POLITECNICO DI TORINO

MASTER'S DEGREE IN MECHATRONIC ENGINEERING

Master Thesis

Implementation of an UAV autonomous mission planning

aimed at collecting geo-tagged images from a multispectral camera

Relatori
Marcello Chiaberge
Luigi Mazzara

Candidate
Donia Afifi
matricola: 279988

April 2022

Abstract

During the last decade, the amount of research that involves unmanned aerial vehicle (UAV) has spread out by offering great number of challenges for artificial intelligence and knowledge representation. Autonomous vehicles have gained a huge relevance in different fields and they have been proposed for a wide range of indoor and outdoor applications, including searching, surveillance, infrastructure inspections and serial aerial image acquisition. The most common mission scenario involves placing on the UAV different sensors like GPS, camera, telemetry for data collection tasks where the data could be processed using off-line or real-time applications supervised by a ground station. The purpose of the thesis gets inspiration from the Spectral Evidence of Ice (SEI) project, aimed at providing innovative tools to inspect and identify the ice on airplanes, using techniques of photogrammetry and spectral analysis, to support operators in improving the efficiency of the deicing process. The autonomous capability of UAV structure can be useful to facilitate this procedure thanks to the usage of a multispectral camera. To achieve this goal, the developed topics are navigation, data analysis, communication protocol and devices interface. ROS (Robot Operating System) plays a key role in the development of the system helping to build the robot application and enabling the communication between the pc companion, meant to be the control unit for mission planning and management, and the autopilot, intended to be the flight manager of the UAV. Consequently, the goal of this study is testing the drone maneuverability, the application program interface and providing a collision free trajectory to reach the desired destination. Localizing, detecting obstacles, direction correcting, and tracking the trajectories with reasonable accuracy are features examined in the path planning problem. Therefore, it is crucial to further extend path planning algorithms and methods for autonomous drones. The algorithms used to develop the path belong to the graph search algorithm which, given a fixed point, generates a circular trajectory around a specific object placed within the test environment. Another accomplished task is the integration of a multispectral camera in the architecture system in order to take geo-tagged images each time the drone reaches a desired waypoint for inspecting and reconstructing purposes. The collected images are involved in several tasks. In the first phase they are exploited to get information about aircraft's location in the apron. They are used to feed an already developed convolutional neural network (DeepWay approach) to create a dataset of segmented pictures. The information extrapolated from those images provides an occupancy grid map of the analyzed sector in order to achieve a new list of waypoints which optimize the path planning. So, in the second phase, the data obtained by the CNN allowed to navigate closer to the aircraft and have a better view and prospective to detect the ice through the multispectral acquisitions. Finally, all this work shall be

tried out on a real case to validate and compare the results gained in the simulation environment.

Acknowledgements

Here you are, the conclusion of this long years spent fully of sacrifice, hard work, laugh and tear. I am grateful to all the people that I meet thanks to the University and study room. Many of them are my second family and they always support me despite my crisis, and strange personality. We have shared a lots of difficulties that allow us to grow together. I would like to thank all the people of that helped to understand and learn a different item, in particular the ones that follow me during the develop my first project with service robot.
I sincerely express my gratitude to all my family that believe in me and a big hug goes to my ladies that always motivate me, and believing in my potentiality more than me.

Contents

List of Figures

Part I

First Part

Chapter 1

Introduction

1.1 Background

In the last century the amount of resource invested in the robotics field had a fast enhancement, indeed it is becoming a well-establish and consolidated reality and these technologies are widely used in all modern companies. The term service robotic defines a new class of intelligent robots able to operate in working and domestic contexts with the aim of increasing the life quality of the human being by avoiding exposure to risk factors and facilitating daily operations. The drone manufactured these days are becoming smarter by integrating open source technology, smart sensors and more flight time. Technological advances such as navigation, dynamics, sensors, and communications make it possible to use them for different applications—not only military but also civilian and commercial uses, such as product delivery, agriculture, serial photography, surveillance or environmental monitoring. The increase in the use of these vehicles for various applications has made the scientific community not only focus on their aerodynamics or the materials or chipsets of which they are composed, but a new challenge arises that is of greater interest - the design of collision-free path/trajectory planning mechanisms, especially in future crowded airspaces [1]. The main interesting characteristic of this type of vehicle is that they are able to fly and carry out some operation without a pilot on board. UAV structures can be driven either by on board systems called autopilot, or they can be controlled by a human operator using a remote controller. The advantage in using this type of technology lie in the (comparatively) low production and operating costs and the flexibility to adapt the aircraft to the specific demands of its mission [2]. More UAVs equipped with sensors, such as cameras, sonars and lidars, are launchd on the civilmarket. It turns out that this technology can performvery well in environmental studies such as retrieving information about some certain area for learning the species [3].

Figure 1.1: Example of UAV's application

1.2 Thesis goal

The ice detecting project has been started up to supporting the operators during the pre-flight inspection of aircraft, in particular in the stage that require the searching and localization of ice stored on the critical areas. The presence of ice on the fuselage, motor's area, wings, tail surfaces and antennas lead the aircraft's balance in a loss of control, so ice investigation and de-icing are necessary treatments for flight safety particularly during winter season. The accumulation of ice occurs when aircraft is flying through visible water such as rain or cloud dribble or may occurred on parked aircraft due to precipitation and atmospheric conditions. The best strategy for ice detection is to limit de-icing fluid used by improving the technology and management of the crew's operations. Thus this is research also has an environment impact, indeed the collected information from the aircraft will be used to reduce the amount of fuel wasted, the quantity of de-icing fluid and Co2 emitted.

The main developed aspect consists of classifying ice through spectral images collecting by multispectral camera and experience different navigation algorithms needed to inspect the aircraft parking in the apron. Trajectory generation has different drawback like limited time and onboard computing resources indeed, there is not any work that guarantee to generate safe and kinodynamic feasible trajectories

at high success rate. there are two keys element belong to trajectory generation: the efficiency and robustness. computing online trajectory require constantly computation in a short time.

The purpose of the project is to design an autonomous UAV structure able to perform an offline mission, testing the most suitable sensor using a radiometric approach .The autonomous flying capabilities and the development of a predefined path of the are aimed to capture and localize the presence of ice on the aircraft inside the apron using a geo-tagged image. The images will be taken by a multispetral camera mounted on the UAV making use of a gimbal support to stabilize and correct the vehicle's orientation along the RPY axis . ROS has been used to run code on multiple computers to prove tools and libraries. UAV agent was tested in a simulation environment, GAZEBO which has been developed using agent-based modelling. By simulating a dynamic environment, the capabilities of a UAV agent can be tested under defined conditions and additionally its behavior can be visualized. Then later experimental tests have been performed replacing all the drivers like PX4 with a real system. In the following chapters the items that will be analyzed in details to develop the described work, can be summarize as follow:

1. UAV structure able to perform a predefined path that has been programmed,
2. collection of data using camera as driver,
3. communication system used to integrate the drone, drivers and the ground station,
4. test of algorithm and drives.

1.3 Related work

The use of UAV's to acquire images for purposes of photogrammetry and, in general, three-dimensional reconstruction from images, represents a widely spreading application field. Among the main reasons for this wide diffusion, there are the low cost and extreme versatility of modern micro-UAVs, able to fly along precise trajectories and to keep a fixed position steadily, as well as the simultaneous decreasing cost and increasing resolution of image sensors.

We can identify different state-of-the-art works [4], [5] in which a camera is mounted on a drone and mission is performed autonomously, exploiting the images acquired on flight both for navigation and for 3D reconstruction. However, to achieve such powerful tasks in real time, these systems need the support of remote high-performance computing facilities, connected with the drone through a high-bandwidth, low-latency wireless link, and running remotely the path planning and 3D reconstruction algorithms on the image stream acquired by the drone. Controlling the flight and its camera for 3D reconstruction of large object [9] reported one of

the first successful applications of mini and micro-UAVs combined with low-weight and low-cost multi-spectral sensors which carries out an autonomous acquisition of the proper image sequence for 3D reconstruction, while processing the acquired video in real time for self-localization and consequent navigation, according to the mission control.

Chapter 2

UAV anatomy

2.1 Overview of the main components

All the robotic applications that exploited in general public used different software inside and much of their intelligence and value depend on it [10]. UAV structure is defined as an aerial vehicle that does not transport a human because it flies autonomously or piloted remotely carrying a lethal or nonlethal payload. It is controlled either by on-board computers or by remote control under the supervisor of a human on the ground [12]. To build a dynamic unmanned aerial vehicle we need to attach many electronic devices. It will be used brushless DC motor, ESC, digital servo motor and telemetry system for real time communication with drone.

In the following section it will be summarize the main components used to build the UAV's system used in this work:

1. Frame is the supporting component of the whole aircraft, to guarantee stiffness, stability, and precise balance of the weight.

2. Rotors with propellers, positioned in the same plane parallel to the ground, are the most important components of the UAV because they are responsible of the thrust generation. It is necessary to be sized to generate a thrust that guarantee good stability and response of the vehicle.

3. Battery is the energy source of the whole structure. To choose battery's dimension is necessary to make a trade-off between capacity, that influence the performance in terms of TOF, and physical dimension of the device.

4. Flight controller is able to measure the movements of the drone using different sensors like accelerometer, barometric pressure sensor, IMU and gyroscope. The collected data are elaborated by the firmware using algorithms which compute the velocity of the motor.

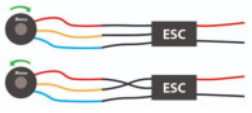
5. Electronic speed controller, ESC, is necessary to actuate motors and take care of the speed and direction regulation by manipulating voltage that is applied to it. The even number of rotors are equally spaced typically arranged at the corners of a central body, attached to the propellers.

(a) Frame

(b) Propellers

(c) ESC

(d) Brushless motor



(e) Connection between motor and ESC

UAVs can be equipped with serials of devices and sensors which allowing them to be employed on very different tasks. They are chosen related to the mission.The role of sensors are ensuring that unmanned systems work properly in order to meet the requirements of the target application, provide and increase their navigation capabilities, and suitably monitor and gain information on several physical quantities in the environment around them. Below, we introduce the main used sensors to achieve this project :

1. Camera with high resolution and gimbal control system.
2. Antenna
3. Telemetry module is a technology that allows data' s transmission from remote or inaccessible points to control station.

The following sensors belong to the Inertial Measurement Unit (IMU):

1. RTK GPS used to transmit a radio signal from satellite, it's possible to monitor the position and speed of aircraft.
2. Accelerometer, barometer and magetometer

Their goal is to control the UAVs during take-off and landing actions, in order to use only IMU data without examining external data.

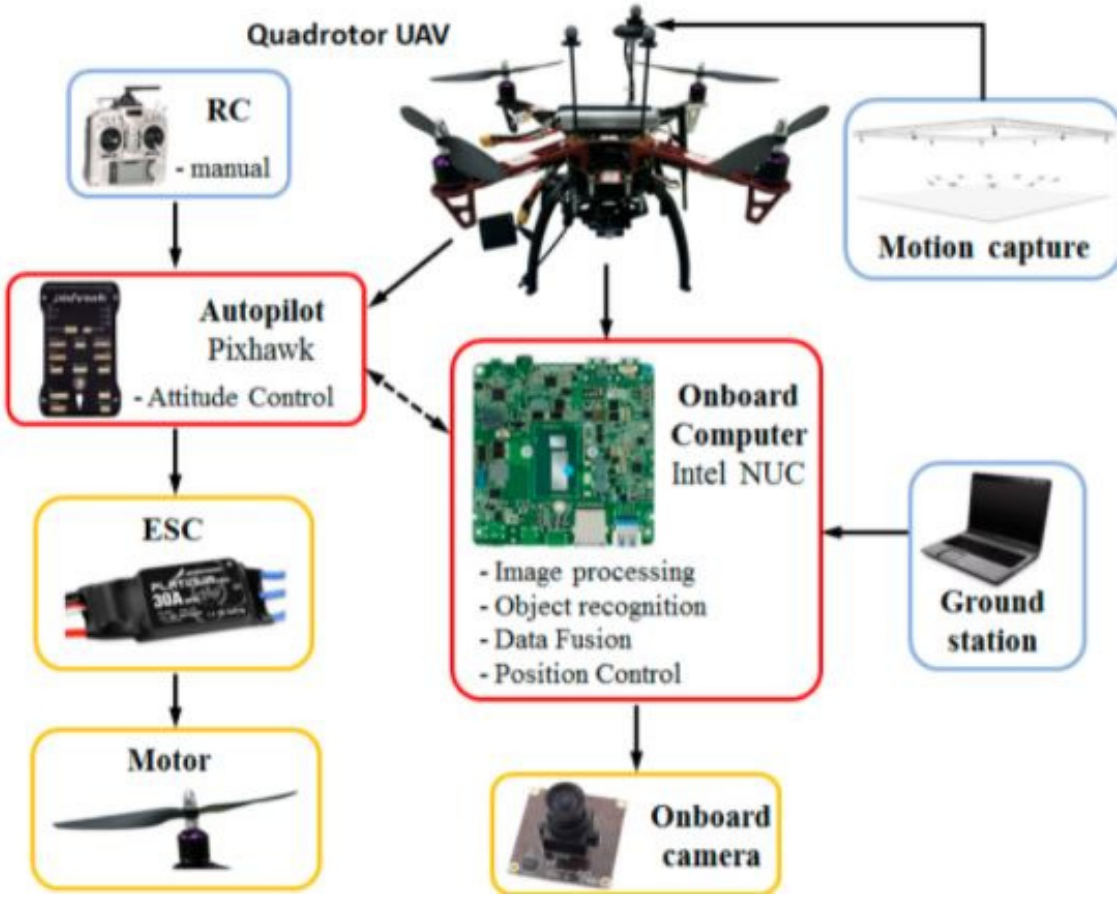


Figure 2.2: Full drone's architecture

2.2 System description

Any vehicle that shall perform operations autonomously, given an initial setup provided by the operator, should have three parts that cooperate with each other: firmware, software and hardware. Autonomous drones in the last year gained a lot of interest both in university project and in industrial companies thanks to versatility of firmware that allows the integration of several hardware components adding new tasks that drone can perform or to enhance already existing ones.

2.2.1 Firmware

The term autopilot refers to a system able to perform missions autonomously without the need for manual remote control. These missions could have different aims like cargo delivery, mapping, surveillance, and many other applications. Generally, operators set the parameters of the mission using ground control stations. An autopilot drone then allows the operator to focus on other aspects of the missions such as telemetry control, data processing or video monitoring. Nevertheless, this type of system facilitates the operator's work but can't be completely replace it. Drone's firmware is the "heart" because it controls everything from flight inputs to battery management, so the first step is to choose the most suitable firmware compatible with the project intent to be realized to ensure safety and reliability. It also allows to integrate hardware components. The two most used flight management firmware of a multi-rotor are Ardupilot and PX4. Both can be used on most available flight controllers:

1. Ardupilot is one of the most used application in the field of autopilot firmware. It can be installed not only on aircraft such as drones or airplanes but also on land and marine vehicles like rovers, boats and submarines. Being very used Ardupilot is also one of the most tested and continuously evolving software (it is open-source code based). The software suite is installed in aircraft from many companies, such as 3DR, jDrones, PrecisionHawk, AgEagle and Kespry. It is also used for testing and development by several large institutions and corporations such as NASA, Intel and Insitu/Boeing.

2. PX4 is an open-source autopilot system oriented toward inexpensive autonomous aircraft. Low cost and availability enable hobbyist use in small remotely piloted aircraft. It is made of two main layers: the flight stack that is an estimation and flight control layer and the middleware. Ardupilot and PX4 fully comply the design constraints as they are both open platform and autopilot firmware and use the MAVLink communication protocol that can be managed by ROS through the MAVROS package.

2.2.2 Software

The interface to the controller is made by software. Q-Ground Control is the application used to configure and tune the vehicle for planning an autonomous mission through telemetry, camera and other devices mounted on the drone. It also allows to monitor drone while flying, including streaming video [14]. Advanced packages or add-ons allow autonomous mission planning, operation, and post-mission analysis. This is generally used by the operator both to program the mission and to analyse log files.

2.2.3 Hardware

It includes all the combination of sensors, controllers and output devises which allow the drone to sense the external environment taking decision on current scenario. It is an electronic circuit with the purpose to vary an electric motor's speed, its direction and possibly also to act as a dynamic brake. It converts DC battery power into 3-phase AC for driving brushless motors. There are brushless motors connected to the number of drone's ends which used to supervise the propulsion. The flight controller interprets input from receiver, GPS module, battery monitor, IMU and other onboard sensors. The flight controller is another component that regulates motor speeds, via ESCs, to provide steering, as well as triggering cameras or other payloads. It controls autopilot, waypoints, follow me, failsafe, and many other autonomous functions. The flight controller is central to the whole functioning of the UAV, it is responsible for executing movements of UAV according to provided signal. The main control signals for making decision come from sensors e.g. GPS. The interface node of the flight controller does not have to be single node. Complexity of task affect the type of control law that shall be applied. It can be possible to divide mission planning into multiple submodules depending on situation e.g. state machine for monitoring the scenario and optimization engine for cooperative operation. Onboard computer also has an important task because it is the main processing unit of the whole system. It is responsible for sending the correct control signal after it has processed the sensor's data. In this project Jetson Nano is used as an onboard computer. All the data collected by sensors are switched to Jatson through MAVROS. This system provides good computing power and it is easily connectable through ROS with other components.

2.3 PX4

For implement the drone's system the used autopilot is PX4. It consists of two main layers: the flight stack(is an estimation and flight control system) and the middleware is a general robotics layer that can support any type of autonomous robot, providing internal/external communications and hardware integration [13]. The main feature of PX4 is that share a single codebase with all the airframes. All the system design can be defined as reactive, which means that:

1. all functionality is divided into exchangeable and reusable components;
2. messages need for communication are passed in an asynchronous way;
3. the system can deal with varying workload.

The diagram below provides a detailed overview of the building blocks of PX4. The top part of the diagram contains middleware blocks, while the lower section shows the components of the flight stack.

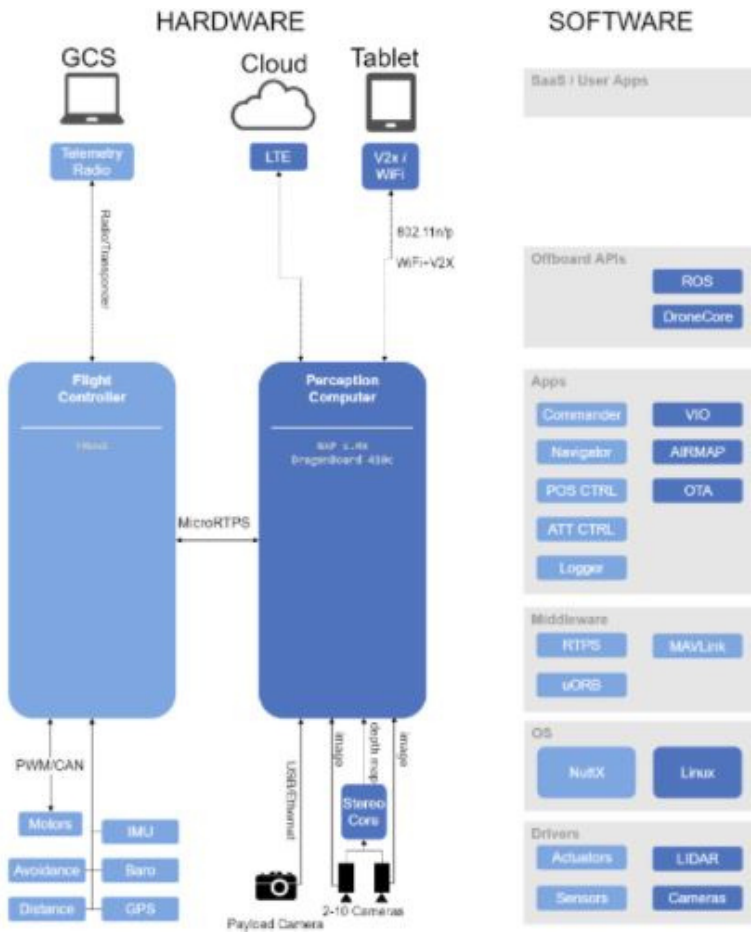


Figure 2.3: Drone platform overview

2.4 Payloads

The final application that the UAV shall perform, affected the equipment mounted on it such as camera, gimbal, battery alarm etc. The most used equipment is the ones that allowed to stream video fitting out by camera, antenna, Video TX. It is a useful system because the pilot on the ground has in a real time the visual of the aircraft.

2.4.1 Multispectral Camera

The capabiliy of a UAV to transmit and capture what it see is considered a fundamental skill. Many applications requires the presence of camera as sensor. It can be very compact, so it is used to observe the surrounding space and sometimes to store images, for spatial mapping and digital space reconstruction. For an environmental study the proposed sensor usually is a camera, which can be placed

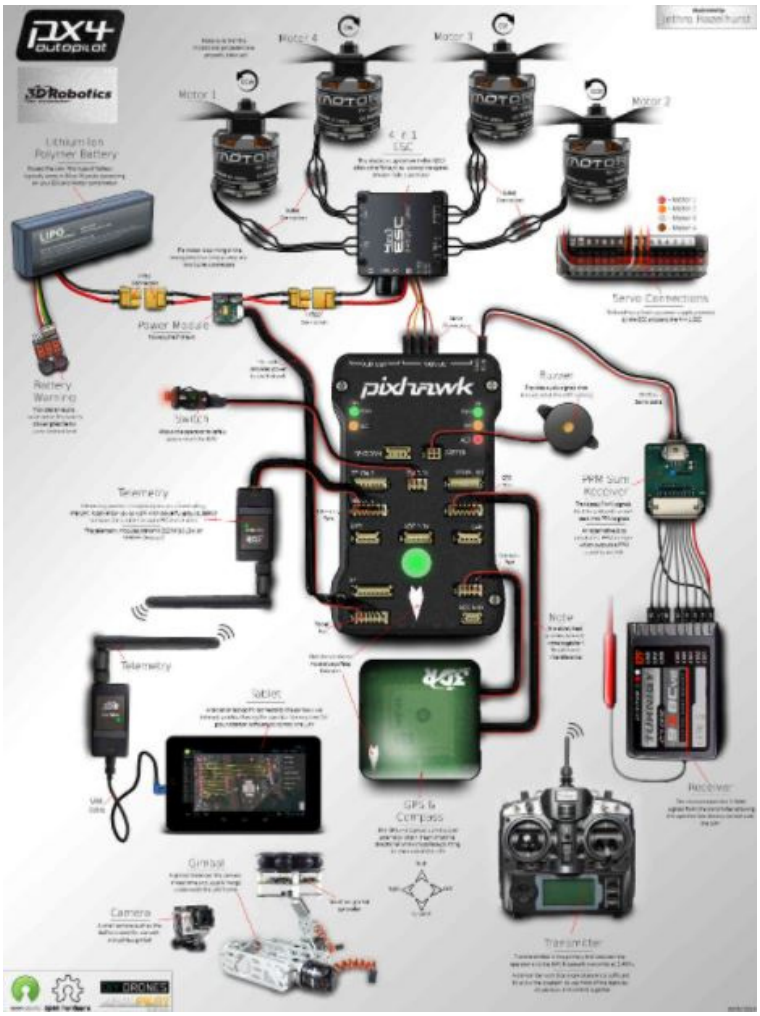


Figure 2.4: Wiring Pixhawk's scheme

either at bottom of the drone or in the front of the drone. The images taken from the camera are from aerial view for bottom camera and from horizontal view front camera. The reasons to locate this sensor in different positions are manifold. For example, if the camera is put at the bottom, it is more used to retrieve information about the target area, such as size and objects. Otherwise placing its at the front, it is used for real time monitoring [16]. Generally the most camera set up on drones is the RGB, which allows to take photo with good quality. However, the extracted spectral information using this device are not sufficient to conduct an assessment of the environment that gives quiet nice results. Using a multi-spectral camera the information gained are more exhaustive, because the collected data from a given object are made by different spectral bands which do not include only red, green and blue band of the electromagnetic spectrum. Multi-spectral cameras work by

imaging different wavelengths of light using one or several images each with a special optical filter that allows only a precise set of light wavelengths to be captured by that imagery. Once processed, the output of the camera data is a set of images where the value of each pixel is equal to percent of light's reflection for that particular wavelength. The collected images, then integrated with specialized software allowed to derive meaningful data.

The chosen on-board camera mounted on hexacopter drone,used for this project, is Sentera 6X, new multispectral sensor designed for unmanned aerial system. It also be available as fully gimbal-stabilized products compatible with DJI Matrice and Inspire series drones, as well as MAVLink-compliant units.

Figure 2.5: Sentera 6X multispectral sensor

In the next chapter will be presented the communication protocol between the different systems, meanwhile the function implemented to trigger the camera will be explained. The idea behind this work is to conceive a procedure that is as simple to carry out as possible, with minimal user intervention. As told before the main goal of the project is to make geo-referenced photos while the hexacopter navigates following a circular trajectory around an object. It will be used a function that trigger the camera each time the vehicle reaches successfully a new waypoint, saving the camera's position and gimbal's orientation, time and data related to the acquisition moments thought Mavros topics. In particular, to handle the information coming from camera, there are two topics which hexacopter is subscribed too: pose and gimbal attitude. Otherwise those type of information require different type of fields, indeed pose belongs to PoseStamped message (location in the three-dimensional space) while the gimbal's orientation is handled using Quaternions (orientation along the four directions). Afterwards each image is saved as png file using cvBridge a ROS library that provides an interface between ROS and OpenCV. The input is the image message, as well as an optional encoding.

2.4.2 Gimbal

A gimbal is used as payload integrated on the implemented system to carry a camera or other sensor with the purpose to stabilize the orientation and rotation in up to three axes. Using a gyroscope, the gimbal allows the camera or sensor to orient and stabilize itself independently of the multi-rotor[15].Normally, the gimbals in order to do move the camera in diverse angles, use servo motors. Thus, the servo motors affected the angles' value. Generally the projects that require the presence of this instrument when the end users are capturing images or video over time and need smooth, stable footage. Without a gimbal, sensor footage is susceptible to noise from vibration or jerky, aggressive motions when the vehicle accelerates or changes attitude. They can also be extremely helpful when there is need for independent control of the camera angle.

Figure 2.6: Drone gimbal

Chapter 3

Communication

As stated before, most of UAV vehicle are equipped with software and hardware used for the communication tasks, usually controlled by a ground control station (GCS). Communication holds a crucial role because it allows all devices to exchange information, store data from the environment, take decisions and so on. There are different kind of communication link that can implemented. In general, the control link is used to send commands to the drone in order to manage its movements. It is normally implemented with a radio link. The control link must be reliable and with low delays. A secondary control link can be implemented using a satellite communication that is less reliable respect to the radio link. It is important because it can replace the primary control link if it fails. Also, the ground control station implements a data link, which can be used for the exchange of telemetry data containing key information about the condition of the UAV vehicle, video streaming and other functional data.

The core of all communications between the drone initial modules is the uORB, which by definition, represents an asynchronous messaging API based on publish and subscribe topics aimed to exchange information. New uORB topics can be added either within the main PX4/Firmware repository or can be added in an out-of-tree message definition. It is necessary to create a new message file in the msg/ directory and add the file name to the msg/CMakeLists.txt list. From this, the needed C/C++ code is automatically generated.

3.1 MAVLink

MAVlink is a messaging protocol that has been designed for the entire UAV ecosystem, made of the drone, internally between their components and the ground control station. It follows a modern hybrid publish-subscribe and point-to-point design pattern: data streams are published as topics while configuration sub-protocols such as the mission protocol or parameter protocol are point-to-point with re-transmission.

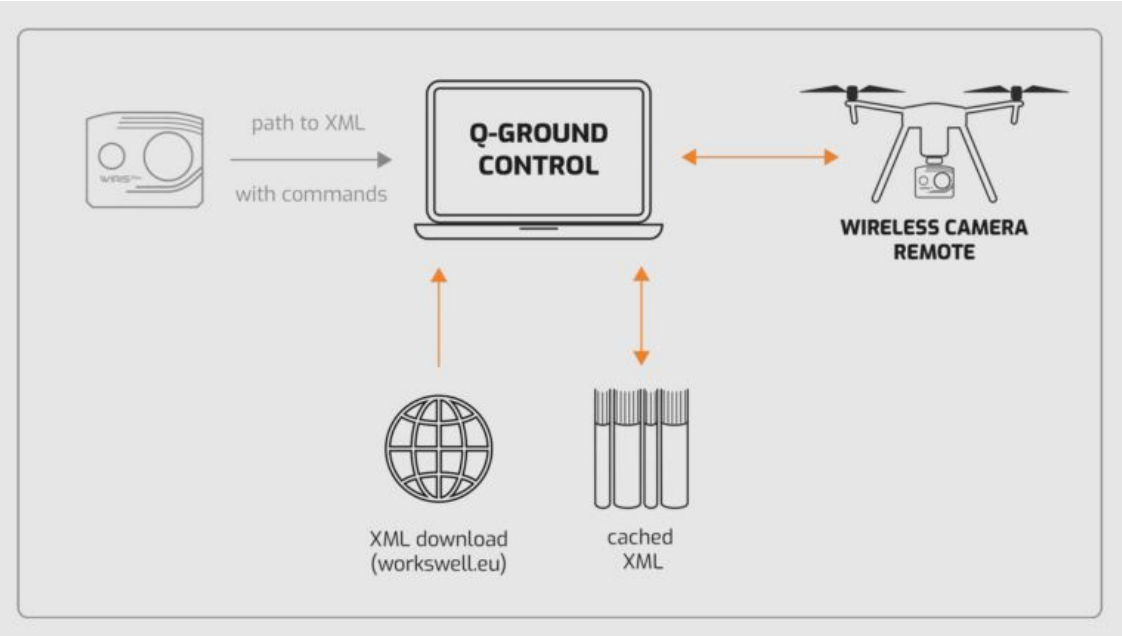


Figure 3.1: MAVLink protocol communication scheme

In particular," PX4 uses MAVLink to communicate with QGroundControl, and as the integration mechanism for connecting to drone components outside of the flight controller: companion computers, MAVLink enabled cameras etc" [17]. MAVLink protocol defines both the message structure and the serialization criterion at the application layer (it basically defines how information should be passed through the network). These messages are then forwarded to the lower layers (i.e., transport layer, physical layer) to be transmitted to the network. One of the biggest advantages using MAVLink is that due to its lightweight structure it supports different types of transport layers and mediums. In fact, it can be transmitted both through serial telemetry low bandwidth channels, operating in the MHz range, namely 433 MHz, 868 MHz or 915 MHz, or through WiFi and Ethernet (TCP/IP networks). Using sub-GHz frequencies guarantees to reach wide communication ranges to handle the unmanned system [16]. On the other hand, using WiFi or Ethernet interface, MAVLink messages are streamed through IP networks. Both the TCP and UDP connection protocols can be used, depending on the required reliability of the application. First MAVLink version (1.0) was released by Lorenz Meier in 2009, whereas a second version with important improvements was released in 2017 (MAVLink 2.0). In the following section the second version will be elaborated as it is the recommended one. Many different programming languages can be used, running on numerous microcontrollers/opening system(including ARM7, ATMega, dsPic, STM32 and Windows, Linux, Android and iOS). It represents an efficient protocol because its handled by packet which has a minimum length use

for overhead, 8 or 14 bytes depending on the used version.

3.1.1 MAVLink 2.0 frame

As can be seen in 3.2 the main MAVLink [18] frame are:

1. STX is the symbol representing MAVLink start of frame.

2. LEN represents the message length in bytes. It is encoded in 1 Byte.

3. INC FLAGS are incompatibility flags: flags affecting the message structure. The flags indicate whether the packet contains some features that must be considered when parsing the packet.

4. CMP FLAGS are compatibility flags. It indicates flags that can be ignored if not understood and it does not prevent the parser from processing the message even if the flag cannot be interpreted.

5. SEQ denotes the sequence number of the message. It is encoded into 1 Byte and takes values from 0 to 255. Once it reaches 255, the sequence number is reset again to 0 and incremented in each generated message

6. SYS ID represents the System ID. Every unmanned system should have its System ID, in particular, if they are managed by one ground station. The System ID 255 is typically allocated for ground stations.

7. COMP ID is the component ID, and it identifies the component of the system that is sending the message.

8. MSGID is the message ID: it refers to the type of the message embedded in the payload. In MAVLink 2.0 it is encoded with 24 bits instead of 8, like in the first version.

9. PAYLOAD is the actual message. It can contain up to 255 bytes.

10. CHECKSUM contains two bytes dedicated for checking the message correctness (1 byte for CKA and 1 byte for CKB).

11. SIGNATURE: finally, MAVLink 2.0 uses an optional Signature field of 13 bytes to ensure that the link is tamper-proof. This features significantly improve security aspects of the MAVLink 1.0 as it allows the authentication of the message and verifies that it originates from a trusted source.

Figure 3.2: MAVLink frame for 8bytes

3.1.2 Camera Protocol

There is a several protocol to communicate with the camera called camera protocol, based on a sequence of exchanged messages to control the device's status and to configure correctly camera payloads. Many flag variables are used to provide information about camera capabilities, if it supports image capture or video streaming and if it needs to be configured in a particular mode for capture etc.

The protocol followed is the Heartbeat ones which sends at constant rate all the request to investigate the camera's status and to assign camera component ID the first time that heartbeat is detected by a new camera [19]. The command ACK message carry the result of a request sent to the camera. On success (result is MAV-RESULT-ACCEPTED) the camera component must then send a CAMERA INFORMATION message. If CAMERA INFORMATION is not received after receiving an ACK with MAV-RESULT-ACCEPTED, the protocol assumes the message was lost, and the cycle of sending MAVlink request message is repeated. If CAMERA INFORMATION is still not received after three cycle repeats, the GCS may assume that the camera is not supported. The CAMERA INFORMATION response contains the bare minimum information about the camera and what it can or cannot do. This is sufficient for basic image and/or video capture. If a camera provides finer control over its settings CAMERA INFORMATION.camdefinitionURI will include a URI to a Camera Definition File. If this URI exists, the GCS will request it, parse it and prepare the UI for the user to control the camera settings.

3.1.3 MavROS gimbal's integration

The UAV, as said before, is equipped with a gimbal which communicate with the fight controller though a Gimbal Protocol. This type of Mavlink protocol does not yet have an associated Mavlink message in Mavros. So, to allow this communication it will be used an implemented Mavsdk Integration, a python component that allows communication ROsBridge and the flight controller. It works like a bridge between gimbal, fight controller and Ros. When it is required a smoothly rotation of a camera along an axis, it integrates an additional device called gimbal. To perform a correct identification of the airplane's positions it is necessary that the gimbal. Vectorial consideration can be done to define the starting direction of the

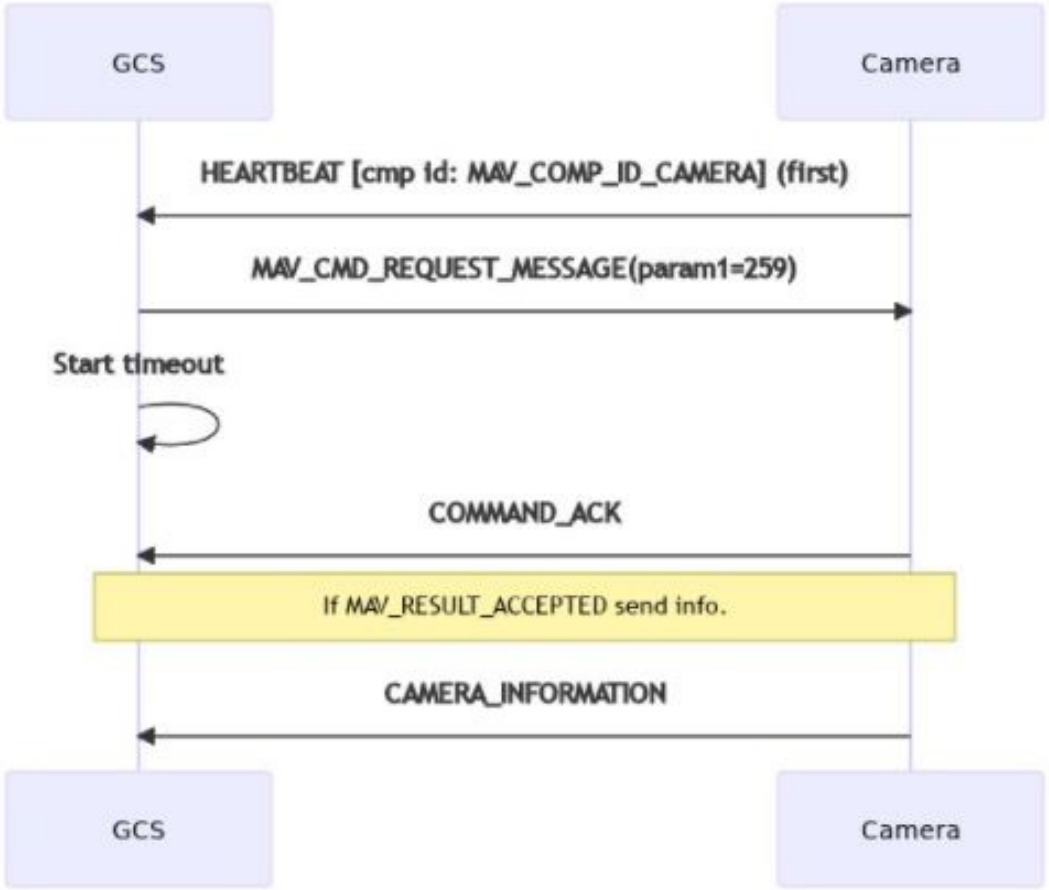


Figure 3.3: Messages

vector related to the gimbal, directed along the x axis . The difference between the actual UAV position and the vector coordinates that locate the parking aircraft will expressed by the as vector. By the Rodrigues' rotation formula a rotation of a vector in space can be computed, starting from an axis and angle of rotation [25]. First, a matrix G is computed which represents the linear transformation between the analyzed vector u and any other vector v, so a cross product is computed [24].

$$u \times v = \quad \mathbf{G} \cdot \mathbf{v} \ , \quad with$$

$$G = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

Those evaluation allows us to compute the direction and orientation of the starting gimbal's position. The final vector point toward the center of the airplane.

Chapter 4

Trajectory

4.1 Path planning

In the field of automation, path planning and trajectory planning have a crucial role. Path planning algorithms generate a geometric path, from an initial to final point, passing through predefined via-points either in the joint space or in the operating space of the robot, while trajectory planning algorithms takes a given geometric path and endow its with the time information. The time of passage at the via-points influences is defined by trajectory planning and has a key role because it influences not only the kinematic properties of the motion, but also the dynamic ones [21]. Path planning is an operation performed to compute a kinematically-feasible curve in space connecting a point. It does not take into account the presence of obstacles, which are generally processed by obstacle avoidance algorithm [23]. Generally the developed algorithm does not make clear distinction between these two subjects; otherwise the common target is to compute a path which avoids any collision with obstacles between initial and final point.
It's necessary to distinguish between path and the word trajectory. The first is the geometric curve in space, whereas the second is a curve parameterized in time. All the path planning algorithms took in consideration work online and in real-time. Indeed this type of algorithm allows to integrate new information over time and update its result according to them. On the contrary, the offline ones compute the path relying on initial base information and there is any way to update the path considering new information acquired by real-time sensing. The real-time characteristic only regards online algorithms and depends on the speed requirements. Therefore if the algorithm cannot respect time constraint, it is not real-time because only real-time algorithms can effectively react to sudden events in the environment. Dijkstra's algorithm, A*, D* and Artificial Potential Field [22], all these methods listed have a common issue, they require that the rules used to compute the trajectory are explicitly coded. This is quite simple if the only goal is to compute the shortest path to desired location. However, if the required path has to include

more complex characteristics, those algorithms become quite restrictive because to include the additional constraints on the path computation can result extremely difficult to code.To solve this limitations, optimization-based path planning algorithms are often used.
Those type of algorithms allow to compute a route optimized with respect to any desired constraint. The only condition is that each constraint must be written as an expression and integrated inside a cost function that will be then minimized in order to find the optimal path. This kind of approach allows to find very efficient paths. Different methods can be exploited to perform the optimization process. One possible strategy is to write the trajectory planning problem as a convex optimization problem (e.g. Linear or Quadratic Programming) and solve it. This solution is fairly fast and the solution existence is guaranteed, but it has some drawbacks. In fact, it is not always easy to cast trajectory planning as a convex optimization problem. Moreover, all the constraints (collision avoidance, minimum travel distance...) also need to be expressed in an algebraic form suitable to be included inside the investigated problem. A different approach to obtain optimization/based trajectories is to use stochastic algorithms. In this case the problem formulation can be easier since the cost function does not need to have a specific shape like e.g. in Linear Programming. Examples of algorithms that can be used to implement a stochastic path planning algorithm are Particle Swarm Optimization and Stochastic Gradient Descent [20]. It is worth noting that, in most cases, the optimization process performed by these algorithms is not a global one, but is limited to coming close enough to the optimum. In fact, the main limitation of optimization-based path planning algorithms is that their computational cost is much higher than the one of the algorithms introduced in the sections above. Since the microcomputers on which these algorithms usually run have limited computational power, it is difficult to have them run in real time, and usually their time constraints are more relaxed than the ones of the deterministic algorithm. Indeed, to keep the computational cost sufficiently low, often these algorithms end up finding a sub-optimal solution (which is, however, generally acceptable for the path planning purpose). To overcome the time limitations posed by optimization algorithms, different strategies can be used.

4.2 Graph search

The first experimented approach was the graph search and in particular the rapidly exploring random tree (RRT).This algorithm is based on random sampling of a configuration space. The sampled configurations are connected to a tree structure in which the result path can be found. The optimization logic behind these algorithms is to achieve the minimization of the distance covered by the vehicle without involving kinematic or performances characteristic. Topological model of the environment is the starting element of a path-finding algorithm, which defines

relations between spaces and determines what poses a vehicle can reach. Among the most used approaches for path planning there are: cell decomposition methods and sampling-based methods. In the cell decomposition methods, the free space of the environment is divided into small regions called "cells". Classical graph search algorithms treat each cells as a graph node and they search the shortest path with 'greedy' logics.The goal is to provide a collision-free path to reach a target pose. A 3D cell decomposition model is a grid where each cell has neighbors that determine the connectivity within the model. The connectivity between a cell and its neighbors can be defined by the faces, edges and vertices of the chosen 3D geometrical structure.

4.2.1 RRT

To apply the RRT algorithm, the preliminary work done is to build a map of the environment where the drone shall navigate. Using basic geometrical figures it has been designed a sketch of the plane that represents only the shape. Rviz is the tool used for this first trail, that has been implemented to test the efficiency of RRT algorithm in order to understand if the results are suitable with the wanted trajectory. The algorithm is implemented using python and 4.1 shows the main steps.

RRT Algorithm (x_{start}, x_{goal}, step, n)

```
1   G.initialize(x_start)
2   for i = 1 to n do
3       x_rand = Sample()
4       x_near = near(x_rand, G)
5       x_new = steer(x_rand, x_near, step_size)
6       G.add_node(x_new)
7       G.add_edge(x_new, x_near)
8       if x_new = x_goal
9           success()
```

Figure 4.1: Algorithm for RRT path search

The starting position and the goal is selected directly inside Rviz using the 2D Navigation topi. It is visible in 4.2 the trajectory is not circular and even if the reached positions are quite closer to the centered objects, the path is not enough smooth.

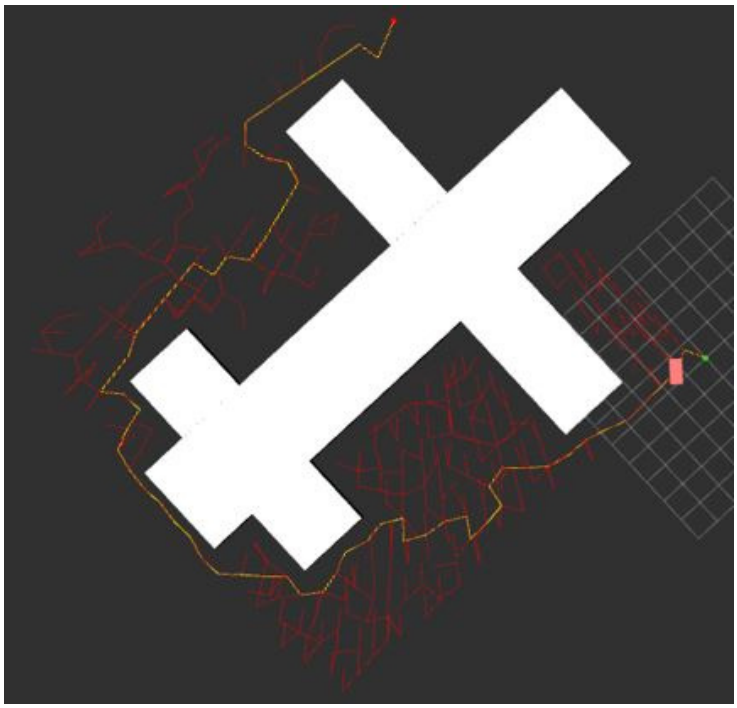


Figure 4.2: In yellow it is point out the evaluated trajectory

4.3 Trajectory estimations starting from images

Since the previous results do not fulfill the target, a new method is experimented to reach the path planning goal finalized to survey the surfaces of the airplane. Images are the key element in this approach because they allow to estimate the local position of the object basing on restricted information. However a data set can be collected and analyzed to reach information about environment and obstacle under the test.
The work will be divided in three phase:

1. Development of a circular trajectory where the obstacle is inscribed to gather photos and localize it.
2. A neural network will be used to segment the previous images in order to create a new list of waypoints
3. Finally the new path is tested to verify if the target is achieved successfully.

4.4 Localize the object's position

In the first phase it's important to choose a suitable value of the height at which the UAV structure shall take the pictures because for safety and legal issues, it is

not allowed to navigate vertically to the airplane. This is the main reason why a circular path is created to localize it using the set of data. It has been supposed that the object under the interest is positioned at the center of the area under the investigation, due to this simplification a circular path that surrounds the airplane is created using the POI coordinates as the region where camera shall point. Also, the radius of the circumference affects the future step, in particular the segmentation, so the generation of the new waypoints. These features will be treated deeply in the next section. To sum up, the only information used to set up the first flight around the airplane are:

1. The center of the apron, coincident with the coordinates of the POI
2. The vertex's coordinates of the apron
3. The starting position, where the drone made the take-off

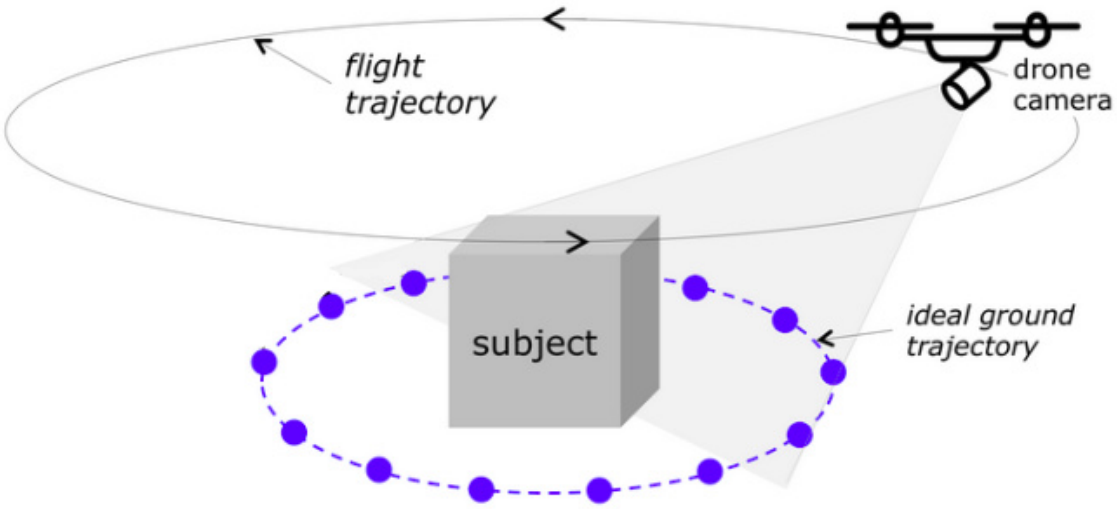


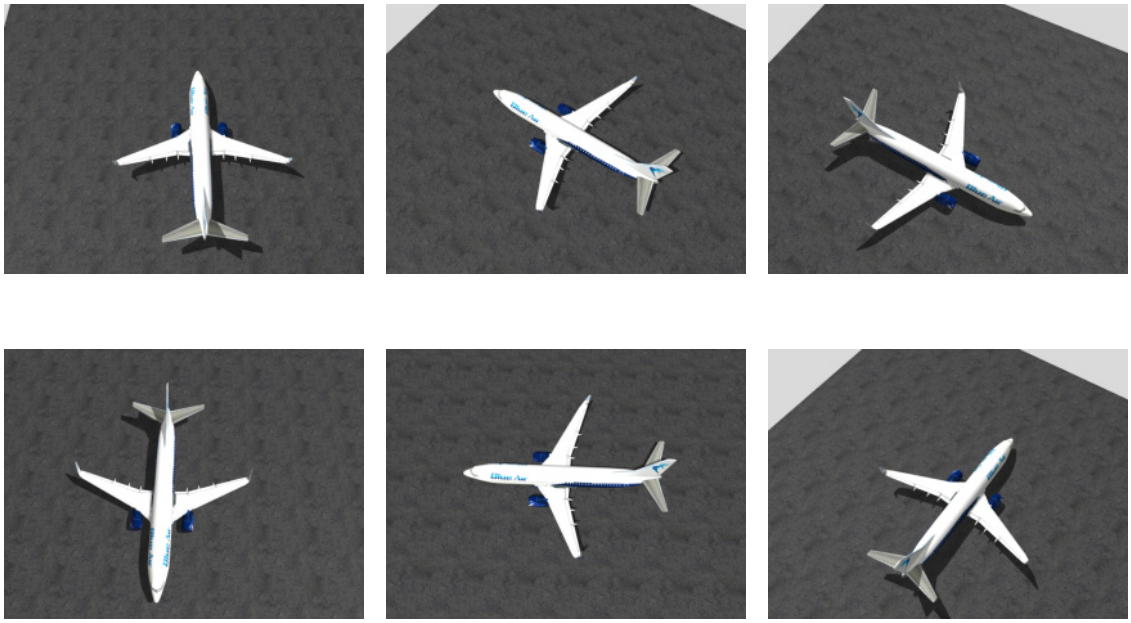
Figure 4.3: The desired target in the first phase

For this step, the trajectory is reached basing on the parameters listed above, but before starting with mathematical computation, some checks are done to satisfy all the condition of safety, in particular the minimum distance to which the UAV shall stay away from the airplane's edge. Through geometrical construction, the round corner is dividend by the total number of waypoints chosen to create a set of points all at the same distance. For each point is evaluated the set of Cartesian coordinates basing on trigonometrical computation and finally the set of waypoints is reorganized in such a way to start the navigation from the nearest evaluated point respect to the home position or rather the location where the typhoon made the take-off. Before sending those computed set of valued, it is associated an attributed with the action that the UAV shall do like, LOITERING, TAKE-OFF, RETURN

HOME, LOITERING or NAVIGATE TO WAYPOINT. Assuming that the local position of the drone is the same of camera so each frame will be geo-referenced by the following information:

1. Cartesian coordinates along the three axes.
2. Quaternions values related to the orientation of the gimbal.

Each subset of data is necessary to make the drone moves autonomously, so they are published on a well-defined topic according to the specific message that must be sent to the autopilot. Before sending the information through Mavlink communication, data are packaged using MavROS. The aim of ROS package is not limited to transmit aircraft attitude, global and local position, speed ecc. So, one of the key features of MavROS is the capability to send command to a vehicle with a service called "offboard mode", which is fully compatible with PX4 autopilot stack. So, in the first planned mission the georeferenced images gather essentially information used to compute the new path. For this purpose, each frame's position and orientation are stored inside a text file.



4.5 New set of waypoints

The images collected during the phase one are necessary to plan the trajectory around the plane in such a way to get more closer to the obstacle and having better images and visualization. Reaching this task has a great impact on the required service investigated in the Spectral Evidence Project since having a closer trajectory

to the aircraft allows localize better the presence of the ice upon the surfaces. It clear that the conventional path planning algorithms like RRT, A etc are not used to generate the waypoints because obstacle avoidance skill is not linked to the task of this application. Indeed, thought the image collected in the first phase, the object's position can be estimated so the staring environment is not completely unknown.
Before starting to explain the method implemented to achieve the best path to navigate around the plane, some observation shall be done due to the fact that reference and measurement system is not only rotated and but also changed from meter to pixel and vice vera. A method to estimate the bitmask created around the aircraft is essentially to quantify the amount of the distance to stay away from the surface of the plane. The length and the width value of the aircraft are picked up from model used in the simulation environment and these will be the reference dimensions on 2D plane. The airplane silhouette is built from an image taken from Gazebo by implementing a python code that extract the edge of the object under the interest. The result will be a scratch of the plane with more details as much as the threshold is set higher. The selected value of the bitmask is applied to the original image and the result can be seen in the next figure. Comparing these two figures, it allows to compute the pixel value of distance that next path can have from the airplane, and it is 210 pixels.

4.5.1 Conversion from pixel to meter

It is needed a method to make conversion from pixel to meters, and to reach this purpose the dimension of the plane is done also in pixel. This allows to compute the ration quantity meter/pixel, because the length head-tail of the plane either in pixel or in meter are already computed across the previous step.

$$\frac{meter}{pixel} = \frac{AircraftLenght_{meter}}{Silhouette'sLenght_{pixel}}$$

So, the estimated distance is about 3.6 meters. As it can be notice by analyzing the figure, the procedure used to extract the contour adds more details due to the shadows and projections items present on the image collected in the simulation. There are additional localized dense points near the winglet and in the back side of the plane, due to the tail-fletching. For this reason, the bitmask in specific part of the silhouette has some irregularity because on those area it will be bigger than the imposed value.
The developed algorithm using the different python's libraries is based on the segmented images reached from the CNN. The network is trained by using the data collected with the multispectral camera each time the UAV reached the waypoints that belong to the circular trajectory planned in the first phase. The convolutional network developed in the Deep Way learning approach is fed with an occupancy

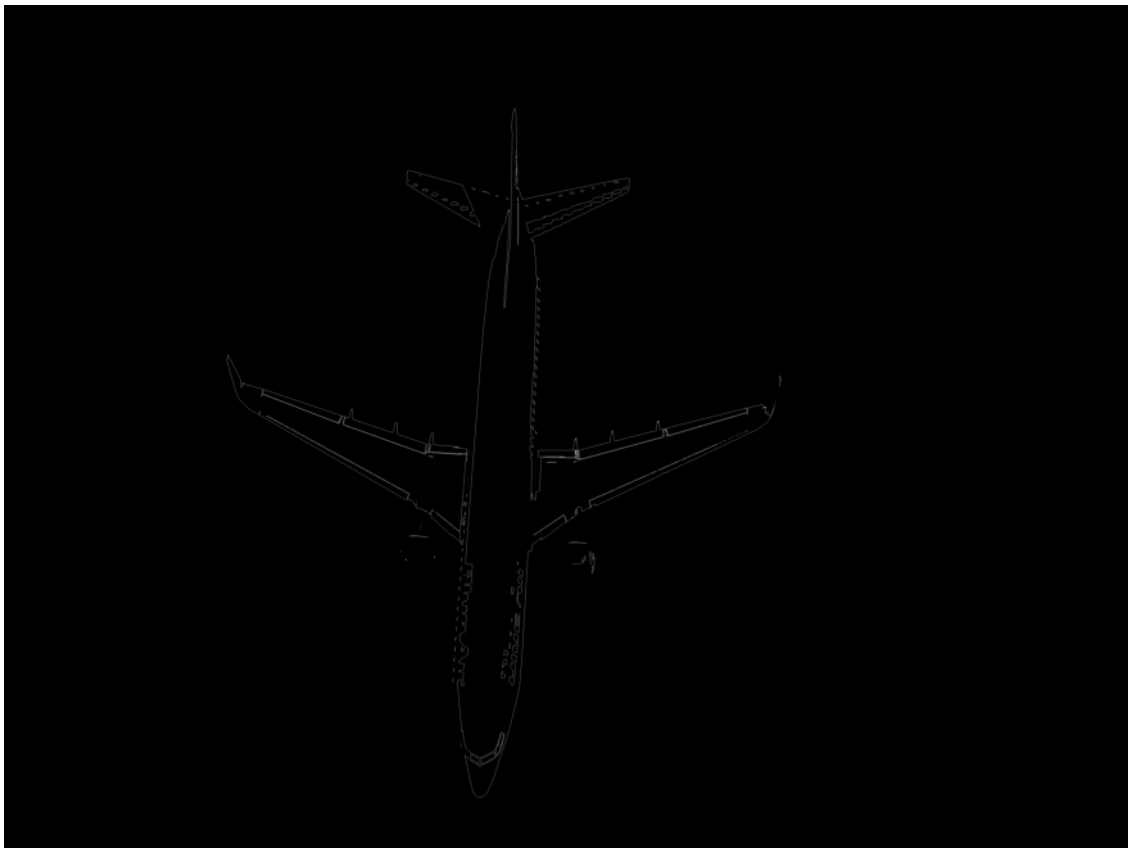
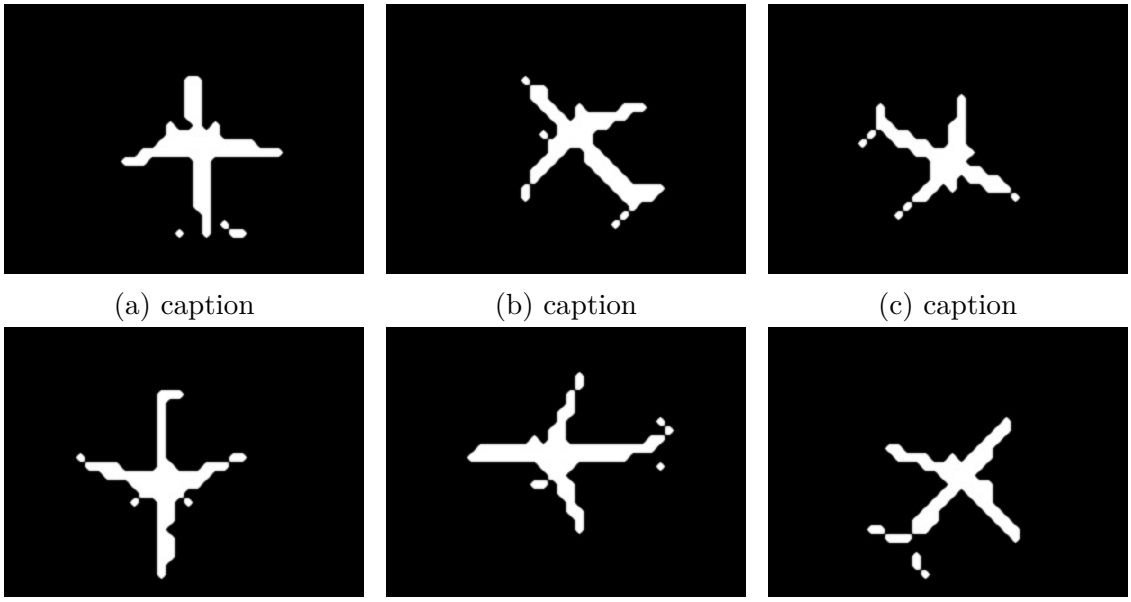


Figure 4.6: Edge of the airplane's model implemented in Gazebo simulation

grid map and predicts waypoints for global path generation [28]. In this application neural network is employed to make segmentation of the collected hundred images. In this way a new set of data is achieved representing the previous scenario as an occupancy grid. Image segmentation is defined as partitioning of an image into non-overlapped, consistent regions which are homogeneous with respect to some characteristics such as gray value or texture. It's an important stage of the image recognition, because it extracts the objects of our interest, for further processing. Segmentation plays a primary role in the field of computer vision and in the next developed algorithm. The importance of scene understanding as a core computer vision problem is highlighted by the fact that an increasing number of applications nourish from inferring knowledge from imagery in practice it is a classification of image pixel. However convolutional Neural Network suffers from a couple of drawbacks for the segmentation task:

1. Feature is not compatible with the segmentation task
2. Features does not contain enough spatial information for precise boundary generation

3. It is a task that requires time and would greatly affect the final performance



(a) caption (b) caption (c) caption

(d) caption (e) caption (f) caption

The reasons listed above, suggest the needs of some consideration on the segmented images, since as it can be seen below, if we compare first phase images with the ones gained with the segmentation, the latter have a strong distortion due to the neural network. Nevertheless, this do not set up a huge problem and in the following section the approach used to solve all those inconsistencies will be presented.
At this stage the object under the interest is precisely localized based on the collected images in the first phase. After, they are passed to the neural network and manually processed to select the best one that enact at the best way our environment. As said before, at the end of the first phase one of the reached information in this stage is a file that list the number of image's frame, the position along the three axis (x ,y ,z) and the quaternion angle that save the camera's orientation respect to the plane. Those data are crucial for the algorithm that will be explained below, which is based on a photogrammetric approach that include the collinear equation.

4.5.2 Processing segmented images to create the mask

Starting from the chosen image, an offset is set around the perimeter of the aircraft to create a safe flight staying sufficiently away from the surface. All the image's pixel will be under the investigation because pixels are the raw building block of an image. They are the smallest portion of an image that a computer is capable of displaying. Every pixel-based image has three basic characteristics:

1. Dimension is the attribute related to the physical size or resolution, so image has no size until it has assigned how large is each pixel, that is defined resolution.

2. Bit depth defines how many shades and colors are present in the image, each pixel's tone is defined by one or more numbers

3. Image mode is the one that dictates whether all those numbers represent shades of gray or colors

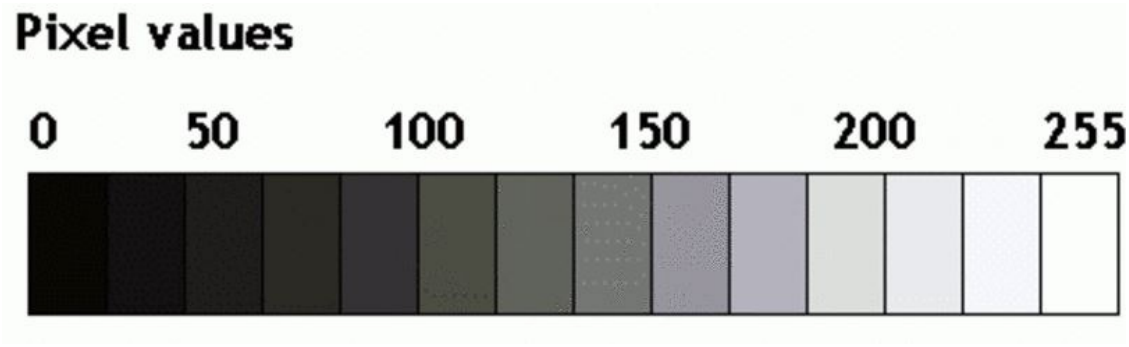


Figure 4.8: Shadow of gray

The selected value of the bitmask to be apply at the contour is chosen using the pixel parameter in such a way to create a distance from the surfaces 5 meter. Once we applied this threshold, trough python OpenCv library, all the point of the edge is selected and stored in a file, using the RBG properties of the pixel.
The feature of the segmented image is that can be treated as a binary grid because it's a 1-bit image (each pixel is represented by one bit of information-either a one or zero) which means a black and white picture. Indeed, looking at the figure we can distinguish easily the free space from the region that has inside the object under the investigation. All the black pixels are an unoccupied area where eventually a new waypoint can be set, while the white pixels delimitate the region that includes the examined object. Therefore, the pixel color information is used to understand if we are looking at a point that belong to the plane or to the free space. In general, the number that describe pixels related to tonal valued and has lower numbers representing darker tones and higher ones indicate brighter tones. So, considering a greyscale image, in the 8-bit channel, 0 represents solid black, while 255 denotes pure white. Therefore, numerical value associated to each image's pixel is used with the edge function to store all the pixel that belong to the plane's perimeter in a text file. So, up to now the airplane is transformed in a matrix where in each row there is cartesian coordinates which point to the pixel position that satisfy the required condition. The next section will focus on how to convert this matrix with pixel value in real coordinates that become the new waypoints.

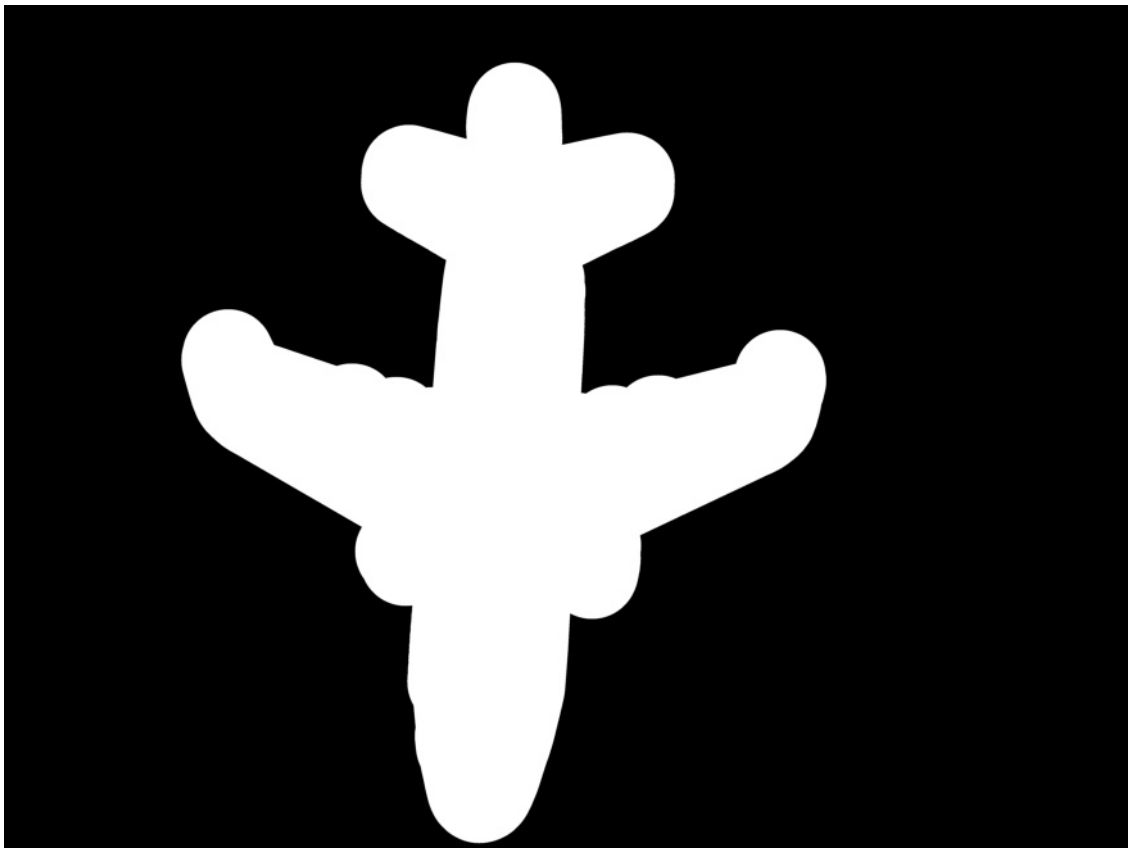


Figure 4.9: Bitmask on around the surface's edge

4.5.3 Pixel conversion into Cartesian coordinates

The main topic discussed topic are photogrammetric and collinear equation. Photogrammetry is a method that allows to obtain reliable information about physical objects and the environment through processes of recording, measuring and interpreting photographic images. In all photogrammetric applications, the accuracy of measurement depends on the geometric reliability of the camera and the feasibility to perform accurate coordinate measurements on digital images. Camera reliability is achieved by performing a robust camera calibration procedure to estimate the interior orientation parameters. The approach applied for this application is a static close-range photogrammetric technique with a single fixes camera and is based on collinearity equation[26]. This approach required a sequence of image taken with single calibration from several position and the last image taken is kept unchanged and it corresponds to camera station. The proposed approach assume that the object's motion is in two dimensions only so the measurement of the image coordinates correspond to a specific point of the object can be used to evaluate

its coordinates in the space by substituting the collinearity equation. The advantage of those mathematic equations is the unnecessary knowledge of the kinematic properties of the object under the analysis since no dynamic model is used. In analytical photogrammetry the most fundamental and useful relationship is the collinearity condition expressed with two equations for any point on an image: the first equation is for the x coordinate and the second for the y ones.

$$x_a = x_O - f\frac{m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)} \quad (1)$$

$$y_a = y_O - f\frac{m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)} \quad (2)$$

Referring to the above equations, x_a and y_a are the photo coordinates of an image point a with respect to the principal point; X_A, Y_A, and Z_A are the object space coordinates of the point; X_L, Y_L, and Z_L are the object space coordinates of the camera (exposure) station; f is the camera focal length; x_O and y_O are the coordinates of the principal point; and the m_{ij} are functions of the three rotation angles. The rotation angles are defined in terms of a right-handed coordinate system by three Euler angles Omega, Phi, and Psi. In the case under the study, some simplifications are done because we use a single fixed camera and the object space coordinates are assumed to be zero. The rotation matrix is computed by firstly convert the quaternion associated camera orientation into Euler angle. Those values are picked from the test file generated in the first phase and they are related to the chosen image segmented with the CNN approach. Equation da quaternion a euler The final rotation matrix shall have a RPY convention so the single rotation around the three axes are multiplied in this order:

$$M = M_z M_y M_x$$

where :

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\psi & -sin\psi \\ 0 & sin\psi & cos\psi \end{bmatrix}$$

$$M_y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

$$M_z = \begin{bmatrix} cos\varphi & -sin\varphi & 0 \\ sin\varphi & cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

It must take into the account in the computation of the X, Y coordinates that

the reference system of the pixeled images is different from the ones set inside the Gazebo world. Indeed, additional rotation matrices are computed to overlap these two systems. In the simulation the system is a right hand frame with X axis is incoming, Z is directed upwards and Y applies to the right while in the subset of segmented data, we have a left handed frame with Y axis facing down, Z entering the plane and X right-directed. So, from the pixels coordinate system to the simulated world it is necessary to apply two rotations:

1. the first around the y-axis, $M_{90,Y}$,
2. the second around the z ones, $M_{90,Z}$.

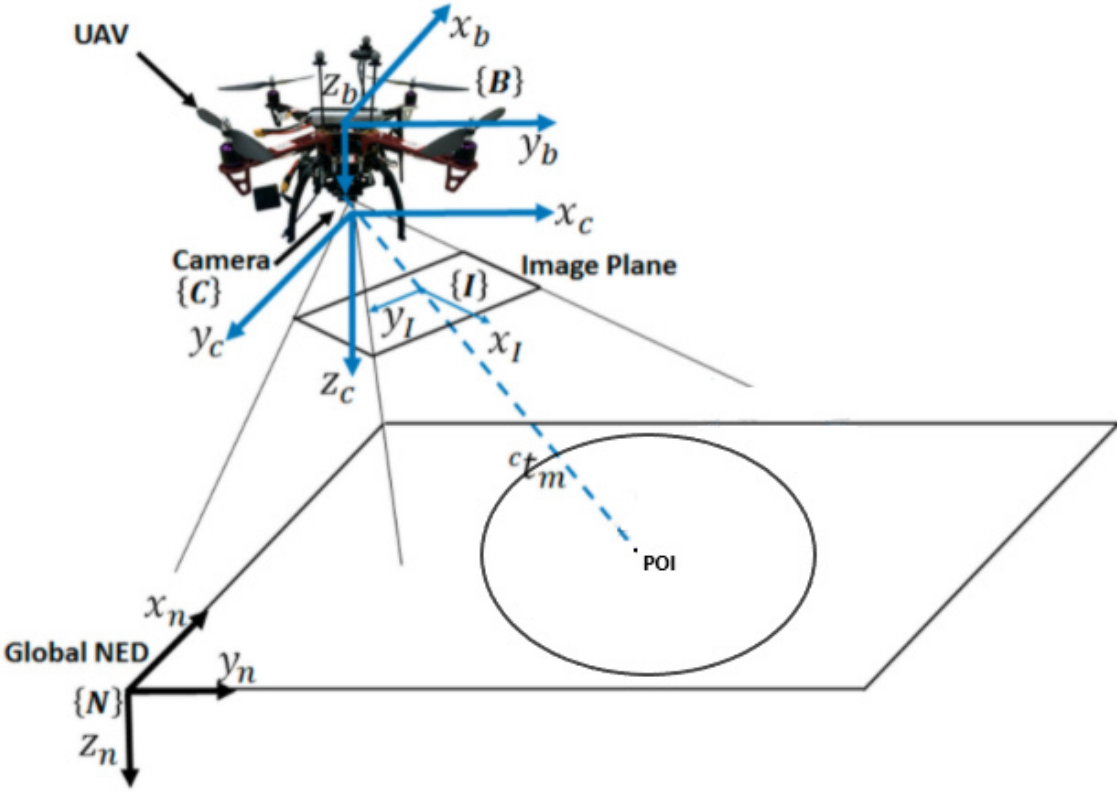


Figure 4.10: Representation of simulated and image plane's reference system

So the final matrix M_{final} will be:

$$M_{final} = M \quad M_{90,Y} \quad M_{90,Z}$$

4.5.4 Euler angles conversion

The computation of the real coordinates applying the collinear equation is done starting from a selected segmented image. As stated before, to each frame it has

been associated a geo-referenced information present inside the text file that is one of the outputs achieved from the first phase. At the selected photo it has been associated a set of coordinates about camera position and orientation, which are required information to proceed with the considered equations. Firstly, the quaternions values, that constitute the orientation information of the gimbal, are converted to Euler angles by these expressions:

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \arctan\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)} \\ \arcsin(2(q_0q_2 - q_3q_1)) \\ \arctan\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)} \end{bmatrix}$$

4.5.5 Condition of Collinearity

From shall it useful to display data from a specific Ros topic to monitor the created publishers. In the equations listed before, there are some coefficients related to the camera that takes photos inside the Gazebo environment during the first phase. The sensor message spectral camera info defines meta information and all the parameter's values are referring to a specific calibration. Intrinsic camera matrix distortion parameters are the ones under the interest. The former picks up the focal length along the x and y axis positioned along the diagonal of the matrix while the coordinates of the principal point are the elements in the last columns of K. the latter depends on the distortion model characterized by 5 parameters. The equation above are rearranged in the following form:

$$X_p = X_L - Z_L\frac{m_{11}x_c + m_{21}y_c - m_{31}f}{m_{13}x_c + m_{23}y_c - m_{33}f} \quad (1)$$

$$Y_p = Y_L - Z_L\frac{m_{12}x_c + m_{22}y_c - m_{32}f}{m_{13}x_c + m_{23}y_c - m_{33}f} \quad (2)$$

where x_c and y_c coordinates are the difference between each coordinate pixel and the principal point $[c_x, c_y]$, to this quantity we add also the distortions correction if they are taken into the account inside the model.

$$x_c = x_{pixel} - c_x \quad (3)$$

$$y_c = y_{pixel} - c_y \quad (4)$$

Finally, the real coordinates associated to the pixel are evaluated and from the starting one thousand positions it has been selected a narrow range, a hundred coordinate couples. They will be the new waypoints and the input to develop the second phase that allows a closer navigation around the airplane. Using the py.plot function the chosen waypoints can be displayed to understand if the new evaluated set of coordinates (X_p,Y_p) are closer to the the attended ones.

4.5.6 Final result and considerations

In 4.11 it can be see the computed path with the collinearity equations which reconstructs the shape of the airplane with the previous set up offset . In yellow, there is also the initial trajectory crossed during the first phase and the new evaluated waypoints are entirely inside the circular path. Clearly comparing the two plots, the new trajectory displays a little distortions more visible in the region with sharpness.
A deep examination is done in order to understand the reasons of this problem, the topics analyzed are the used rotational matrices, the height value set as Z coordinate from where the images in the phase one are captured, the inclination angles of the gimbals. The first error is introduced by the neural network that gave the input dataset to the algorithm that generate the mask around the object. Actually if the value of the imposed bitmask is computed around all the perimeter, different values will be found. This phenomena happens because all shadows' areas can not be discarded from the collected pictures during the Gazebo simulation. It is necessary to find the correct value of pixel chosen as safety distance from the surfaces, trying to not over-estimate it.
However the error threshold introduced by the CNN can not be improved , because this network has been developed for the estimation of a Deep-waypoints, starting from a dataset made by satellite footage, so the initial conditions are quite different. One of the main cause of error is related to the camera's orientation and the UAV's position. indeed, it has been supposed that pictures are taken belonging to the same plane because the Z coordinate of the drone is selected with a higher value. As can be seen in 4.13 the projection on the ground plane related to all the points with the major distance from the camera's lens, cause errors during the computation of the new coordinates with the collinear equations. So the mapping of the path is outside the ideal circular trajectory.

4.5.7 A new experiment with different threshold

To solve those problems, a new simulation has been started with a Z coordinates equal to 100 meters and as radius of the initial circumference is chosen 25 meters. Also the number of collected images are reduced because the purpose of this experiment is to evaluate apply the collinearity conditions to all the available

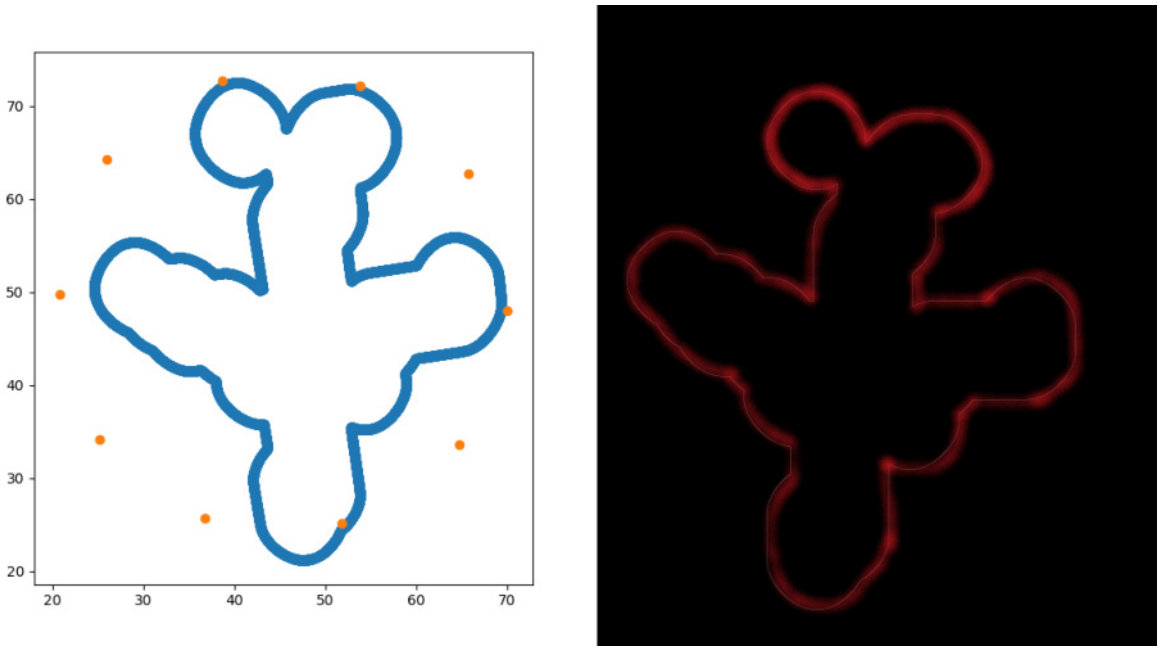


Figure 4.11: Result of collinearity equation applied to a selected image

segmentation reached in this trial. The goal is to estimate a path using all the coordinates that are gained with the computation. In this way the

4.6 Flight with smoothly trajectory and closer to the target

Once the text file with all the waypoints that belong to the new generated path is ready so, the last phase can start by creating a new list with a set of coordinates that allow a closer navigation. As used before, it will be reorganized according to the proximity criteria from the position where the drone is and according to the target it can be still in loitering or in take-off position. In order to send the computed positions across the Mavlink communication, at each of them it is necessary to assign a command of navigation type or loitering. Since in the evaluated coordinates using the collinearity equation, the value of Z coordinate of the drone is not assigned because it can be set arbitrarily and it is packaged into the Mavros message that will be sender to the autopilot. To proceed with the mission a message of type geometry message PoseArray is published using a rostopic with the list of new trajectory is plotted

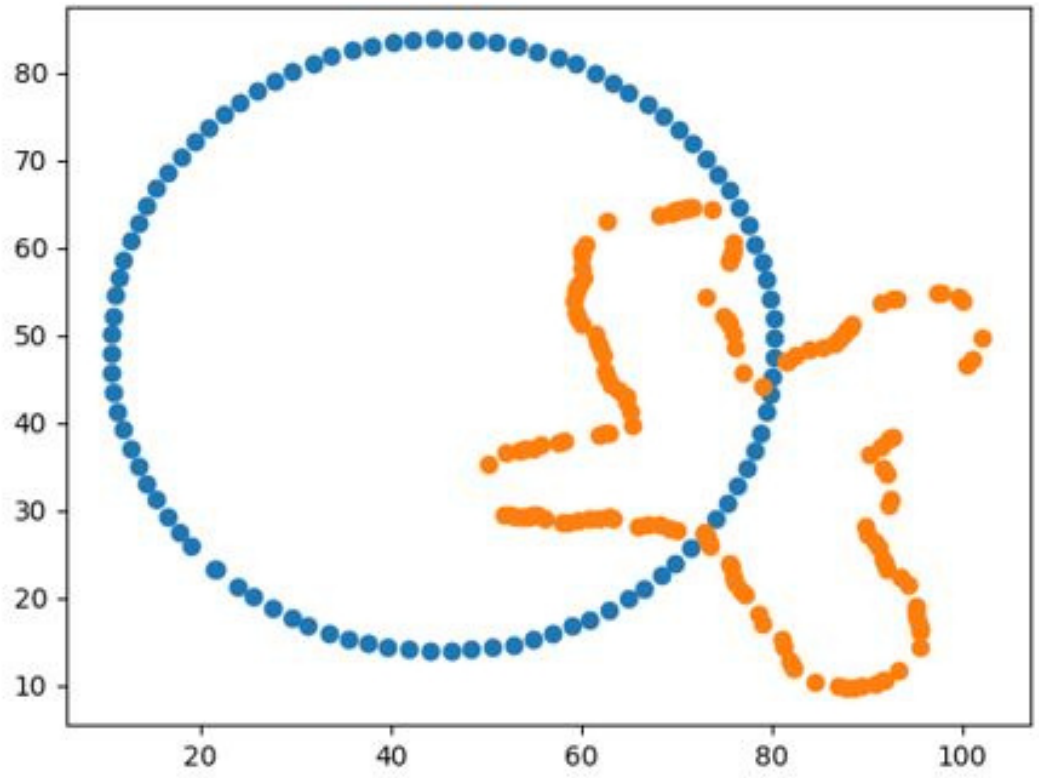


Figure 4.12: Error of computation of points more distant from the lens

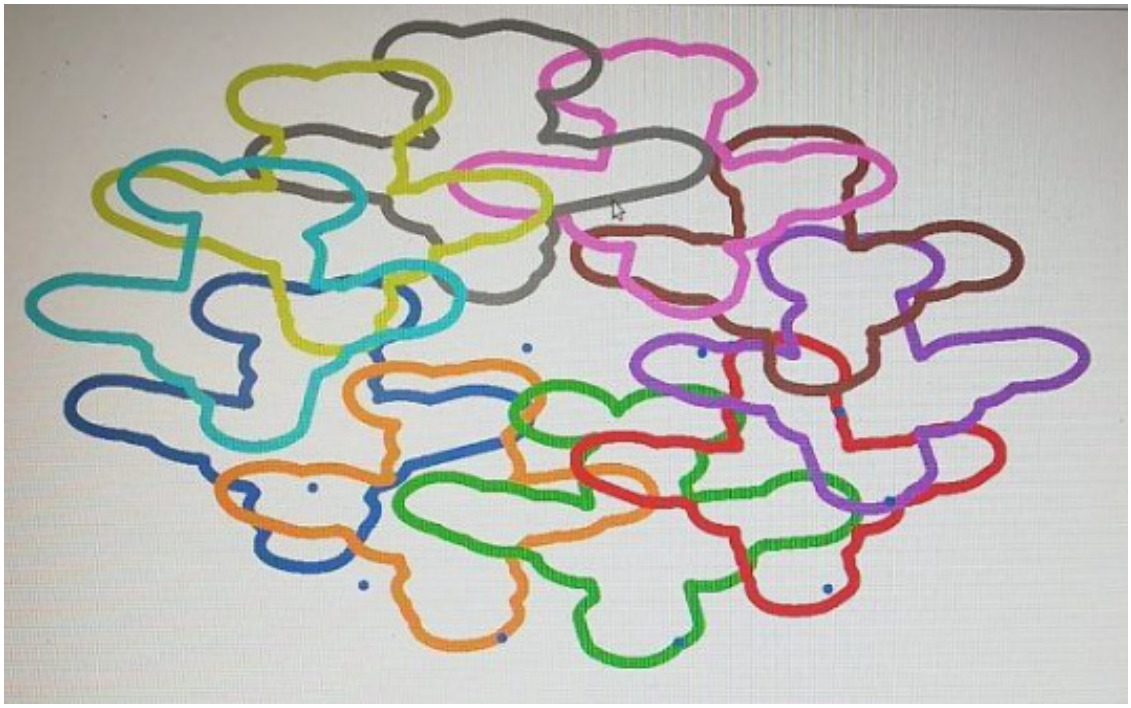


Figure 4.13: Plot of all the possible trajectory for the entire dataset

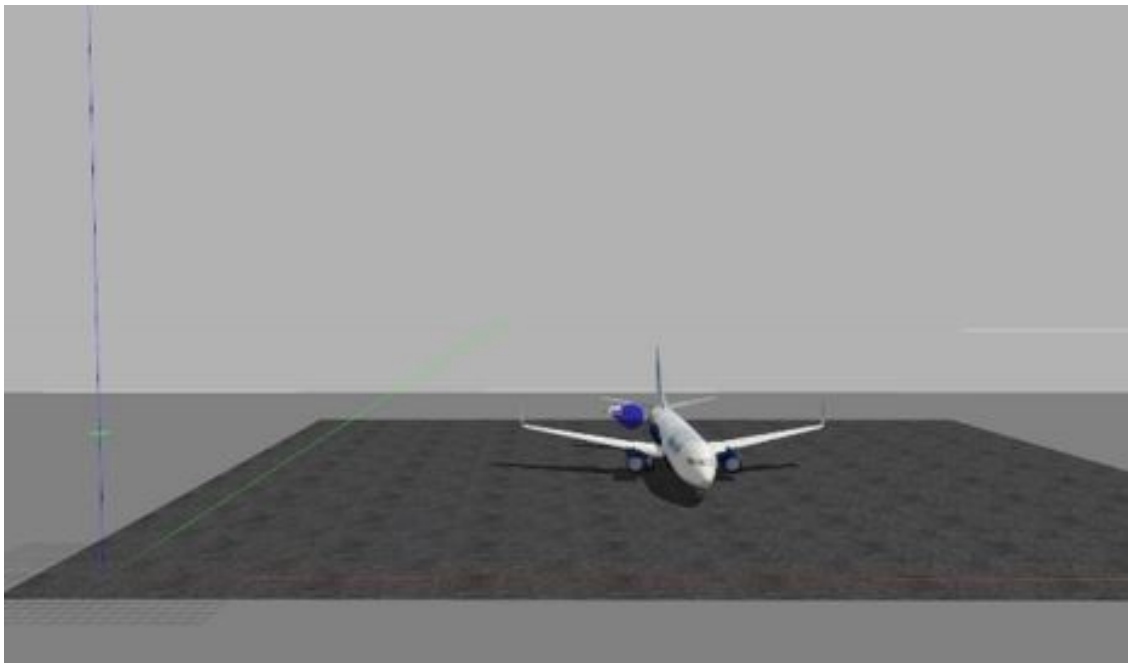


Figure 4.14: Gazebo simulation of phase two

Part II

Second Part

Chapter 5

Simulation environment

5.1 PROJECT tools

5.1.1 ROS

ROS is an open-source collection of software frameworks for robot software development, providing operating system-like functionality. ROS provides standard operating system services such as hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management [29]. Software developed with ROS can consist of numerous modules (called nodes) connected with each other through publisher-subscriber system. Another big advantage of using ROS is availability of vast number of component (ROS packages), in particular MAVROS (MAVLink extendable communication node for ROS with proxy for Ground Control Station) package which implements MAVLink (Micro Air Vehicle Communication Protocol) protocol and provides means for communication between flight controller and onboard computer. ROS can provide several functions similar to the traditional operating system, like hardware abstraction, underlying equipment control, massage passing between threads and software packaging management. In addition, it also provides quantity of utilities and libraries. In our project, the most commonly used ROS tools and packages are RViz, ardroneAutonomy,arTrackAlvar and tum ardrone. Its architecture is based on Peer-to-Peer (P2P) communication between different nodes. These nodes can be defined as programs that perform different tasks and are running on one or more computers being part of a network. The P2P communication is performed through messages that are posted, received and multiplexed by the different running nodes. The available documentation, the existent community and the convenient tools provided by ROS made of this OS the framework of choice. The ROS distribution that has been used is ROS Kinetic, released on May 2016. The Python ROS client libraries have been the implementation of choice to code

the entire project, which is available online on github. RViz stands for ROS visualization. It is a general-purpose 3D visualization environment for robots, sensors, and algorithms. RViz can plot a variety of data types streaming through a typical ROS system, and can be used for any robot and rapidly configured for a particular application.

5.1.2 GAZEBO

Gazebo is an open-source 3D robotics simulator. Gazebo was a component in the Player Project from 2004 through 2011. Gazebo integrated the ODE physics engine, OpenGL rendering, and support code for sensor simulation and actuator control. In 2011, Gazebo became an independent project supported by Willow Garage[30]. In 2012, Open Source Robotics Foundation (OSRF) became the steward of the Gazebo project. OSRF changed its name to Open Robotics in 2018. Gazebo can use multiple high performance physics engines, such as ODE, Bullet, etc (the default is ODE). It provides realistic rendering of environments including high-quality lighting, shadows, and textures. It can model sensors that "see" the simulated environment, such as laser range finders, cameras (including wide-angle), Kinect style sensors, etc. Gazebo can interact with ROS via node communications. Each sub navigation module is encapsulated as individual node subscribing necessary inputs and publishing outputs as topics. To set the required environment for starting simulation related to the described application, all the elements involved are programmed in simulation description format (SDF) language. It is a complete description of all vehicles and their attached sensors. The same description can be done using URDF, that is a standardized XML format file in ROS. In URDF, users can specify the kinematic and dynamic properties of a single robot. It heavily relies on XML descriptive languages. Sensor are configured in robot URDF file to interacts with the environment simultaneously publishing raw data. After a systematic way of information processing in navigation system, new control reference will get published to high fidelity control dynamics simulation node. The dynamic behavior is simulated and reflected in Gazebo environment based on ODE. This loop continues until user-defined termination commands are issued. Sensors are defined inside the plugins, a fragment of code that links together different libraries to increase the functionality of the simulated model. Several types of sensors can be used like lasers sensors, odometry, ultrasound, force, camera etc. The most used plugins are world, sensor, visual, system and GUI and each of them have several functionalities. In this work the most used plugin is the sensor because it acquires information from the environment and to control the state and behavior of the sensor.

5.1.3 Simulation environment

The world under the test have the following elements:

1. Typhoone H 480 model that is the selected UAV structure.
2. A rectangular apron with a real asphalt plane as ground.
3. A realistic aircraft.

The description of the hexrotor model is taken from the STIL gazebo folder defined inside the simulated PX4-Autopilot directory. It is equipped with different sensors like camera, gimbal and lidar.

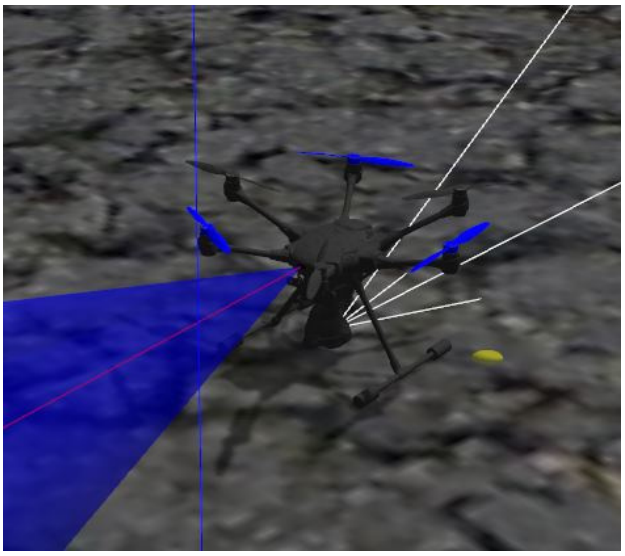


Figure 5.1: Typhoone H480

5.1.4 RVIZ

There are different methods for accessing and reading the main ROS topics to monitor and record the data while mission is performed. From bash, it can be listed all the ROS topics use involved from the implemented application and selecting the correct message type allows also to publish the content related to the specific topic. So, at each instant it can be tracked for example the position of the drone. Another powerful visualization tool is Rviz

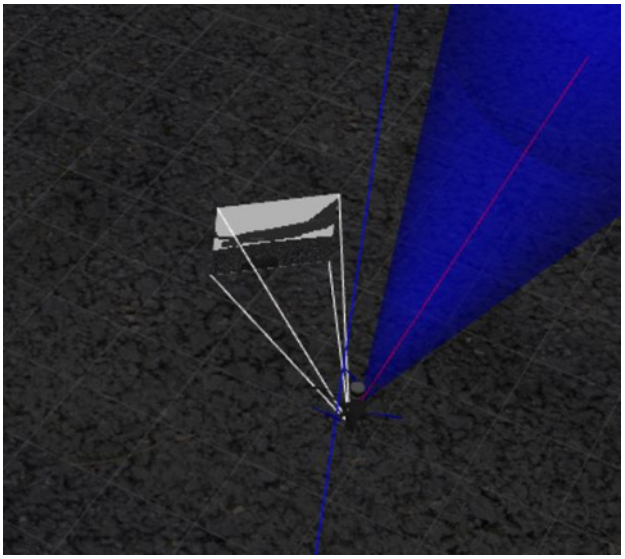


Figure 5.2: Camera's view

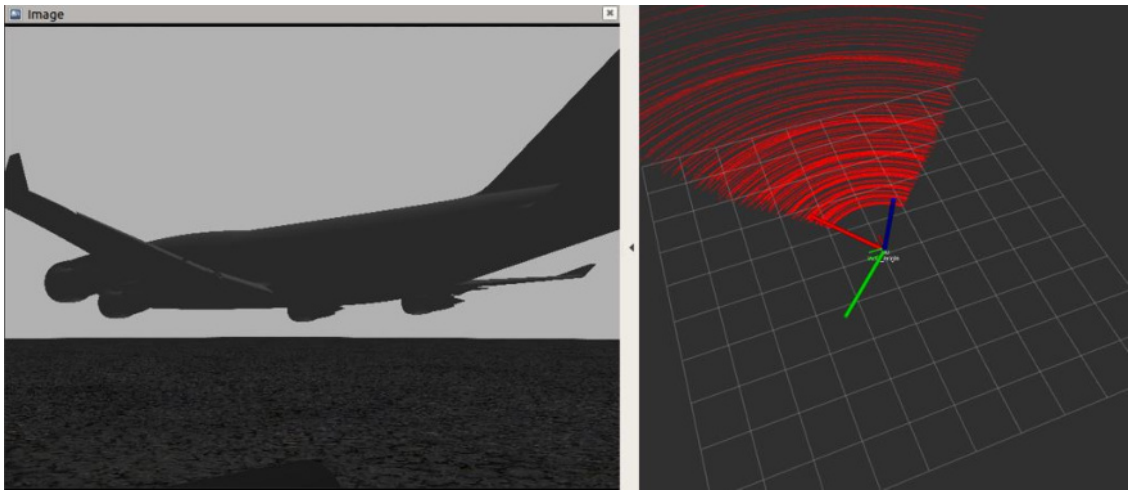


Figure 5.3: Spectral raw image tropic using Rviz

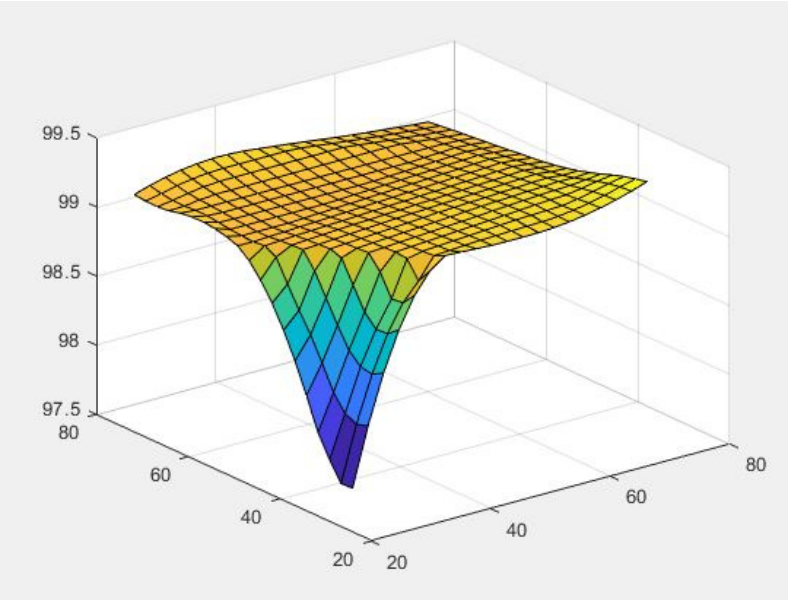


Figure 5.4: 3D plot of trajectory from local positions' data

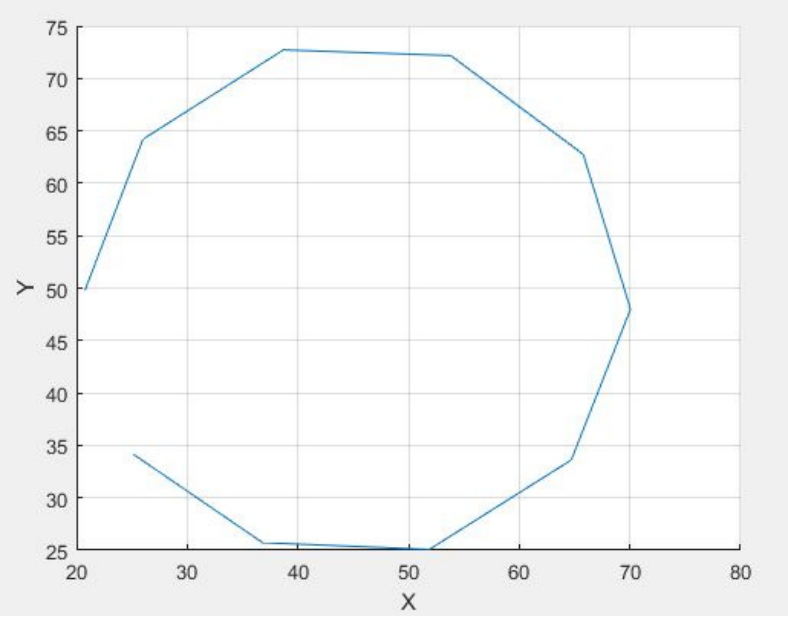


Figure 5.5: Tracking of trajectory while simulating phase one

Chapter 6

Conclusion

The discussed work present a state of art that starting from image processing reach a path that allows to navigate around an object. There are different topics involved in this application, starting from the communication between different sensors, to the path planning which provide a collision free trajectory. The developed algorithm starts with the design of an offset to avoid impact on the object's surfaces. So, a preliminary the perimeter of the specific object is extracted from the images and reshaped with the chosen offset.At this step, the output of the algorithm is a matrix which include all the pixel's coordinates that belong to the evaluated edge. Finally the algorithm is completed by selecting randomly a subset of this points and the collinearity conditions are applied with the suitable consideration about the camera, image and world frame. The computed path is plotted to check if really satisfy the target. Launching out simulated environment in Gazebo, the phase one is tested different time changing the values of radius and height of the drone for collecting the data. The new set of waypoints is proved for a single selected image.
The last part of the work is the achievement of global path analyzing all the possible trajectory evaluated for each image in the built dataset.

6.1 Next step

The future development of this project could be the improvement of algorithm in order to get a robustness model that allows to decrease the threshold of error on a single estimated waypoint. Also, it can be useful to create the dataset of all the computed path using the pixels conversion in local coordinates and estimate a global path. Finally, another step could be test on a physical bodies the achieved results and check if all the hardware configurations are correct.

Bibliography

[1] Shakeri R., Al-Garadi, M.A., Badawy, A., Mohamed, A. Khattab, T., *A Comprehensive Survey and Future Directions*

[2] Grzegorz Chmaj, Henry Selvaraj *UAV obstacle avoidance using image processing techniques*

[3] Shakeri R., Al-Garadi, M.A., Badawy, A., Mohamed, A. Khattab, T., *Distributed Processing Applications for UAV/drones: A Survey*

[4] Meyer D., Fraijo E., Lo E.K.C., Rissolo D., Kuester F. *Optimizing UAV Systems for Rapid Survey and Reconstruction of Large Scale Cultural Heritage Sites.*

[5] Mur-Artal R., Montiel J.M.M., Tardós J.D. ORB-SLAM, *A Versatile and Accurate Monocular SLAM System*

[6] Hartley R.I., Zisserman A., *Multiple View Geometry in Computer Vision*

[7] S.Mentasti, F.Pedersini, *Controlling the Flight of a Drone and Its Camera for 3D Reconstruction of Large Objects*

[8] Hartley R.I., Zisserman A., *Multiple View Geometry in Computer Vision*

[9] S.Mentasti, F.Pedersini, *Controlling the Flight of a Drone and Its Camera for 3D Reconstruction of Large Objects*

[10] Hartley R.I., Zisserman A., *Unmanned aerial vehicles (UAV) are a class of aircrafts that can fly without the onboard presence of pilots*

[11] R.Merket, James Bushell, *Managing the drone revolution: A systematic literature review into the current use of airborne drones and future strategic directions for their effective control*

[12] R.Bappy, A.Rafii, S.Islam*Design and Development of Unmanned Aerial Vehicle (Drone) for Civil Applications*

[13] PX4 *PX4 Architectural Overview - PX4 Developer Guide*

[14] PX4 *Software Overview - PX4 Autopilot*

[15] Mateo Gašparović, Luka Jurjević *Gimbal Influence on the Stability of Exterior Orientation Parameters of UAV Acquired Images*

[16] Sarah Simpson *Next-Generation Multispectral Sensor Features Real-Time Processing*

[17] MAVLink Messaging, PX4 Developer Guide *https://docs.px4.io/master/en/ros*

[18] S. Rapisarda *ROS-Based Data structure for Service Robotics Applications*

[19] Mavlink services *Camera Protocol*

[20] A. Mirshamsi, S. Godio, A. Nobakhti, S. Primatesta, F. Dovis, and G. Guglieri, *A 3d path planning algorithm based on pso for autonomous uavs navigation*

[21] Felix Stache, Jonas Westheider, Federico Magistri, Marija Popović, Cyrill Stachniss, *Adaptive Path Planning for UAV-based Multi-Resolution Semantic Segmentation*

[22] Christian Zammit, Erik-Jan Van Kampen, *Comparison between A* and RRT Algorithms for UAV Path Planning*

[23] Kwangjin Yang, Salah Sukkarieh, *Real-time continuous curvature path planning of UAVS in cluttered environments*

[24] George B. Arfken, Frank E. Harris *One can use the Rodrigues formula to derive the orthogonality and normalization of the Pn*

[25] Laura Downs, Alex Berg *Computing Rotations in 3D*

[26] Paul R. Wolf, D.Bon A. Dewitt, D.Benjamin E. *Elements of Photogrammetry with Applications in GIS, 4th Edition*

[27] Vittorio Mazzia, Francesco Salvetti, Diego Aghi, Marcello Chiaberge *DeepWay: a Deep Learning Waypoint Estimator for Global Path Generation*

[28] Heramb Nandkishor Joshi, Prof. J. P. Shinde *An Image Based Path Planning And Motion Planning for Autonomous Robot*

[29] Melodic - ROS Wiki *http://wiki.ros.org/melodic.*

[30] Gazebo *http://gazebosim.org/.*