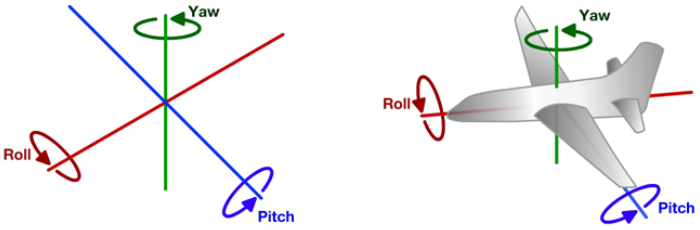


FIGURE 1.17: Roll, Pitch and Yaw

With these setup, we reach the 6 DOF (Degrees-Of-Fredom) desired for a complete control of the flight of a multirotor: linear velocities (x,y,z) and angular velocities (pitch, roll and yaw). [referencia de los modelizados]

1.3 Sate Of The Art

Since the born of UAV's, this vehicles has reach a great deal of application spread in different fields. Some of the most prestigious companies, research centres or even governments has noticed the huge potential of this technology, and have invest an uncountable amount of resources on the development of UAV's. This industry id nowadays suffering an enormous change and every day still growing new projects in this field. From a general point of view, the world of UAV's could be describe dividing in the following fields:

1.3.1 Research

ETH Zürich University, the Institute for Dynamic Systems and Control, is possibly the most important research centres in the UAV's sector. Its facilities include the Flying Machine Arena (FMA) [http://www.flyingmachinearena.org/] where a group of researchers conducted their experiments on the control of UAV's. This laboratory has a high precision motion capture system, a wireless network and software developed by themselves that, using very sophisticated algorithms allow estimation and control of UAV's. The motion capture system is able to locate multiple objects at a rate that exceeds 200 frame per second, UAVs in space system can move at a speed of 10 m / s, which is about 5 centimetres to move between two successive catches. The information in this system intersects with other data and dynamic system models to predict the state of an object

in the future. The system uses this data to determine the following command must be executed in the vehicle for a certain movement, and then the wireless network this command is sent to the vehicle performing the action.

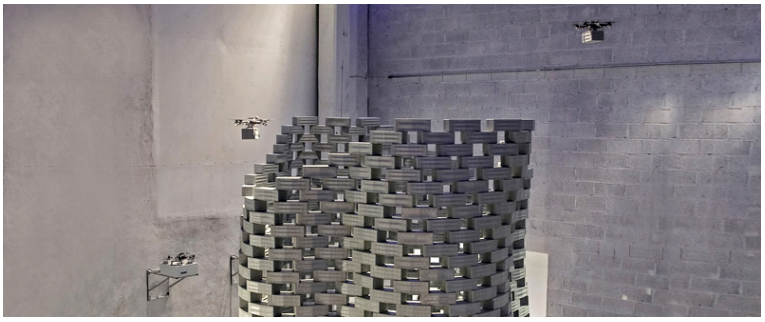


FIGURE 1.18: UAV building a tower

La Figura 1.18 anterior muestra uno de los experimentos realizados por este grupo. Se trata de la construcción de una torre de 6 metros de altura que necesito 1500 ladrillos de espuma. El experimento se realizó con la colaboración de los arquitectos Gramazio & Kohler, con la idea de explorar el potencial de los cuadricópteros en la construcción de edificios. La imagen fue captada en un museo de arquitectura en París.

FMA group carry on several research projects, in which it could be find interesting projects as:

- "Quadcopter Pole Acrobatics" [reference] system that allows quadcopters to balance an inverted pendulum, throw it in to the air, and catch and balance it again on second vehicle. Based on first principles models, a launch condition for the pole is derived and used to design an optimal trajectory to throw the pole towards a second quadcopter. An optimal catching instant is derived and the corresponding position is predicted by simulating the current position and velocity estimates forward in time. Algorithm is introduced that generates a trajectory for moving the catching vehicle to the predicted catching point in real time. By evaluating the pole state after the impact, an adaptation strategy adapts the catch maneuver such that the pole rotates into the upright equilibrium by itself.

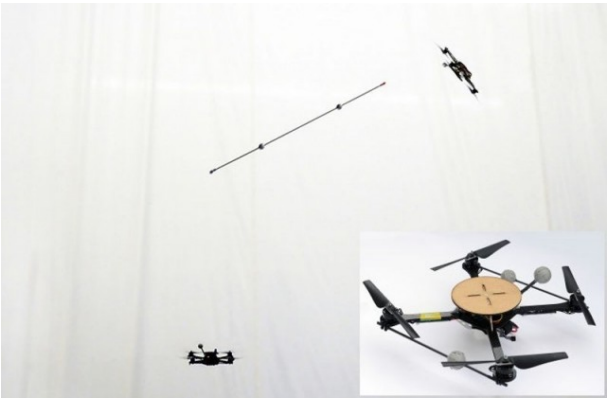


FIGURE 1.19: Poleacro quadcopter

- "Omni-Directional Aerial Vehicle" [reference]design and control of a novel six degrees-of-freedom aerial vehicle. Based on a static force and torque analysis for generic actuator configurations, we derive an eight-rotor configuration that maximises the vehicle's agility in any direction. The proposed vehicle design possesses full force and torque authority in all three dimensions. A control strategy that allows for exploiting the vehicle's decoupled translational and rotational dynamics is introduced. A prototype of the proposed vehicle design is built using reversible motor-propeller actuators and capable of flying at any orientation.

- "Tailsitter" an hybrid vehicle that takes off vertically like a multirotor but is also able to fly horizontally like a fixed-wing airplane, a new algorithm for robustly controlling a tailsitter flying machine in hover position. Using the algorithm, the tailsitter is able to recover from any orientation, including upside down.

- "Distributed Flight Array" [reference] is a flying platform consisting of multiple autonomous single propeller vehicles that are able to drive, dock with their peers, and fly in a coordinated fashion. Once in flight the array hovers for a few minutes, then falls back to the ground, only to repeat the cycle again.

The SHERPA project [http://www.sherpa-project.eu/sherpa/], found by European Union, aims to develop a mixed ground and aerial robotic platform to support search and rescue activities in a real-world hostile environment like the alpine scenario.

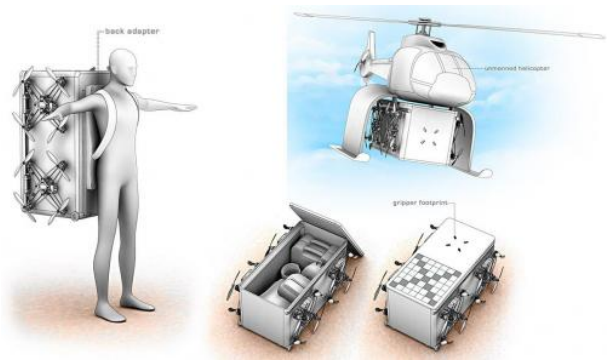


FIGURE 1.20: Sherpa Box

The technological platform and the alpine rescuing scenario are the occasion to address a number of research topics about cognition and control pertinent to the call.

What makes the project potentially very rich from a scientific viewpoint is the heterogeneity and the capabilities to be owned by the different actors of the SHERPA system: the "human" rescuer is the "busy genius", working in team with the ground vehicle, as the "intelligent donkey", and with the aerial platforms, i.e. the "trained wasps" and "patrolling hawks". Indeed, the research activity focuses on how the "busy genius" and the "SHERPA animals" interact and collaborate with each other, with their own features and capabilities, toward the achievement of a common goal.

The ARCAS (Aerial Robotics Cooperative Assembly System), [http://www.arcas-project.eu/] project proposes the development and experimental validation of the first cooperative free-flying robot system for assembly and structure construction. The project will pave the way for a large number of applications including the building of platforms for evacuation of people or landing aircraft, the inspection and maintenance of facilities and the construction of structures in inaccessible sites and in the space.

ARCAS is funded by the European Union and involving two Spanish universities, "Universidad de Sevilla" and the "Universidad Politécnica de Cataluña".

1.3.2 Defence and Space

US army is leading the defence sector on UAV's, The Air Force Research Laboratory (AFRL) [http://www.wpafb.af.mil/AFRL/] is a scientific research organisation operated by the United States Air Force Materiel Command dedicated to leading the discovery,

development, and integration of affordable aerospace war-fighting technologies, planning and executing the Air Force science and technology program, and provide war-fighting capabilities to United States air, space, and cyberspace forces.

Since the Laboratory's formation in 1997, it has conducted numerous experiments and technical demonstrations in conjunction with NASA, Department of Energy National Laboratories, DARPA, and other research organisations within the Department of Defence. The majority of the work of AFRL is considered qualified information.

Among other governments and organisations, in Spain we have INTA (Instituto Nacional de Técnica Aeroespacial) [http://www.inta.es/programasAltaTecnologia.aspx?Id=1&SubId=3]. For years INTA have been working in a research program to develop the necessary technologies to design and build a range of unmanned aircraft. The result of these activities, the Institute has developed the following products:

- SIVA: A sophisticated unmanned aerial surveillance multiple applications in the civil and military field, which can be used as a vehicle for real-time observation.
- ALO: It is an observation system low cost and high reliability suitable for the acquisition of aerial images in civilian and military missions short range.
- ALBA: It is a complete system of aerial guided target suitable for improving the operation of anti-aircraft artillery units through their training under real fire.
- MILANO [reference]: It is a strategic system of monitoring and observing composed by UAV's linked via satellite with a ground control station. From the station is planned, monitors and controls both the aircraft and the payloads shipped. The aircraft have a range of more than 20 hours and can operate at altitudes up to 26,000 feet.
- DIANA [reference]: A system of high-performance aerial target developed to simulate real threats, It can reach 200m/s and has a flight mode call "roza-olas" that allows the vehicle to fly under 15m height in the sea. Because of its versatility, the system can be used as air training system for large numbers of current and future weapons.

FIGURE 1.21: Diana UAV

In addition to these developments, the Institute continues to develop drone new generation of mini UAV's and micro UAV's.

1.3.3 Commercial

In December 2013, Amazon surprised the world showing interest in sending packets with drones. According to the company it is currently in the phase of research and testing with some UAV's eight rotors that are capable of lifting loads of 2.3 kg and deliver the package over a period of 30 minutes since the customer can do their order to through the web. In the same month, the DHL shipping company showed interest in the use of drones for parcel deliveries. They have some UAV's with a load capacity of 2.99 kg and a range of 1km. Not only are interest in shipping goods with these devices, the Madrid City Council in

Google's Project Wing is the code name for the company's drone delivery service it hopes to launch in 2017. The company first publicly announced Project Wing in a YouTube video back in 2014, but since then has been keeping mum about how this service will work. Now a new patent filing from the company reveals how part of Project Wing deliveries could deliver packages.

FIGURE 1.22: Googlewing Wing

Packaging delivering is the most shown use of commercial UAV's. However there exist worldwide a huge offer of vehicle with very different purposes. Some examples are:

- Matternet [http://mttr.net/] is an American start-up that has developed a system based on UAV's for sending supplies to the poorest regions in the sub-system developed countries. Often the population in these regions have to travel kilometres through desert terrain, large rivers, lakes, mountains or geographical depressions. Existing communication infrastructure in many regions are inefficient or simply nonexistent.

FIGURE 1.23: Matternet Proyect

- DeltaDrone [http://www.deltadrone.com/fr/] is a French company that is dedicated to providing various services with UAV's. Quadricopters and planes have to perform their functions and work in areas such as research, industry, safety and study the natural environment for government institutions and energy agencies. It is part of the Aerospace Valley which is a group of companies and aerospace research centres located in France.

- Ixion [http://www.ixion.es/] is a Spanish company born with the aim of automation of society. It is created as a TCP Systems and Engineering spin-off company. It offers its customers the development of monitoring and control of infrastructure, development of unmanned vehicle systems, machine vision systems for surveillance and development R & D with public and private institutions. Ixion develops navigation systems that allow vehicles to carry out their tasks with a high degree of

autonomy. They are based on advanced sensor systems that perceive their environment. Data from these sensors intersect with those obtained by the inertial unit and GPS information in order to obtain a safe and reliable navigation.

- Aerotools-UAV [http://www.aerotools-uav.com/] is a Spanish company dedicated to designing, manufacturing, marketing and operation of systems with UAV's. Also they collaborate with universities and research centres in process R & D for the development of quadricopters. They offer services to various sectors such as the use of these vehicles in civil, industrial or military applications.

1.3.4 Competition

Few years ago, competition for UAV's have been growing worldwide. In these competitions, where teams have to compete in creating an application to overcome a particular challenge it is to encourage research into these devices. They are normally focused on college students although some competitions for high school students.

The UAV Outback Challenge [https://uavchallenge.org/] is an annual competition for the development of unmanned aerial vehicles. The competition was first held in 2007 and features an open challenge for adults, and a high-school challenge. The event is aimed at promoting the civilian use of unmanned aerial vehicles and the development of low-cost systems that could be used for search and rescue missions. The event is one of the largest robotics challenges in the world and one of the highest stakes UAV challenges, with $50,000 on offer to the winner of the Open segment of the Challenge. The events involve a thorough scoring system with an emphasis on safety, capability and technical excellence. In particular there is a strong tendency towards autonomous flight. Notably, teams share technical details of their entries, allowing successful innovations to proliferate and increasing the speed of technological development.

The format of the Challenge changes as technological improvements make the tasks more achievable. In 2011 it changed to a search and rescue challenge. No teams successfully completed the challenge until 2014 when several teams were successful. In 2016 the open challenge will change to an automated medical retrieval task.

IARC (International Aerial Robotic Competition) [http://www.aerialroboticscompetition.org/] is an aerial robotics competition, founded in 1991 at the Georgia Institute of Technology,

in competing universities around the world. The main purpose of the competition is to advance research behaviors aerial robotics. From 1991 through 2013 they have proposed a total of 7 missions. Each is to develop a fully autonomous behaviours that is able to pass the test. New missions are created every time the above is overcome, so, a mission may be years unaddressed until one team achieves the goal.

FIGURE 1.24: UAV inspectioning a building in IARC

Chapter 2

Objectives

In this chapter the purpose and goals and requirements of the End-of-degree project are going to be describe, as well as the working methodology and work plan to achieve it successfully.

2.1 Project Objective

The general achievement of the project is to design and build an UAV vehicle, provide it with step-up autonomous intelligence and make it capable to fulfil an outdoor society useful mission, in the most robust and efficient way possible. For meeting the objective, the problem have been divided into three simpler goals:

- Design and construction of an open-source aerial vehicle able to offer to this project and future ones s reliable and effective platform for a long research.
- Program the desire software to provide the vehicle connection with JdeRobot [http://jderobot.org] infrastructure and supply all the communication channels and information offerd by the vehicle sensors to design an autonomous task.
- Plan and execute an useful autonomous mission, exportable to real society problem solutions.

2.2 Requirements

The requirements of the project have to be define in order to ensure its consistency, its correct development and to do not lose focus in any step of the project.

- The developed infrastructure integrated in JdeRobot platform.
- The infrastructure would perform its mission having concern of its level of abstraction, disturbing as less as possible higher or lower levels obs abstraction, as the vehicle stabilisation process for example.
- The interfaces between components would make use, as much as possible, the existing ones; defined in JdeRobot and autopilot source code.
- All the supplied and resulting code should be open-source, for future projects to take advantages from this one.
- The communication channels should stream and be reliable.
- The infrastructure should make an efficient use of the computational resources.

2.3 Working Methodology

Given the nature of the project, and like any other engineering project, using a model that defines the life cycle of the application is necessary. For the development of this project we have decided to adopt the spiral development model. This model defined by Barry Boehm in 1986, is based on a spiral in which each iteration represents a set of activities. These activities do not have priority, they will be chosen at the stage of risk analysis. Thus, each iteration is divided into the following activities:

- Determine targets: this activity in order to define the current iteration. Following this model the final goal of the project is divided into sub-goals.
- Risk Analysis: This activity is carried out several studies in order to meet potential threats or unwanted events that may occur in the current target.

- Develop and test: at this point verify the correct operation performed in iteration to correct the errors and that they do not continue in the following iterations will be necessary.

- Planning: In this activity the previous phases will be reviewed to determine whether they should continue with the following activities.

To realise these activities and iterations we held weekly meetings throughout the development work. Thus, every week a new subgoal was established. If the previous had been completed, we were planning the next. If, however, had not been completed, make the current target deeper to correct errors or re-plan it. During the course of the project a web logbook [http://jderobot.org/J.canoma-tfg] was agreed and the different goals of the project where published. In addition, all the source code generated is kept in a repository with version control system (GitHub [https://github.com/]) accessible from the Internet. Thus the tutor and any other interested person has access at any time to code.

2.4 Work Planning

Once project have been consider validated, the execution phase of the project has follow the next procedure:

1. Design and construction of the vehicle platform

   a) Conception: goals evaluation, design requirements establishment for the correct fulfilment of the achievements and platform type choice between all available (fix wing, helicopter, multirotor, etc...).

   b) Design: market study, preliminary design and final components choice regarding requirements established, resulting on a theoretical platform design with estimated specifications

   c) Integration: construction of the platform itself and connection of all the component for making them work properly. Also integrating the components in an space efficient way.

d) Initial configuration: auto pilot initialisation and configuration for our platform in concrete.

e) Flight test: verification of the correct performance of the platform as a system.

2. Development of the Driver

a) Learn software theory background: JdeRobot infrastructure, robotics, computer vision and communication protocol; theory necessary for the correct development of the driver.

b) Develop the driver software: server that supply to the infrastructure all the communication channels necessary and provide information captured by sensors.

c) Test driver software: verify the correct performance of the driver.

3. Client software and Autonomous mission

a) Client software development: software that interprets the data acquired and enables decision making.

b) Mission concept: mission plan, risk analysis and validation

c) Client-mission integration: Program the mission, carry out the decision making and send actuation commands.

d) Final Flight test: verify the correct performance of the whole project.

Chapter 3

Infrastructure

This chapter describes the infrastructure that this project is based on; mainly software in which the project use as starting point. The code developed in this project trust on the performance of the software following described.

3.1 Ardupilot

ArduPilot [http://ardupilot.org/ardupilot/] (also ArduPilotMega - APM) is an open source UAV's platform, able to control autonomous multicopters, fixed-wing aircraft, traditional helicopters and ground rovers. It was created in 2007 by the DIY Drones community. It is based on the Arduino open-source electronics prototyping platform. The first Ardupilot version was based on a "*thermopile*", which relies on determining the location of the horizon relative to the aircraft by measuring the difference in temperature between the sky and the ground. Later, the system was improved to replace "*thermopile*" with an Inertial Measurement Unit (IMU) using a combination of accelerometers, gyroscopes and magnetometers. Ardupilot is an award winning platform that won the 2012 and 2014 UAV Outback Challenge competitions.

ArduPilot contains a control system on board making use of sensors, it is capable of automatically manoeuvres such as take-off, stabilisation, landing and the possibility to hover at a point at a certain height. This system allows configuration of certain parameters and reception values for the movement of the drone and plug a great variety of sensors for configure the vehicle for the purpose desired.

Today, the ArduPilot project has evolved to a range of hardware and software products, including the APM and Pixhawk/PX4 [https://pixhawk.org] line of autopilots, and the ArduCopter, ArduPlane and ArduRover software projects.

The free software approach from Ardupilot is similar to that of the PX4/Pixhawk and Paparazzi Project [https://wiki.paparazziuav.org/], where low cost and availability enables its hobbyist use in small remotely piloted aircraft, such as micro air vehicles and miniature UAV's.

Features:

- C++ open source based code.
- High quality auto-level and auto-altitude control.
- Offers both enhanced remote control flight (via a number of intelligent flight modes) and execution of autonomous missions.
- Worldwide spread and tested with a large community support.
- Multiple sensor options.
- Allows different communication channels (MAVLink protocol).
- Endless options for customisation and expanded mission capabilities.

3.2 APM Mission Planner

Mission Planner [http://ardupilot.org/planner/] is a full-featured ground station application for the ArduPilot open source autopilot projectdeveloped by Michael Oborne and is part of ArduPilot project. It can be used as a configuration utility or as a dynamic control supplement for autonomous vehicles.

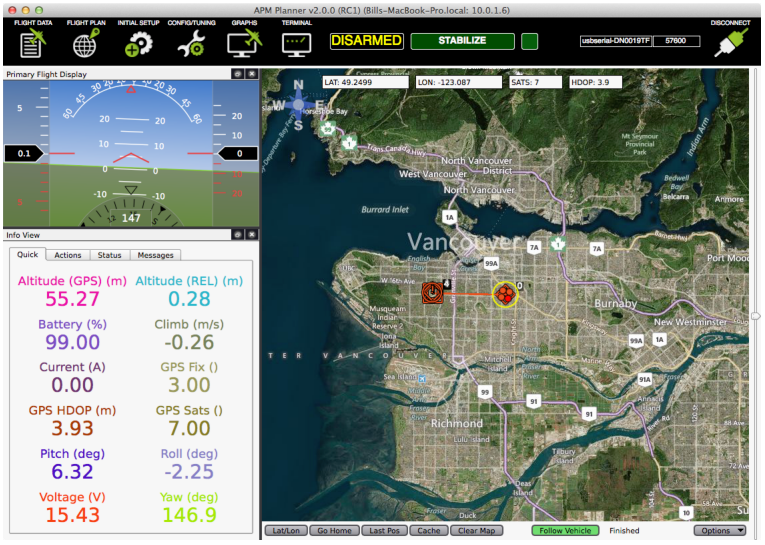


FIGURE 3.1: Mission Planner

- Load the firmware (the software) into the autopilot (APM, PX4...) that controls te vehicle.
- Setup, configure, and tune the vehicle for optimum performance.
- Plan, save and load autonomous missions into the autopilot with simple point-and-click way-point entry on Google or other maps.
- Download and analyse mission logs created by the autopilot.
- Interface with a PC flight simulator to create a full hardware-in-the-loop UAV simulator.
- With appropriate telemetry hardware, monitor the vehicle's status while in operation.

3.3 MAVLink Protocol

MAVLink (Micro Air Vehicle Communication Protocol) [http://qgroundcontrol.org/mavlink/] is a very lightweight, header-only message marshalling library for micro air vehicles.

It can pack C-structs over serial channels with high efficiency and send these packets to the ground control station. It is extensively tested on the PX4, PIXHAWK, APM and Parrot AR.Drone platforms and serves there as communication backbone for the

MCU/IMU communication as well as for Linux interprocess and ground link communication. MAVLink was redeveloped by Lorenz Meier [http://people.inf.ethz.ch/lomeier/].

MAVLink messages are defined in XML and then converted to C/C++, C# or Python code, every message is identifiable by the ID field on the packet, and the payload contains the data from the message. The general message structure is divided in 8 fields:

- Start-of-frame: Denotes the start of frame transmission
- Pay-load-length: length of payload (n)
- Packet sequence: Each component counts up his send sequence. Allows to detect packet loss.
- System ID: Identification of the SENDING system. Allows to differentiate different systems on the same network.
- Component ID: Identification of the SENDING component. Allows to differentiate different components of the same system
- Message ID: Identification of the message - the id defines what the payload "means" and how it should be correctly decoded.
- Payload: The data into the message, depends on the message id.
- Cyclic redundancy check: Check-sum of the entire packet, excluding the packet start sign (LSB to MSB)

Message detailed fields could be found here [https://pixhawk.ethz.ch/mavlink/].

EXAMPLE???

However, this protocol do not limits to a message marshalling library, MAVLink ecosystem encompass a great variety of programs and applications for the use of this protocol. this is one of the reasons why it has become the most popular protocol between MAV (micro aerial vehicle) developers, a great deal of projects are based on this protocol, such as ArduPilot, ETH Flying Machine Arena or Sky-Drones [http://www.sky-drones.com/] among others.

3.4 MAVProxy

MAVProxy is a fully-functioning GCS (group communication system) for UAV's. The intent is for a minimalist, portable and extendable GCS for any UAV supporting the MAVLink protocol. It has a number of key features, including the ability to forward messages from UAV's over the network via UDP (User Datagram Protocol) to multiple other ground station software on other devices.

- It is a command-line, console based app. There are plugins included in MAVProxy to provide a basic GUI.
- It is written in 100
- It is open source.
- It's portable; it should run on any POSIX OS with python, pyserial, and function calls, which means Linux, OS X, Windows, and others.
- It supports loadable modules, and has modules to support console/s, moving maps, joysticks, antenna trackers, etc...

3.5 JdeRobot

JdeRobot [http://jderobot.org/] is a software development suite for robotics and computer vision applications. These domains include sensors (for instance, cameras), actuators, and intelligent software in between. It has been designed to help in programming such intelligent software. It is mostly written in C++ language and provides a distributed component-based programming environment where the application program is made up of a collection of several concurrent asynchronous components. Components may run in different computers and they are connected using ICE communication middleware. Components may be written in C++, Python, Java... and all of them interoperate through explicit ICE interfaces.

JdeRobot simplifies the access to hardware devices from the control program. Getting sensor measurements is as simple as calling a local function, and ordering motor commands as easy as calling another local function. The platform attaches those calls to

remote invocations on the components connected to the sensor or the actuator devices. They can be connected to real sensors and actuators or simulated ones, both locally or remotely using the network. Those functions build the API for the Hardware Abstraction Layer. The robotic application get the sensor readings and order the actuator commands using that API to unfold its behaviour.

Several driver components have been developed to support different physical sensors, actuators and simulators. Currently supported robots and devices:

- RGBD sensors: Kinect and Kinect2 from Microsoft, Asus Xtion.
- Wheeled robots: Kobuki (TurtleBot) from Yujin Robot and Pioneer from MobileRobotics Inc.
- ArDrone quadrotor from Parrot.
- Gazebo simulator.
- Firewire cameras, USB cameras, video files (mpeg, avi...), IP cameras (like Axis).
- Pantilt unit PTU-D46 from Directed Perception Inc.
- Laser Scanners: LMS from SICK and URG from Hokuyo.
- Nao humanoid from Aldebaran.
- EVI PTZ camera from Sony.
- Wiimote.
- X10 home automation devices.

JdeRobot includes several robot programming tools and libraries. First, viewers and teleoperators for several robots, its sensors and motors. Second, a camera calibration component and a tuning tool for colour filters. Third, VisualHFSM tool for programming robot behaviour using hierarchical Finite State Machines. And several more. In addition, it also provides a library to develop fuzzy controllers, a library for projective geometry and computer vision processing.

Each component may have its own independent Graphical User Interface or none at all. Currently, GTK and Qt libraries are supported, and several examples of OpenGL for 3D graphics with both libraries are included.

JdeRobot is open-source software, licensed as GPL and LGPL. It also uses third-party software like Gazebo simulator, ROS, OpenGL, GTK, Qt, Player, Stage, GSL, OpenCV, PCL, Eigen, Ogre

3.6 External Libraries

3.6.1 ICE

ICE (Internet Communications Engine) [https://zeroc.com/], is an object-oriented middleware that provides remote procedure calls, grid computing and client / server functionality developed by ZeroC 10 under a dual GNU GPL and a proprietary license. It is available for C ++, Java, .Net languages, Objective-C, Python, PHP and Ruby, in most operating systems. There is also a version for mobile phones called Ice-e. ICE allows to develop distributed applications with minimal effort, abstracting the programmer to interact with low network interfaces. The process of application development should focus only on the logic and not in the peculiarities of the network. It is a multilanguage middleware platform and thus, we can implement clients and servers in different programming languages and on different platforms. ICE works with distributed objects, so that two objects in our application need not be running on the same machine. Objects can be on different machines and communicate across network through the sending of messages between them.

JdeRobot uses ICE for communication between its nodes, therefore, the task of reading values from a sensor or command orders to a robot is as simple as running a method of an object in the application. A significant advantage is the possibility to develop applications independent from context. A programmer can develop a driver in C ++ for a particular robot that is embedded in the robot, on the other hand itself, another developer can develop an application for image processing in Python that runs on a PC. Through ICE we can use these two applications, which originally were independent, as a

single application without having to worry about low-level communications. With this we can develop modular applications of greater complexity without additional effort.

3.6.2 OpenCV

OpenCV (Open Source Computer Vision Library) [http://opencv.willowgarage.com/wiki/] is a library of artificial vision developed by Intel, and now maintained by Willow Garage [https://www.willowgarage.com/]. It is under a BSD license, which allows free use for commercial purposes. This has meant that is commonly used for all kinds of projects, from surveillance systems with motion detection. It is written in C ++, multiplatform and contains interfaces for C, C ++, Java, Python and MATLAB. In 2010 an interface for GPUs based on CUDA and OpenCL in 2012 for developed. Having been developed by Intel it provides system integrated performance primitives Intel, which are a set of routines under specific level for Intel (IPP) processors.

3.6.3 OpenSSH

OpenSSH [http://www.openssh.com/] is the premier connectivity tool for remote login with the SSH protocol. It encrypts all traffic to eliminate eavesdropping, connection hijacking, and other attacks. In addition, OpenSSH provides a large suite of secure tunneling capabilities, several authentication methods, and sophisticated configuration options.

The OpenSSH suite consists of the following tools:

- Remote operations are done using ssh, scp, and sftp.
- Key management with ssh-add, ssh-keysign, ssh-keyscan, and ssh-keygen.
- The service side consists of sshd, sftp-server, and ssh-agent.

OpenSSH is developed by a few developers of the OpenBSD Project and made available under a BSD-style license.

Chapter 4

Hardware Platform

In this chapter the systems and subsystems that compose the vehicle platform are going to be described. First, the conception phase, design and integration process; Secondly, the description and technical specifications of the components and of the complete vehicle.

4.1 Conception

Based on the objectives established, the aerial vehicle type choice is the first to be done. A quadcopter configuration is the most effective configuration for this purposes, due to its manoeuvrability, ability to hover, simplicity compared with other multirotors or helicopters and market material offer.

The vehicle we would like to design and built from scratch, should try to achieve as much as possible the following features:

- Medium size, between 2-5kg weight and ¡1.5m size
- Open source software
- Reliable
- Several communication channels
- Outdoors usable

- Modular design
- Function extendable
- Cost efficient

4.2 Design

In the design of a quadcopter, or any multirotor vehicle, is crucial the correct choice between propeller design (size and pitch), electric motor (size and power) and power source characteristics based mainly on the vehicle weight and desired vehicle response. These parameters define the effectiveness and efficiency of the vehicle that is directly reflected on manoeuvrability and endurance of the vehicle.

Many facts take part in the vehicle design and there exist not a unique way to achieve similar vehicle characteristics, only trial-failure method combined with the experience produces successful results. In engineering, and more in this aspect, theory and reality are parallel lines, not convergent ones. Generalising, next table shows the variable of the configuration to be decided and its estimated effects:

TABLA

After several sketches, the final vehicle design is:

- Frame: Tarot Ironman 650 3M CFRP
- Flight controller: Pixhawk
- On-board computer: Intel Compute Stick STCK1 A8 LFC
- Camera: ELP 1080 ov2710 36mm lens
- GPS: U-blox LEAS-6H with compass
- Telemetry module: 3DRobotics 433MHz
- RC channel: FrSky Taranis - X8R-SBus
- Motors: T-motor MT2814 710KV 440W antigravity series
- ESC regulators: T-motor S35A opto 600Hz SimonK

- Propellers: GenFan 14x4.4 Carbon-Nylon / Quanum 13x4.4 CFRP
- Battery: Zippy 3S 5800mAh 30C / Tattu 4S 6600mAh 35C

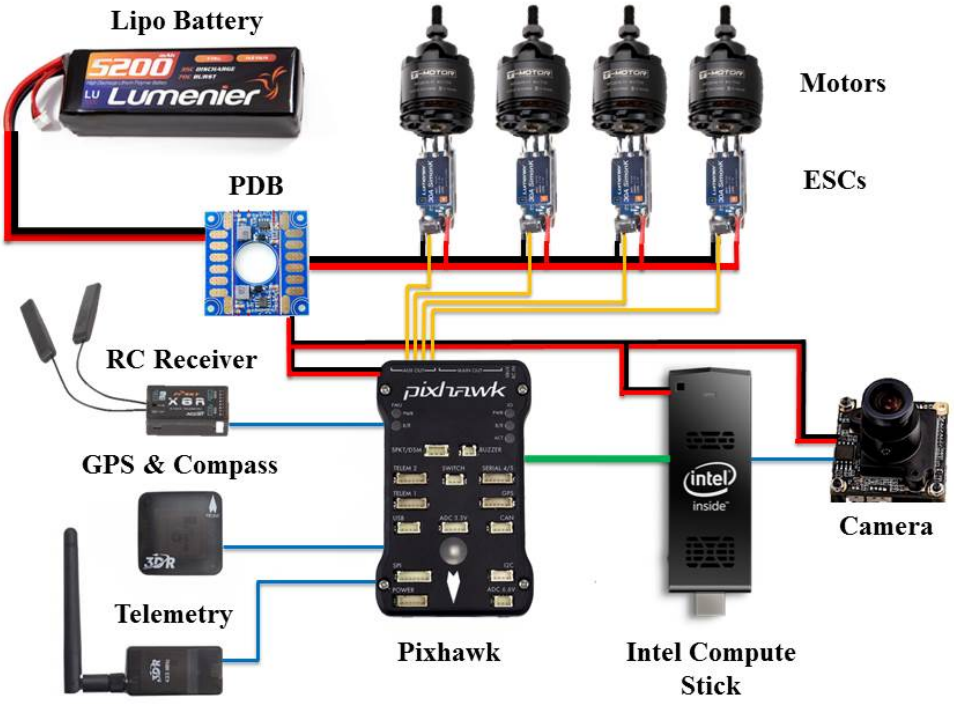


FIGURE 4.1: Hardware Scheme

Tarot Ironman 650

The Tarot Iron Man 650 is built by Toray 3K carbon fiber woven cloth fiber board, with 3K hollow twill pure carbon fibre tube, a full CNC machining design with higher standards. All the carbon and CNC work means the full frame weighs only 476 grams. The Full folding design makes for a high portability design and is highly adjustable to meet the needs of most utilitarian applications.

Features:

- Fold-ability for transportation and storage
- Toray 3K carbon fibre
- Lightweight

- Folding landing gear
- 60mm rail mount

Specs:

- Size motor to motor: 650mm
- Weight: 476g
- Height from ground to lower rods: 180mm
- Height from ground to top: 220mm

Pixhawk 3Drobotics

Pixhawk is an advanced autopilot system designed by the PX4 open-hardware project and manufactured by 3D Robotics. It features advanced processor and sensor technology from ST Microelectronics® and a NuttX real-time operating system, delivering incredible performance, flexibility, and reliability for controlling any autonomous vehicle.

The benefits of the Pixhawk system include integrated multithreading, a Unix/Linux-like programming environment, completely new autopilot functions such as sophisticated scripting of missions and flight behaviour, and a custom PX4 driver layer ensuring tight timing across all processes. Pixhawk allows existing APM and PX4 operators to seamlessly transition to this system and lowers the barriers to entry for new users to participate in the exciting world of autonomous vehicles.

Features:

- Advanced 32 bit ARM Cortex® M4 Processor running NuttX RTOS
- 14 PWM/servo outputs (8 with failsafe and manual override, 6 auxiliary, high-power compatible)
- Abundant connectivity options for additional peripherals (UART, I2C, CAN)
- Integrated backup system for in-flight recovery and manual override with dedicated processor and stand-alone power supply

- Backup system integrates mixing, providing consistent autopilot and manual override mixing modes
- Redundant power supply inputs and automatic failover
- External safety button for easy motor activation
- Multicolor LED indicator High-power, multi-tone piezo audio indicator
- MicroSD card for long-time high-rate logging

Specifications:

- Microprocessor:
  - 32-bit STM32F427 Cortex M4 core with FPU
  - 168 MHz/256 KB RAM/2 MB Flash
  - 32 bit STM32F103 failsafe co-processor
- Sensors:
  - ST Micro L3GD20 3-axis 16-bit gyroscope
  - ST Micro LSM303D 3-axis 14-bit accelerometer / magnetometer
  - Invensense MPU 6000 3-axis accelerometer/gyroscope
  - MEAS MS5611 barometer
- Interfaces:
  - 5x UART (serial ports), one high-power capable, 2x with HW flow control
  - 2x CAN
  - Spektrum DSM / DSM2 / DSM-X® Satellite compatible input up to DX8 (DX9 and above not supported)
  - Futaba S.BUS® compatible input and output
  - PPM sum signal
  - RSSI (PWM or voltage) input
  - I2C®
  - SPI

  - 3.3 and 6.6V ADC inputs
  - External micro USB port

- Power System:
  - Ideal diode controller with automatic failover
  - Servo rail high-power (7 V) and high-current ready
  - All peripheral outputs over-current protected, all inputs ESD protected

- Weight and Dimensions:
  - Weight: 38g
  - Width: 50mm
  - Thickness: 15.5mm
  - Length: 81.5mm

Intel Compute Stick STCK1 A8 LFC

Intel Compute Stick is a cost-efficient computer for portable uses, it delivers a Mini PC with full-size performance, reliability, and ease of use. Intel Compute Stick has all the performance needed for running thin client, embedded, collaboration, or cloud applications.

- Processor: Intel®Atom Processor Z3735F
- Graphics: Intel®HD graphics via HDMI
- Audio: Intel®HD audio via HDMI
- System memory: 1GB soldered, single channel, DDR3L memory
- Storage: 8GB eMMC
- Connectivity: Integrated 802.11 bgn wireless, Bluetooth
- 4.0, USB2.2, microSD slot
- Power requirements: 5V 2A DC
- Size: 103mm x 37mm x 12mm

T-motor MT2814 710KV 440W antigravity series

T-motor Brushless motor is a worldwide leading supplier of components and systems for the Aerial photograph ,some Industry and commercial application in the precision drive technology sector

Specifications:

- KV: 710
- Max Continuous Power: 440W
- Max Continuous current: 27A
- Max efficiency current(6-18A): ¿83
- Internal resistance: 125mΩ
- Configuration: 12N14P
- Stator Diameter: 28mm
- Stator Length: 14mm
- Shaft Diameter: 4mm
- Motor Dimensions: 35mm x 36mm
- Weight: 120g
- ldle current (10v): 0.4A
- N°of Cells (Lipo): 3S - 4S

EFFICIENCY TABLE!!!!

4.3 Integration

Integration process is matter of place, connect, secure the different components and make and smart use of the space available, not forgetting the weight importance. Even it seems that it is an easy task, in reality integration is a complex and long process in

which is necessary to deal with electronics, electromagnetics, chemistry, materials theory and soldering.

As a result, the final vehicle set up is reached and now the components works together as an unique and complex system, an UAV.

Vehicle specifications:

- Rotor axis distance: 650mm diagonal
- Vehicle span: 950mm diagonal
- TOW: 2000g
- MTOW: 3000g
- Flight time: 25min
- Range: undefined
- Maximum combined thrust: 6000g
- Maximum combined power: 1760W
- Lipo Battery supported: 3S - 4S

Sensors:

- GPS
- gyros
- accelerometers
- compass
- barometer
- camera

Communication channels:

- Radio control: 2.4GHz

- Wifi
- Bluethooth
- Telemetry: 433MHz MAVLink Protocol based.

4.4 Configuration and testing

To configure the Pixhawk firmware and update it, APM mission planner software have been used. This program offers a lot of tools that help the user to access to the software without fighting with the code lines. With this program some configuration was performed: update firmware, calibrate some sensors (compass, gyros and accelerometers), calibrate the transmitter and establish the different flight modes on the transmitter.

Prior to the vehicle first flight, it is necessary to make some tests on the ground, verifying the correct work of the motors and the radiolink for safety reasons. Then, the flight test were performed:

- Fully manual: the main objective is to verify all the components are connected and working correctly, almost without software intervention.
- Autonomous stabilisation and landing: in this case, we verify the correct working and the positioning sensors (gyros, accelerometers, compass, barometer and GPS) and testing the low level software performance.
- Simple waypoint mission: verification of the correct use of the point to point navigation that is previously configured and correct GPS and IMU data fusion.

Once the flight tests of the vehicle platform were performed, the quadcopter is ready to start the development, the next objective for the system was to be connected with JdeRobot.

Chapter 5

Software Platform: Driver

This chapter describes the software component developed for the project, it is responsible access to the sensors and actuators of the vehicle based on MAVLink protocol communication messages, and translate it into JdeRobot interfaces.

5.1 MAVLinkServer

MAVLinkServer is a driver based on MAVProxy and developed to act as a translator middleware, it is design as a JdeRobot driver in Python language.

MAVLinkServer relies in the MAVProxy parser, the part of the program that is encharge of the management of the MAVLink messages. It establish connection with Pixhawk autopilot, maintains communication channel operative, acquired and interprets messages and creates and send new ones with the information proportioned by the application.

The developed code is mainly responsible of the management of the JdeRobot interfaces. It is able to handle the information given by the MAVProxy side, it regulates the creation and modification of the classes where information is temporally stored and opens ICE communication channels to make the component usable for JdeRobot.

These two sides of the component provide a reliable and muli-compatible driver thanks to the ICE characteristics; MAVLinkServer would connect with application written in other different languages through JdeRobot interfaces.

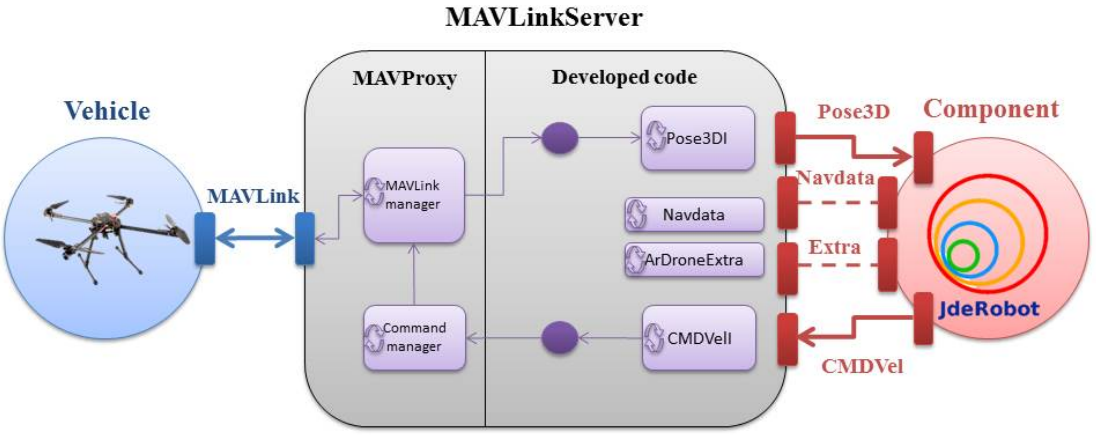


FIGURE 5.1: server Scheme

The most used JdeRobot interfaced in this project have been Pose3D and CMDVel. Note that Pose3D make use of quaternions instead of Euler angles, this is done to avoid singularity and computationally less intense compared to other attitude parameters such as Euler angles or a direction cosine matrix.

Pose3D

```
Pose3DData{
    float x;  /* x coord */
    float y;  /* y coord */
    float z;  /* z coord */
    float h;  /* */
    float q0; /* qw */
    float q1; /* qx */
    float q2; /* qy */
    float q3; /* qz */
    };
```

CMDVel

```
class CMDVelData{
    float linearX;
```

```
float linearY;
float linearZ;
float angularX;
float angularY;
float angularZ;
};
```

5.2 Driver code explanation

First step is to define the corresponding interfaces, it have be explain previously, MAVLinkServer supply all the JdeRobot interfaces for the use of any external component. Here it is an example of Pose3D interface.

```
<class POSE3D VERBATIN>
```

This class inherits "jderobot.Pose3D" and corresponding functions are defined. It can be notice the use of programming "locks" for protecting the data stored in the class. This is because MAVLinkServer is constantly refreshing the information provided by the sensors and also constantly publishing it through ICE; this could cause writing or reading errors and interfere in the component performance. With the use of the locks, the information could not be read while other task is writing on it and the other way around, ensuring the correct manage of the information.

CMDVel interface has the same structure as Pose3D, with the use of locks also.

```
<class CMDVel VERBATIN>
```

MAVLinkServer launch several threats to communication channel, one for each. Even Pose3D and CMDVel are the ones to be use, the driver offer the remaining interfaces (NavData and ArDroneExtra) for compatibility and future uses.

```
<interface threats VERBATIN>
```

Each threat make use of its own function were the ICE configuration is performed. This is an sensitive step, where ICE publication is done and it is important to ensure which data is send, in order for other applications to get the it correctly and ensure compatibility. The same procedure is performed for all the interfaces. The following code represents the CMDVel function as an example.

```
<open channel function VERBATIN>
```

MAVproxy is constantly handling MAVLink messages in a "while true" loop, this take advantage of it and complement its task with the refresh of the sensor information required. As a high level driver, this program make use of treated information of the vehicle, do not interfere the data fusion performed by Pixhawk and it trusts on its, more than tested, reliability.

```
<main loop VERBATIN>
```

For the vehicle command delivery other threat is launched, as the ICE channel ones, which its corresponding associated functions. It make use of the MAVProxy order management, in this case, with the command "velocity(x y z)".

```
<sendCMDVel2vehicle VERBATIN>
```

Finally, Pixhawk make use of a NED (North East Down) reference frame, that in aeronautics is call "local horizon" and applications would take command decisions on the body reference frame (front, right, down). A change of reference frames is performed with a function, using Tait-Bryan convention as in aeronautics world (zyx order). Also a function to change from GPS coordinates (lat, long, alt) to global Cartesian coordinates (x,y,z) is been done.

```
<small functions VERBATIN>
```

As mentioned before, this project has made use of quaternions for its advantages, however there is not a popular or well tested library for the use of quaternions, so some simple functions has been defined for the correct usage of this system.

```
<quaternion functions VERBATIN>
```

5.3 Driver operation

MAVLinkServer is a multy-threat component, the flow of the information and the operation if the driver pursues the following tasks path:

- Driver starts running all the different threats, opening its communication channel and defining the information that would be in each other.
- MAVLink Messages from Pixhawk came into the driver, and they are been interpreted in order to acquire the position and attitude information.
- Acquired information is treated to transform it in JdeRobot standards (Pose3D). Latitude and longitude are transformed into global xy coordinates, using WGS84 as earth model, and Euler angles attitude is transformed into quaternions system.
- Pose3D is written in the corresponding local classes in a controlled way, making use of the lock system.
- Classes are publish in the corresponding ICE channels as the threats are running, to other components to access to it.
- Commands are received through CMDVel channel, ones external decision has publish into the ICE channel.
- Information is extracted from the CMDVel class with the mentioned lock system.
- Commands are treated and a reference frame change is performed in order to be synchronised with Pixhawk. Body referenced commands are transformed into NED coordinates system making use of the quaternion functions designed.
- Commands are delivered to MAVproxy command manager.
- Commands and translated into a MAVLink message and sent to the vehicle.

It can be seen that each thread has a task and each one does not have the same workload, for this reason the times of the control loops of each wire are different. However the threats are working and the driver and even they has different rhythms, the component is successfully working in all its different tasks.

Chapter 6

Software application

Once presented in the previous chapter the software infrastructure to access the sensors and actuators of the UAV, in this chapter the developed application that implements autonomous predefined navigation and visual navigation techniques will be described. The visual control is a control system in which the feedback obtained from a video camera.

6.1 MAVLinkClient

MAVLinkClient is a JdeRobot component completely developed by this project using Python as programming language. The objective of the component is to generate velocity commands (CMDVel) based on the processing of the information available, that is, produce its own decisions and perform and autonomous task.

This application aims to perform a target search mission around a designated localisation; ones target has been spotted, loiter over its position and send an alarm to the base. This mission use the information supplied by the vehicle attitude sensors and the on-board camera. One possible direct real implementation of this mission would be a "man overboard" search in the sea.

MAVLinkClient has four different functions: First is to design and manage the mission and establish trajectory, second is to calculate the velocity commands to try to follow the trajectory or mission decision, third is to analyse the images and identify the target, and fourth loiter over the target.

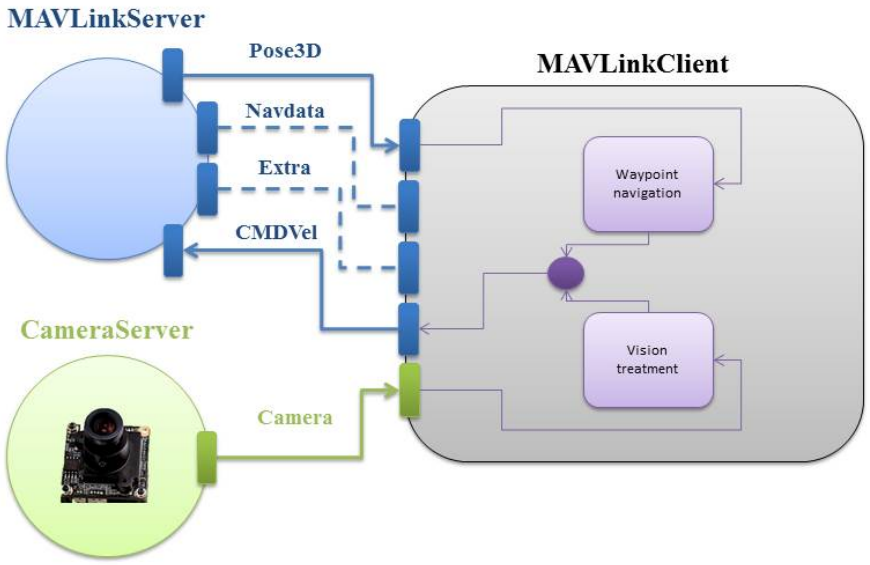


FIGURE 6.1: Client Scheme

6.2 Mission and trajectory

As the mission has been established a search one, an area scan is necessary to be perform. The scan method choose in this project is the spiral, in which the vehicle describes a squared spiral trajectory starting in the middle and covering further distance each revolution. The spiral is defined by four values:

- Centre point: (x,y) in global coordinates in metres
- Mission height in metres.
- Scan distance: distance between parallel sides of the spiral in metres.
- Number of spins: number of revolution of the spiral before restarting trajectory.

The next function design the trajectory based on the information given and returns a list of waypoints of the trajectory, in order to the rest of the application to use and modify the trajectory:

```
<spiralTrajectory VERBATIN>
```

This method is based on a "last-position-known" probabilistic method which is based on the concept that a target is more probable to be found around the last position known, if an established trajectory or movement is not recognised.

If during the trajectory follow mode, the target is found, the vehicle change its mission and tries to loiter over it; using the camera and vision treatment to identify the relative position between the vehicle and the target.

6.3 Trajectory Follow Commands

The trajectory follow mode is based on point-to-point navigation, in which a set of waypoints are determined and the vehicle try to catch up them with an order. This part uses the current position and attitude of the vehicle (Pose3D) and calculates the relative position of the waypoint and the vehicle.

The difference vector between the position and the waypoint determine the path to follow, so the commands are set in CMDVel and send via ICE. A waypoint is consider caught up when the distance between vehicle and waypoint is less than a pre-determined distance ("ReachedDist").

```
<nextWaypoint VERBATIN>
```

6.4 Target Identification: Object Tracking

The detection of an object within an image can be done in multiple ways. This project has chosen to identify objects from its colour, his attribute detection has some disadvantages compared to other detection methods. The biggest drawback is the light of the environment where the robot is and the shadows that can project other objects to the target. The images obtained from the camera are encoded in RGB, this colour space is not very robust to changes in ambient light. For this reason it was decided to change the RGB colour space to HSV, which although does not end solve the problem of light,

is robust enough for detecting an object in an environment of changing light, as long as the changes light are not very sharp.

The pipeline of the perception system is computationally simple and lightweight. The following sections will be explained in more detail each of the stages.

As a result, the component obtain a positive or negative identification of the object, and in the case, the horizontal relative position between the vehicle and the target.

6.4.1 RGB to HSV

If a colour image is about to be treated, the most common encoded values are in the RGB colour space (Red, Green, Blue) that determines the colour composition in terms of the intensity of the primary colours of light. The RGB model is commonly used to display colours on screens, TVs, etc., but has a big disadvantage when trying to identify a colour within a real image. The RGB colour space is not very robust against light changes. In the field of robotics, where robots are in very changing environments it is convenient to use another colour space more robust to changes in light, for example the HSV space.

HSV (Hue, Saturation, Value) is a nonlinear transformation of RGB model in cylindrical coordinates, thus, the colour is defined by:

- Hue: is the angle between 0 and 360 or representing colour tone.
- Saturation: is the offset of black-white brightness values ranging from 0 to 100.
- Value: is the height in the white-black axis has values ranging from 0 to 100.

To changes in brightness changes are usually reflected in V, leaving the S and H more or less unchanged. Due to the possibility to define a colour based on a dye and a saturation, the HSV model is an excellent candidate for robust identification of objects through their colour in an image within a changing environment.

Once the RGB image is obtained, to change the colour space to HSV the following algorithm is been used: Been MAX the maximum value of the components (R, G, B) and MIN the minimum one.

<rgb2hsv VERBATIN> or image

6.4.2 Image Smoothing

Within the processing stage image, there are some techniques that improve some aspect of the image, as smoothing technique. Noise, signal degradation, errors in the recruitment process or image transmission or lack of uniform illumination are image characteristics that can interfere the object analysis. The perception system component implements smoothing technique Gaussian Blur, which is the result of convoluting an image with the Gaussian function:

<equation> or image

Where x is the distance from the origin of the X axis, and is the distance from the origin of the Y axis and σ is the standard deviation of a Gaussian distribution. This distribution values are used to construct a convolution matrix that is applied to the original image. The central pixel value is given greater weight (higher value Gauss) and adjacent pixels receive a smaller weight depending on the distance from the original pixel increase. The result of this process is a blur retaining borders and edges of the object. This task is performed by the function of OpenCV GassianBlur().

6.4.3 Colour Filtering

The choice of colour is an important aspect object detection, a vivid colour will be easily detectable, colour with more muted tones will be more difficult to detect. The object tracking application allows us to select the colours that we want to identify the object, defining their HSV values. By applying thresholds to a colour image is transformed into a binary image, where objects of interest stand out with a different background pixel value. In the project a fixed threshold is used, the resulting image B is defined from the original image. If the pixel value passes the threshold, the resulting pixel will be white, while if it fails, it will be black. The result is the binary image. inRange() function, belonging to OpenCV, apply thresholds to the image.

6.4.4 Segmentation

The purpose of object detection is to obtain the centre coordinates of the object in the image. Thus we can distinguish whether the object has moved in different iterations of the algorithm and distinguish where. To accomplish this task the system of perception component performs a series of steps:

- Once the binary image obtained with the desired colour pixel detection, segmentation technique based on edge detection is used. The idea of this method is that if the borders of objects, with the background content within these boundaries may target objects.
- With the outlines of objects located, the next step is to approach these polygons. From a list of points that determine the outline of an object, the approxPolyDP () function OpenCV, which implements the algorithm Ramer-Douglas-Peucker, is able to return another list of points with a small number of points. This algorithm is based on finding critical points from a linear tolerance. The critical points, which constitute the simplified line, which gradually will be reaching a distance.
- Knowing some point in the rectangle on the image, it can find the centre of the box following a simple formula. The boundingRect () function OpenCV, returns a list of rectangles from a list of points. This function attempts to create the minimum rectangle that contains within it all the points indicated as a parameter to the function.
- After calculating the coordinates of the centre of the rectangle, they will be in the reference system of the image itself, so you have to make a change to the reference system used in object tracking, centred image.

6.4.5 Low-Pass Filter

Due to the ambient light conditions, the proper motion of the drone or the object we are detecting would leave the image, the rectangle approximated by the perception system fluctuates. These fluctuations cause the estimated coordinates of the centre of the object vary and in many cases these changes do not mean that the object is actually moved. For example, a change in ambient light cause the sensing system detects an object

different size different from the size of the same object previously detected. To combat against this behaviour and the application has implemented a temporary low pass filter at the output of calculating the coordinates of the detected object. This filter takes into account the previously captured image (t-1) and the current image (t). It is to give a percentage of credibility to the current calculated images, relying also on the above observations.

Where P (t-1) estimated the centre of the object of the above observation and O the centre of the object of the current observation and applying this filter can estimate the final coordinates of the object (P) using the formula. In object tracking the used filter is conservative, since it relies 60% on the previous observation. Applying this low-pass filter temporary smoothing is achieved in estimates, making not change abruptly.

6.5 Loitering Control

In this control only aspect to take into account is to position the drone above the colour that marks the position of the object on the ground. Only two dimension are controlled, the X axis and Y axis; height control is done by the system itself, with the established altitude.

Loitering control will take as reference the image the central point (0.0) and the difference of it with the point P (t) estimated by the perceptual system will give us the error of the position of the drone relative to the target in both X and Y . These errors will feed actuation system which we have based on three proportional–integral–derivative controller (PID) controllers, one for each axis.

A PID controller is a control loop feedback mechanism commonly used is UAV's. The controller attempts to minimise the error over time by adjustment of a control variable, in this case the monition through CMDVel commands. PID make use of three different treats of the error to make corrections:

- Proportional: produces an output value that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant Kp, called the proportional gain constant.

- Derivative: is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain Kd. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, Kd

- Integral: is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain, Ki, and added to the controller output.

The output of the PID controller is translate into CMDVel commands and send to the driver in order to minimise the error and centre the target in the image. It is important to notice that as this code is calculation the relative position between the vehicle and the objective, the resulting CMDVel is based on body reference frame.

When the system internally Ar.Drone board attempts to stabilise the drone in the air, it causes the vehicle rotates, producing a movement of the image along the X and Y axes This slight drift influences the detection object, causing the centre of the rectangle containing the colour fluctuate over time, without thereby vehicle has moved excessively. This behaviour is intrinsic to the multirotor system, so object tracking recognise this movements as noise. Dead bands have been created along the X and Y axes, if the distance error in the X axis is within the dead band, the control does not command any action on that axis.

6.6 Client operation

Chapter 7

Conclusions and Next Steps

In this chapter, a critical analysis of the outcomes development in the previous chapters is done to obtain conclusions about the vehicle design and construction and the development of the driver and the application. Finally, a rating of the future works or next steps that can take this Project as a benchmark will be done.

7.1 Conclusions

7.2 Next Steps

Bibliography