

UNIVERSIDAD CARLOS III DE MADRID  
ESCUELA POLITÉCNICA SUPERIOR



---

**Bachelor Thesis**

---

**UAV MISSION PLANNING: FULL AREA COVERAGE  
APPLICATION TO SAVE AND RESCUE**

September 2015

A bachelor thesis submitted in fulfillment of the requirements for the Degree  
in Aerospace Engineering

AUTHOR:

Diego Provencio Moreno

TUTOR:

Manuel Fernando Soler Arnedo



# Acknowledgements

First of all I would like to thank Manuel Soler Arnedo for propounding me the idea of this project and for putting me in contact with CENTUM SOLUTIONS where I have made a lot of really good friends.

Secondly, I would like to express my gratitude to CENTUM for its aid providing me with the necessary LIFESEEKER data and its encourage to develop this bachelor thesis.

Also, I would like to thank Manuel again, together with David Morante and Maite Arteta for their support through our weekly laboratory meetings.

Last but not least, I would like to thank my family who has always backed me.



# Abstract

Every year hundreds of people get lost in the mountains and other environments. In order to find them, aerial search and rescue missions are executed. The firm CENTUM has develop the LIFESEEKER system to aid the rescue teams to accomplish this task. LIFESEEKER is a system that can detect the telephone's signal of a missing person and send its exact location to the rescue teams. The goal of this project is to develop the algorithms needed to generate a path planning, for a RPAS which carries the LIFESEEKER, to fully cover the searching area where the missing person is. This objective is going to be reached by implementing an area coverage algorithm and a path planning algorithm for a benchmark problem and a real one. The more critical maneuvers will be proven in a test environment previously designed in order to see the flight performance when tracking the generated flight plan. Since the results show a good vehicle's performance, a real tool that controls this algorithms has been developed. The objective is to show how it should be the final application for its implementation in a real mission.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background information . . . . .	1
1.2	Socioeconomic environment . . . . .	3
1.3	Legal framework . . . . .	4
1.4	Project's focus . . . . .	4
1.5	Project objectives . . . . .	6
1.6	Methodology . . . . .	6
1.7	Project's planning . . . . .	7
1.8	Project's budget . . . . .	7
1.9	Document overview . . . . .	7
<b>2</b>	<b>State of the art</b>	<b>9</b>
2.1	RPAS solutions for search and rescue missions . . . . .	9
2.2	Area coverage alternatives . . . . .	12
2.2.1	Area coverage for a simple area . . . . .	12
2.2.2	Area coverage for real areas . . . . .	13
2.3	Path planning alternatives . . . . .	13
2.4	Test environment alternatives . . . . .	14
<b>3</b>	<b>Benchmark problem</b>	<b>15</b>
3.1	Problem definition . . . . .	15
3.2	Area coverage . . . . .	16
3.2.1	Area coverage problem statement . . . . .	16
3.2.2	Waypoint set definition . . . . .	16
3.3	Path planning . . . . .	18
3.3.1	Closed TSP statement . . . . .	18
3.3.2	Open TSP statement . . . . .	19
3.3.3	Genetic algorithms . . . . .	19
3.3.4	Waypoint sequence . . . . .	22
3.4	Test environment . . . . .	23
3.4.1	Helicopter modelling . . . . .	23
3.4.2	State-space model . . . . .	24
3.4.3	PID controller design . . . . .	25
3.4.4	Helicopter simulation . . . . .	30
<b>4</b>	<b>Real case of study</b>	<b>41</b>
4.1	Problem definition . . . . .	41
4.1.1	Universal Transverse Mercator background . . . . .	42
4.2	Area coverage . . . . .	42
4.2.1	Set cover problem statement . . . . .	43
4.2.2	Greedy algorithm . . . . .	43
4.2.3	Waypoint set definition . . . . .	45
4.3	Path planning . . . . .	45
4.4	Test environment . . . . .	46

---

<b>5</b>	<b>Commercial application</b>	<b>49</b>
5.1	An application for the real world . . . . .	49
5.1.1	Area definition . . . . .	50
5.1.2	Area coverage . . . . .	52
5.1.3	Path planning . . . . .	53
5.1.4	Log messages . . . . .	54
<b>6</b>	<b>Conclusions</b>	<b>55</b>
6.1	Future work . . . . .	56



# List of Acronyms

AUVSI: Association for Unmanned Vehicle System International  
BOE: Boletín Oficial del Estado  
CC: Creative Commons  
CC-BY-SA: Creative Commons Attribution Share-Alike  
CTSP: Close Travelling Salesman Problem  
GFDL: GNU Free Documentation License  
GNU: GNU's Not Unix  
GUI: Graphical User Interface  
ING: Instituto Nacional de Geografía  
MDT: Modelo Digital del Terreno (Digital Terrain Model)  
MTOW: Maximum Take-Off Weight  
NP: Non-deterministic Polynomial  
OTSP: Open Travelling Salesman Problem  
PID: Proportional Integral Derivative  
RPA: Remotely Piloted Aircraft  
RPAS: Remotely Piloted Aircraft System  
RS: Recommended Standard  
RX: Receiver  
SISO: Single Input Single Output  
TSP: Travelling Salesman Problem  
TX: Transmitter  
UAS: Unmanned Aircraft System  
UAV: Unmanned Aerial Vehicle  
US: United States  
USAF: United States Air Force  
UTM: Universal Transverse Mercator



# List of Figures

1.1	MQ-9 Reaper . . . . .	1
1.2	DJI Phantom 2 Vision . . . . .	2
1.3	Aircraft classification . . . . .	2
1.4	LIFESEEKER system . . . . .	5
1.5	LIFESEEKER system . . . . .	5
1.6	Gannt diagram . . . . .	8
2.1	Huginn X1 . . . . .	9
2.2	eBee . . . . .	10
2.3	Schiebel S-100 Camcopter . . . . .	10
2.4	Ambulance RPAS . . . . .	11
2.5	Raptor 90 SE Helicopter . . . . .	11
2.6	Close TSP example . . . . .	13
2.7	Open TSP example . . . . .	14
3.1	Simple area to be covered . . . . .	15
3.2	Hexagon pattern . . . . .	17
3.3	Circle area coverage . . . . .	17
3.4	X-Y representation of the waypoint position . . . . .	18
3.5	Flip operator between elements 2 and 8 . . . . .	20
3.6	Swap operator between elements 2 and 8 . . . . .	20
3.7	Slide operator with element 2 . . . . .	21
3.8	Genetic algorithm flowchart . . . . .	21
3.9	X-Y representation of the waypoint sequence for CTSP algorithm . . . . .	22
3.10	X-Y representation of the waypoint sequence for OTSP algorithm . . . . .	23
3.11	Block diagram of a PID controller in a feedback loop . . . . .	26
3.12	Block diagram of the PID waypoints tracking controller . . . . .	27
3.13	Example of the two phases of the guidance logic . . . . .	28
3.14	Set of coordinate systems . . . . .	30
3.15	Inertial X-Y representation of helicopter's trajectory in the close path planning . . . . .	31
3.16	Inertial Y-Z representation of helicopter's trajectory in the close path planning . . . . .	31
3.17	Inertial X-Y representation of helicopter's trajectory in the open path planning . . . . .	32
3.18	Inertial Y-Z representation of helicopter's trajectory in the open path planning . . . . .	32
3.19	Heading angle of helicopter in the closed path planning . . . . .	33
3.20	Heading angle of helicopter in the open path planning . . . . .	34
3.21	Inertial velocities of the helicopter between waypoints 1 and 2 . . . . .	35
3.22	Attitude angles between waypoints 1 and 2 . . . . .	35
3.23	Helicopter's controller inputs between waypoints 1 and 2 . . . . .	36
3.24	Helicopter's $z^I$ variation between waypoints 1 and 2 . . . . .	36
3.25	Inertial velocities of the helicopter during first turn in the open path planning . . . . .	37
3.26	Attitude angles of the helicopter during first turn in the open path planning . . . . .	38
3.27	Yaw rate of the helicopter during first turn in the open path planning . . . . .	38
3.28	Helicopter's controller inputs during first turn in the open path planning . . . . .	39

4.1	MDT200 Section of Madrid, Spain . . . . .	41
4.2	UTM division . . . . .	42
4.3	3D representation of the waypoints needed to cover the area . . . . .	45
4.4	3D representation of the open path planning . . . . .	46
4.5	3D representation of the close path planning . . . . .	46
5.1	GUI . . . . .	49
5.2	GUI Area definition panel . . . . .	50
5.3	GUI Map panel when a real map is introduced . . . . .	50
5.4	GUI Map panel when a simple map is introduced . . . . .	51
5.5	GUI UTM North-East map coordinates panel . . . . .	51
5.6	GUI Real map reduction panel . . . . .	51
5.7	GUI Map panel when real map is reduced . . . . .	52
5.8	GUI Area coverage panel . . . . .	52
5.9	GUI Map panel when the area coverage algorithm finish . . . . .	53
5.10	GUI Path planning panel . . . . .	53
5.11	GUI Map panel when the path planning algorithm finish . . . . .	54
5.12	GUI Log messages panel after several commands introduced by the user . . . . .	54

# List of Tables

1.1	LIFESEEKER specifications . . . . .	6
1.2	Project’s budget summary . . . . .	7
2.1	Raptor 90 SE key specifications . . . . .	12
4.1	Input covering matrix . . . . .	44



# Chapter 1

## Introduction

### 1.1 Background information

The term Unmanned Aerial Vehicle (UAV) refers to an unmanned aircraft, a flying vehicle without an onboard human pilot or passengers. As its name indicates, "unmanned" implies no direct human piloting, and therefore, its control may be either onboard (autonomously) or offboard (remotely controlled). Those UAV that are remotely piloted are called Remotely Piloted Aircraft (RPA). However, these terms are obsolete nowadays, and their are substituted by Unmanned Aircraft Systems (UAS) and Remotely Piloted Aircraft Systems (RPAS) respectively. The reason laying behind these new terms is to remark that we are refering to aircraft, which implies that airworthiness will need to be demonstrated, and moreover, to indicate that they are systems consisting of ground control stations, communications links, and launch and retrieval systems in addition to the aircraft itself.

According to ICAO [1], the categorization of the aircraft's type is as shown in Figure 1.3. Any of these aircraft can be an RPAS, but in particular, there are two main categories that have experience a big increase in this market: aeroplane and rotorcraft.

Aeroplanes, also called fixed-wing aircraft (see Figure 1.1), require both relative velocity for the generation of the necessary aerodynamic forces and a runway for landing and take-off (although launchers can be used also for takeoff). On the other hand, rotorcraft (see Figure 1.2), have unique flight capabilities such as the ability to move in all directions independently of its heading or their vertical flight capability (Vertical Takeoff and hover maneuvers).



Figure 1.1: By U.S. Air Force photo/Staff Sgt. Brian Ferguson (USAF Photographic Archives (image [permalink](#))) [Public domain], via Wikimedia Commons



Figure 1.2: By Capricorn4049 (Own work) [CC BY-SA 4.0 (<http://creativecommons.org/licenses/by-sa/4.0>)], via Wikimedia Commons

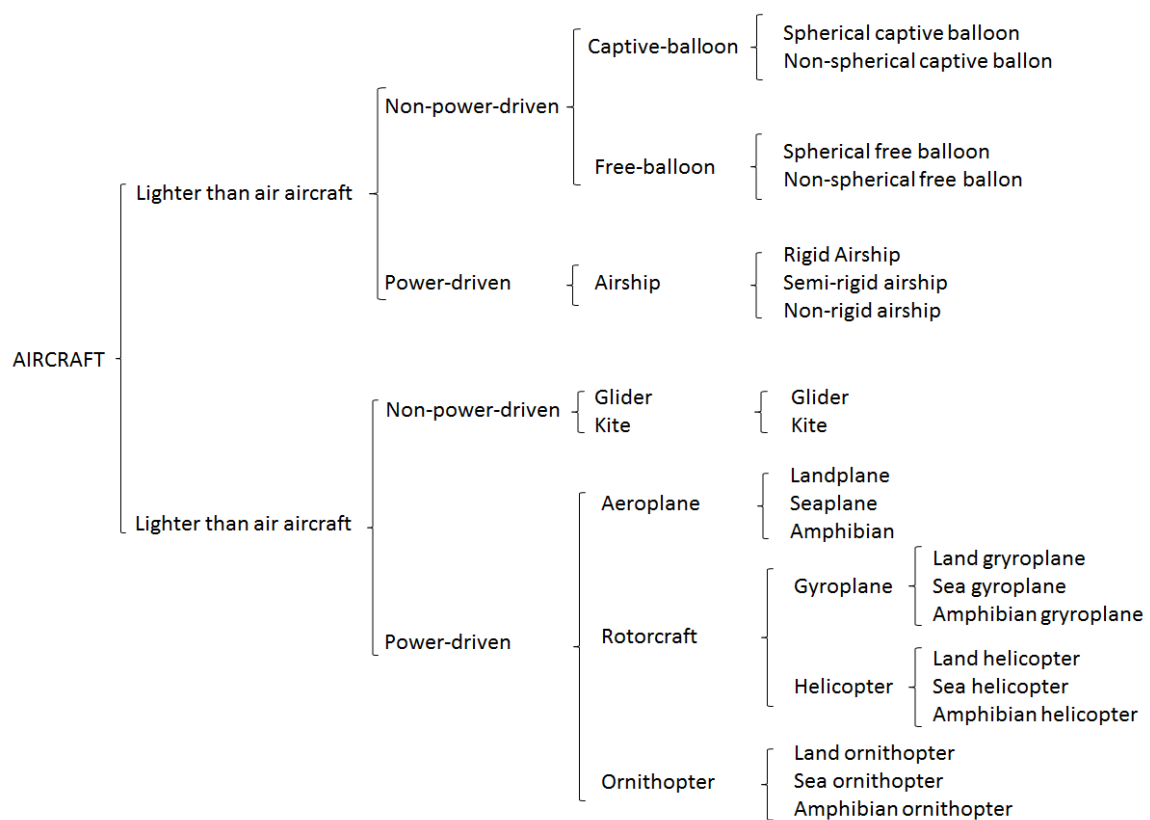


Figure 1.3: Aircraft classification



## 1.2 Socioeconomic environment

According to AUVSI<sup>1</sup>, the Economic Impact of Unmanned Aircraft Systems Integration in the United States report shows the economic benefit of RPAS integration:

- In the first three years of integration (2015-2017) more than 70,000 jobs will be created in the US. Consequently, this will have an economic impact of more than \$13.6 billion.
- By 2025, it is foreseen more than 100,000 jobs created and economic impact of \$82 billion.

Moreover, TEAL GROUP Corporation<sup>2</sup> calculates:

- A global RPAS procurement and R&D expenditures of \$14 billion in the year 2025 (\$ 4 billion in 2015): \$90 billion total in 10 years.
- The military research will add \$ 30 billion.
- The military market will suposse the 72 %.
- With 66% and 10% of the worldwide RPAS sales, the U.S. and Israel dominate the sector.
- The production of European countries, all together, does not represent more than 10 %.
- 35,000 RPAS will be produced worldwide in the next 10 years.
- The European market should experience the same growth trend but at lower scale.

Due to the development of the market many countries are starting to develop economic policies that take into account this emerging sector. For instance, the galician government is committed to invest 45 million of euros in the developing of the RPAS industry, making Galicia an European reference<sup>3</sup>.

This huge increase in the drone market incomes is due to the big expansion of the RPAS at very different sectors. Some of these sectors are:

- Aerial photography.
- Cartography.
- Agriculture.
- Forest services.
- Search and rescue.
- Cinema and extreme sports.
- Works monitoring.
- Security.
- Border surveillance.

And these are just few applications of this huge market, which is evolving so quickly that is not possible to predict what other applications could appear during the following years.

---

<sup>1</sup>AUVSI. *AUVSI's The Economic Impact of Unmanned Aircraft Systems Integration in the United States*. URL: <http://www.auvsi.org/auvsiresources/economicreport>

<sup>2</sup>Teal Group. *World Unmanned Aerial Vehicle Systems Market Profile and Forecast*. URL: <http://www.tealgroup.com/>

<sup>3</sup>La voz de Galicia. *Multinacionales como Airbus o Boeing se interesan por el plan de drones de la Xunta*. Online; accessed 28-August-2015. July 2015. URL: [http://www.lavozdeg Galicia.es/noticia/galicia/2015/07/18/multinacionales-airbus-boeing-interesan-plan-drones-xunta/0003\\_201507G18P9991.htm](http://www.lavozdeg Galicia.es/noticia/galicia/2015/07/18/multinacionales-airbus-boeing-interesan-plan-drones-xunta/0003_201507G18P9991.htm).

## 1.3 Legal framework

Due to the huge and quick expansion of the drone market, nowadays the current international legal framework is not able to regulate this sector. In an attempt to create documentation that enables a satisfactory international regulation, ICAO has stepped forward and has started developing its own regulation [1]. However, some countries have created their own legislation and relevant bylaws to enable operation of small RPAS within their territory

For instance, the Spanish government has implemented its own regulation [2]. This regulation is evolving with time due to the quick development of the industry and new laws are expected during the following months. It should be noticed that these laws only are applied to RPAS with MTOW below 150 kg because the regulation of these ones depends on each European nation, while for RPAS with higher MTOW it depends on Europe.

According to the BOE [2], aerial activities could be realized by civil RPAS at day and under visual meteorological conditions with subjection to the following requirements:

- RPAS with a maximum takeoff weight (MTOW) below 25 kilograms (see Figure 1.2) can operate in areas out of buildings in cities, villages or settled areas, in non controlled airspace inside the visual range of the pilot, at a distance from him not greater than 500 meters and at a height from floor not greater than 120 meters. However, it also specifies that these RPAS can operate ahead of visual range of the pilot with the appropriate certificate specified in this BOE.
- RPAS with a MTOW > 25kg (see Figure 1.1) used to fire fighting or search and rescue applications can operate with the conditions and limitations established at their airworthiness certificate in non controlled airspace

Nevertheless, although Spain and other countries as United States, Canada, Australia or Brazil has its own regulations to situate this sector in a legal framework, there is still the necessity of an international and stable legislation to let the market grows.

## 1.4 Project's focus

As it has been introduced in Section 1.2, there are a lot of possible applications in the RPAS's market that can be the object of study, and this project is going to be focused on one of them, the Search and Rescue applications with RPAS in Spain.

Every year hundreds of people get lost into the mountains and a huge percentage of them die. Only in Spain, according to Guardia Civil recordings<sup>4</sup>, in 2013, 893 interventions were done and 94 persons died. Furthermore, in 2014, 244 dinghy carrying 3421 persons were rescued in the middle of the sea, 13 died and 43 were not found according to *Salvamento Marítimo* [3].

The aim of this project is to contribute to the field of search and rescue missions by optimizing the path planning followed by the aerial vehicles that search for them. Nowadays this search is done by piloted helicopters such as MBB/Kawasaki BK117, but they have some limitations that could be overcome through the use of RPAS. For instance, it could be possible to search during night through different kind of sensor like infrared cameras or systems that searches the mobile signals of the missing person, or to fly with low visibility or under meteorological conditions that would make impossible any other kind of aerial missions. This translates into a tapping of the risks that these missions usually implies for pilots.

For this project, the vehicle used is going to be an RPAS carrying a system called LIFESEEKER, developed by CENTUM Research and Technology (see Figure 1.4). LIFESEEKER is an on-board system capable of locate the mobile phones of missing people and transmit their position to the authorities. This system, when installed in aerial platforms, allows an efficient and quicker search at remote zones and also provides a communication channel with the victim. This system is autonomous, so it can be perfectly integrated in an RPAS.

---

<sup>4</sup>Ministerio del Interior. *El Servicio de Montaña de la Guardia Civil ha rescatado a mas de 3.000 personas durante el 2013*. 29. Feb. 2014. URL: [http://www.interior.gob.es/prensa/noticias/-/asset\\_publisher/GHU8Ap6ztgsg/content/id/1636509](http://www.interior.gob.es/prensa/noticias/-/asset_publisher/GHU8Ap6ztgsg/content/id/1636509).



Figure 1.4: By: CENTUM Research and Technology ([http://centum.es/wp-content/uploads/LifeSeeker\\_EN.pdf](http://centum.es/wp-content/uploads/LifeSeeker_EN.pdf))

LIFESEEKER makes use of the huge social penetration that the communication technologies had experiment the last years, turning a mobile phone into a radar beacon (see Figure 1.5), which is able to guide the rescue teams to the exact position of the victim. But, as it is expected, this system has a limited radio-range coverage (see LIFESEEKER specifications<sup>5</sup> in Table 1.1) to detect the radio-beacon, so the project's task is to generate an appropriate path planning that minimizes the distance travelled by the vehicle while ensuring a 100 % coverage of the searching area.



Figure 1.5: By: CENTUM Research and Technology (<http://www.iqube.es/en/>)

<sup>5</sup>CENTUM Research and Technology. *LIFESEEKER brochure*. URL: <http://www.centum-rt.com/media/images/pdf/LIFESEEKER-CRT-EN.pdf>

Technical Specifications	
Rx RF channels	4
Tx RF channels	1
RS-232 ports	2
Size	143x168x215mm
Weight	<8kg
Power	80W
Transmission power	4W
Frequency range	890-960 MHz
Radio range coverage	4000m

Table 1.1: LIFESEEKER specifications

## 1.5 Project objectives

The main objective of this project is to develop the appropriate algorithms to generate path plannings for RPAS carrying the LIFESEEKER or other similar system that minimize the distance travelled while ensuring 100% of coverage of the searching area. But, in order to achieve this goal, several objectives have to be achieved first:

1. Generate the path planning for a benchmark problem with a simple area to be covered. For that, several sub-objectives should be accomplished.
  - Solve an area coverage problem in order to find the minimum set of waypoints that fully cover the area.
  - Optimize the path planning in order to reach all the previously found waypoints while the distance is minimized.
  - Develop a test environment in order to see the performance of an RPAS following the generated path planning.
2. Generate the path planning for a real problem where the area to be covered is a real Spanish topography. For that, the same sub-objectives as for the benchmark problem have to be reached.
3. Finally, develop a real application for these algorithms to be used in a search and rescue mission through the implementation of a GUI.

## 1.6 Methodology

The methodology used through the project starts with a research on the different RPAS alternatives, area coverage methods, path planning algorithms and test environments that could be implemented to test the resulting flight trajectories.

Afterwards, a bottom-up development will be implemented by starting from a benchmark problem and finishing with a real case of study. For that, the benchmark problem is defined and solved through the implementation of an area coverage and a path planning algorithm. The resulting flight plan will be tested in a developed test environment to see the performance of the vehicle when following that flight plan.

Once the path planning is validated, the real case of study will be defined and solved following the same steps as in the benchmark problem. This problem will have a real scenario to be covered. To develop the path planning, another kind of area coverage algorithm will be implemented.

Finally a GUI will be created to automatize the path planning development process introduced during the project. The purpose of this GUI is to show the main aspects that the real application in search and rescue missions should contain.

## 1.7 Project's planning

The time management is also an important issue to ensure an appropriate evolution of the project. In order to make a good planning a Gantt diagram has been used, where the planned working hours are showed against the real ones. This diagram is presented in Figure 1.6.

## 1.8 Project's budget

The budget associated with this project is divided into working hours, hardware and software used.

The working hours dedicated to the realization of this project were around around 20 hours per week during 22 weeks (see Table 1.6), that lead to an amount of 440 hours. If we consider the remuneration salary to be 10 euros per hour, the total cost is 4400 euros.

The hardware used is a laptop Asus K555ld-Xx668h value in 699 euros; and the software that has been used is Matlab and Simulink R2012b with an student version which cost 124 euros.

A summary of the project's budget is shown in Table 1.2.

Total budget			
Working hours (euros)	Hardware (euros)	Software (euros)	TOTAL (euros)
4400	699	124	5223

Table 1.2: Project's budget summary

## 1.9 Document overview

The document is structured in the following way:

- Chapter 2 introduces the state of art.
- Chapter 3 develops the algorithms used to fully cover a simple area, introduces the algorithms used to generate the waypoint sequence minimizing the distance traveled, and a test environment to test the path planning is developed. All these algorithms are applied to a benchmark problem and tested with the test environment to see the quality of the designed flight plan.
- Chapter 4 develops the same process as in Chapter 3 for a real scenario with a much more complex area.
- Chapter 5 shows the developed Graphical User Interface for a real application in a real search and rescue mission.
- Chapter 6 provides the project's conclusions and future work.

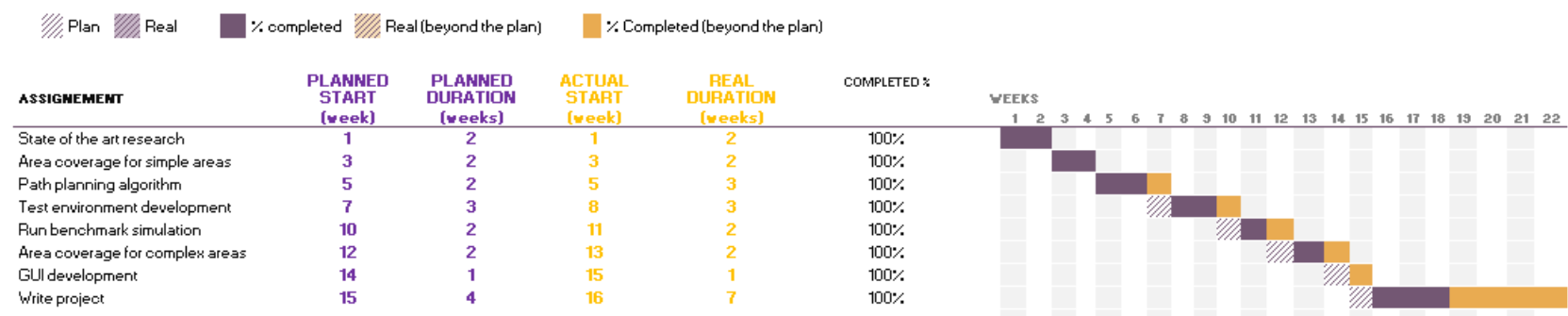


Figure 1.6: Gannt diagram

## Chapter 2

# State of the art

### 2.1 RPAS solutions for search and rescue missions

Nowadays, there is no similar solution to the LIFESEEKER system developed by CENTUM, but there are other kind of RPAS solutions that had already contributed to search and rescue operations. Through this section, different cases of RPAS being used in search and rescue missions will be presented. These cases that are going to be described are just a small selection of the whole branch of cases.

Danoffice IT, has developed a commercial rotorcraft solution for disaster purpose. This vehicle was used for instance at the typhoon Yolanda at Tacloban (Philippines), where it aided on the planning of the operation site, and on the identification of which roads were usable after the disaster [4].



Figure 2.1: By: Sky-Watch (<http://sky-watch.dk/what-we-do/products/huginn-x1/emergency-management/>)

Furthermore, Another RPAS was used at this disaster to capture aerial photographs of downtown Tacloban<sup>1</sup>. This vehicle was a fixed wing drone called eBee and their images contributed on the generation of the most detailed and up-to-date maps of the region. These maps were used by different humanitarian organizations and even by the Filipino Government for the planning of the relief efforts.

---

<sup>1</sup>Patrick Meier. *Humanitarians in the Sky: Using UAVs for Disaster Response*. Online; accessed 01-September-2015. June 2014. URL: <http://irevolution.net/2014/06/25/humanitarians-in-the-sky/>



Figure 2.2: By senseFly (<http://www.geo-matching.com/products/id2184-ebec.html>)

Regarding RPAS helicopters, the Schiebel S-100 Camcopters has been also used to scan the Mediterranean sea to identify migrants travelling onboard of precarious boats, allowing the rescue teams to react faster<sup>2</sup>.



Figure 2.3: By User:Stahlkocher (Own work) [GFDL (<http://www.gnu.org/copyleft/fdl.html>) or CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>)], via Wikimedia Commons

Moreover, although it is not still at the market, the Delft University of Technology<sup>3</sup> has developed an ambulance rotorcraft that is able to transport an automated external defibrillator, medication, cardiopulmonary resuscitation aids or other things that could be needed to save the life of the victim before the emergency teams are able to arrive to the zone.

---

<sup>2</sup>UAS Vision. *Schiebel Camcopter on Refugee Mission in Mediterranean Again*. May 2015. URL: <http://www.uasvision.com/2015/05/04/schiebel-camcopter-on-refugee-mission-in-mediterranean-again/>

<sup>3</sup>Webredactie Communication. *TU Delft's ambulance drone drastically increases chances of survival of cardiac arrest patients*. Online; accessed 29-August-2015. Oct. 2014. URL: <http://www.tudelft.nl/en/current/latest-news/article/detail/ambulance-drone-tu-delft-vergroot-overlevingskansbij-hartstilstand-drastisch/>.





Figure 2.4: By Delft University of Technology (<http://www.io.tudelft.nl/onderzoek/delft-design-labs/applied-labs/ambulance-drone/>)

As it can be seen, there are a lot of different RPAS that are being used in search and rescue applications. For the purpose of testing the path planning that is going to be developed, it would be highly recommended to use a vehicle that has already been employed within this rescue field. However, obtaining the necessary data to carry out the study has not been possible. For that reason an alternative RPAS is used for the development of the case study.

The selected helicopter is the Raptor 90 SE (see Figure 2.5), which is one of the largest radio control helicopter in the market. As it can be seen from specifications<sup>4</sup> Table 2.1, this helicopter could not carry the LIFESEEKER due to the limitations in their MTOW, although it could be used for the proof of concept and to give an insight about the path planning to be accomplished.



Figure 2.5: Raptor 90 SE Helicopter By HighTechScience.org ([http://www.hightechscience.org/raptor\\_90\\_se\\_helicopter.htm](http://www.hightechscience.org/raptor_90_se_helicopter.htm))

---

<sup>4</sup>Raptor. *Raptor 90 SE parts list*. URL: [http://www.raptor.com.tw/parts\\_list/4891TW-K10/Thunder%20Tiger/1\\_4.html](http://www.raptor.com.tw/parts_list/4891TW-K10/Thunder%20Tiger/1_4.html)

Full fuselage length	1.410m
Full fuselage width	0.190m
Total height	0.476m
Main rotor diameter	1.605m
Tail rotor diameter	0.260m
Gear ratio(main rotor:engine:tail rotor)	1:8.45:4.65
No-load weight	5.45kg
MTOW	11kg
Power source	Nitro fuel

Table 2.1: Raptor 90 SE key specifications

The main reasons of using an helicopter instead of a fixed wing aircraft is that as it was explained in Chapter 1, the LIFESEEKER system is able to transmit the exact position of the missing person once it detects the signals of his mobile phone. Moreover, it is able to establish a communication link between this mobile phone and the ground control station of the RPAS. In order to do that it has to maintain a distance with the missing person that does not overcome the range of the system. Then, it can be deduced that a rotorcraft is going to have an operational advantage at this point because of its ability to hover and land and take-off vertically. In this way the vehicle can land near the victim and allows the communication between the ground control station and the victim.

In addition as it has been exposed, there are rotorcrafts that are able to carry first aid equipment, and by landing near the victim, provide him with them. In this way, the victim can survive more time before the rescue teams reach them. Then, this is another operational advantage of rotorcrafts against fixed-wing.

In particular, the selection of an helicopter instead of another rotorcraft is due to its higher energy efficiency.

## 2.2 Area coverage alternatives

As it was establish in Chapter 1, two different kind of scenarios are going to be studied, a simple theoretical area (a 2-D plane) and a real one. With the purpose of finding the minimum needed waypoints to fully cover both areas, different approaches are going to be used.

### 2.2.1 Area coverage for a simple area

As it has been explained, the LIFESEEKER coverage can be modeled as an sphere of a determined radius. The vehicle is going to be set to fly always at the same altitude to simplify the algorithms and fulfills always comply with the Spanish regulation of flying below 120 meters above the terrain. Then, the intersection of both, the plane to be covered and the sphere, will lead to circles with a radius,  $R$ . Then, the problem to be solved is to determine the minimum number of circles with equal radius that fully cover the searching area.

The optimal solution of this problem has been found for a concrete number of circles [5], [6], [7]. But these solutions cannot be generalized for any dimensions of the plane or the circle's radius. Due to this, in order to provide a solution that can find a nearly optimal solution for any case it has been decided to implement a pattern to find the position of the waypoints.

One of the most important related work in the field of coverage of unbounded areas with patterns is Kershner's works [8], who proved in 1939 that the best way to cover unbounded areas with circles is to use the honeycomb structure (also known as triangular lattice) and circumscribe the circles to the hexagons.

Nevertheless, it needs to be noticed that the areas to be covered are bounded, as the area where the people get lost are delimited by the emergency teams. Then, some corrections should be done near the boundaries to optimize the number of waypoints needed to fully cover the area. For that reason, the work done in [9] about how to cover bounded rectangular areas will be applied. This paper presents that the best pattern to fit a rectangle applying the honeycomb structure is placing the first vertical row of hexagon such that just its most left vertex lay out of the rectangle, and the other two vertex at the left side coincide with the left side of the rectangle. Moreover the second vertical row of hexagons must be placed so that the lowest one has its lower edge coinciding with the lower side of the rectangle.

### 2.2.2 Area coverage for real areas

For real areas, digital models of the terrain are going to be used. This models generates a mesh of points with the  $x,y,z$  position of terrain's elements. For this case, due to the complexity of the area, the patterns previously described for a simple area cannot be used, but again the flying altitude,  $z$ , is going to be set to a constant value over the terrain.

Due to the enormous amount of waypoints to be treated, they will be reduced to a small set that still allows to cover the whole area. For that instance, a mesh of possible waypoints is going to be established above the terrain mesh, with the same mesh length of the digital model of the terrain, so that the error due to the set reduction will be lower for meshes with high accuracy.

Now, using the model of the sphere for the LIFESEEKER coverage, each possible waypoint covers a set of points of the terrain mesh, and there is a set of waypoints that can cover all the points. Then, this problem can be described as the set cover problem.

The set cover problem is an NP-hard<sup>5</sup> problem and there are many algorithms that can be used for solving it. For instance, there are exact algorithms as the ones described in [10] and [11], but this kind of solution methods requires a huge computational cost. Furthermore, there are heuristics algorithms, which are able to find a near optimal solution in a reasonable time. Inside the heuristics algorithms, it can be found different techniques as the greedy algorithms [12], simulated annealing [13] or natural network algorithms[14] among others. Nevertheless, due to the simplicity to be implemented, a greedy algorithm has been used for solving the set cover problem.

## 2.3 Path planning alternatives

In the selection of the sequence to be followed to reach all the waypoints selected by solving the area coverage problem, two approaches are going to be studied:

- The vehicle covers the whole area starting at one point and finishing at another point.
- The vehicle covers the whole area starting and finishing at the same point.

The first scenario is going to have less distance to be cover by the vehicle, but it only will work under the assumption that the vehicle has enough battery/fuel to cover the area and ultimately return to the ground station from any waypoint after the last one has been reached. On the other hand, the second scenario covers the possibility that the RPAS has not enough battery/fuel to perform the trajectory of the first case. In this way, although the vehicle covers more distance, there is no chance to lose it due to lack of endurance.

According to these requirements the planning problem that fulfills them is the well known Travelling Salesman Problem (TSP). The TSP is NP-hard and no general method of solution is known. An example of the Travelling Salesman Problem (Open and closed) is given in Figure 2.6 and 2.7.

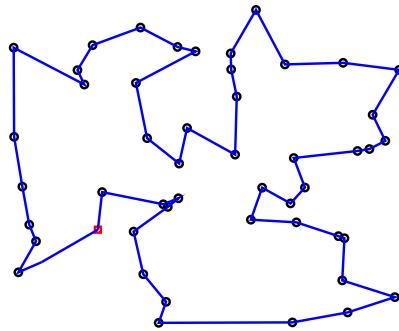


Figure 2.6: Close TSP example

<sup>5</sup>Weisstein, Eric W. "NP-Hard Problem." From MathWorld—A Wolfram Web Resource. URL: <http://mathworld.wolfram.com/NP-HardProblem.html>

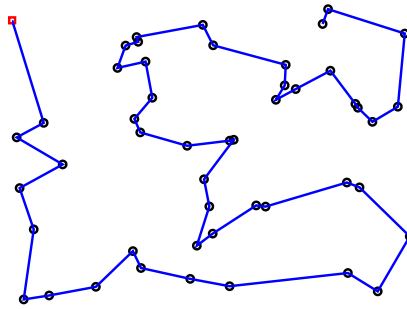


Figure 2.7: Open TSP example

As for the set covering problem, there are exact and heuristic methods to solve the Travelling Salesman Problem. Nevertheless, for the same reasons as for the set covering problem, an heuristic method is going to be implemented.

Inside the heuristics algorithms there is a branch of possibilities to solve the TSP problem but, in particular, it has been decided to use genetic algorithms. The reason is that although they may not find the best solution, they can find a near perfect solution for up to 100 waypoints in less than one minute [15]. At search and rescue missions, time is crucial, so this feature of the genetic algorithms has been found to be quite interesting as it adds value to the solution. In addition, the number of waypoints to be managed is going to be small, so the algorithm can perfectly find the optimal solution.

## 2.4 Test environment alternatives

One of the objectives of this project is to elaborate a test environment to check the suitability to perform the developed flight plans. This could be done through one of the available flight simulators in the market, as for instance the Flight Gear Simulator, which is the most extended open-source simulation software in the market or through the X-plane simulator produced by Laminar Research.

On one hand, Flight Gear is a multiplatform simulator that was created with the purpose of being configurable by the user and give him more freedom of the currently provided by commercial software. In addition, as is open-source, it is possible to see how the software works inside, what makes it a good option for learning. Furthermore, it is possible to interact with Flight Gear with Simulink (which is the program used for its development).

On the other hand X-plane allows the user to customize and modify their aircraft models to reach the desired flight characteristics. Furthermore, it allows to import and export data in real time, a feature that is not available in Microsoft Flight Simulator.

Nevertheless, none of this simulators is going to be used. The main reason is that it has been found more interesting to start a new test environment to increase the *know how* of the firm CENTUM and start the way to generate a personal test environment for this company.

Then, in order to generate our own test environment to track the path planning that is going to be generated, a waypoint tracking controller should be developed. This controller could be linear or non-linear. A non-linear controller would lead to more reliable results since the helicopter model does not need to be linearized, but this type of controller are much more difficult to implement.

For real-life applications, it is common to start with the less complex approach, in this case the linear controller. Inside the linear controllers there is again a branch of possibilities, but with the objective of implement first the most simple approach, a controller composed of four SISO (Single Input Single Output) PID (Proportional Integral Controller) feedback loops is going to be used. This kind of controller is a very common start point in real life design applications, since it does not require knowledge of the helicopter model and the gains of the controller are tuned empirically.

## Chapter 3

# Benchmark problem

### 3.1 Problem definition

The objective of this Chapter is to develop the algorithms needed to elaborate a path planning for the detection of a missing person, for a particular simple case, in order to expose the algorithms and test the resulting path.

The first model of the area is a flat plane (see Figure 3.1) which will emulate a topography where the changes in altitude are small. Although this area is very simple, it can be a good approximation for a lot of real areas such as a desert, a tableland or the sea, where the changes in altitude are small.

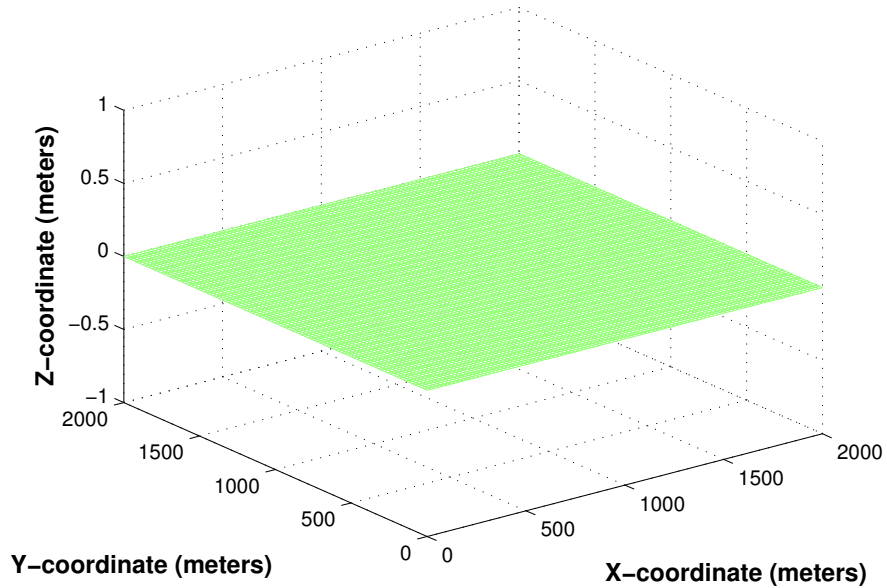


Figure 3.1: Simple area to be covered

Due to the limitations of the Raptor 90 SE, the area to be covered is a surface with dimensions [2000m x 2000m]. The radio-range coverage of the LIFESEEKER system is assumed to be 500m in order to be proportional to the area used. Finally, the flying altitude of the vehicle is set to 100 meters over the surface.

Summarizing, the problem data is:

- Flying altitude,  $z=100\text{m}$
- Searching area dimensions,  $[w,h]=[2000,2000\text{m}]$
- Radio-range coverage,  $R_{system}=500\text{m}$

## 3.2 Area coverage

Once the problem parameters has been defined, it is time to develop the area coverage algorithm. Its objective is to find the minimum set of waypoints that are necessary to fully cover the searching area with the radio-range coverage of the used system. It is very important to remark the requirement of "fully cover", this is the one of the key elements of this project. If there are some non explored areas, it is possible that the missing person is there and the whole mission would fail. Furthermore, the false negative could lead the rescue teams think that the missing person is not there, increasing the time needed to find it.

As it was explained in Chapter 2, the area coverage problem to be solved is to determine the minimum number of circles with equal radius that fully cover the searching area. For the particular case under study, the radius of the circles is:

$$R = \sqrt{R_{system}^2 - z^2} = 489.8979 \text{ m} \quad (3.1)$$

### 3.2.1 Area coverage problem statement

According to [16], this area coverage problem can be defined as follows: Consider a set of waypoints  $S = \{s_1, s_2, s_3, \dots, s_n\}$  in a 2D plane, all with the same covering range  $r$ . The target area  $A$  to cover is considered to be a rectangle  $A = [0, w] \times [0, h]$  of width  $w$  and height  $h$ . The position of each waypoint is given by the coordinates  $(x_i, y_i)$

**Definition 1.** A point  $(x, y) \in A$  is covered by a waypoint  $s_i$  if  $\sqrt{(x_i - x)^2 + (y_i - y)^2} \leq r$ . The target area  $A$  is  $k$ -covered by the waypoints  $S$  if each point  $(x, y) \in A$  is covered by at least  $k$  ( $k \geq 1$ ) different waypoints.

Then, the problem to be solved is "Find the smallest number of waypoints  $S_n(w, h)$  that should be used to achieve  $k$ -coverage ( $k \geq 1$ ) for a rectangular area of sizes  $w$  and  $h$  with waypoints of the same radius coverage".

### 3.2.2 Waypoint set definition

In order to solve the problem defined in Section 3.2.1, the presented work in Section 2.2 will be applied. Recalling the description presented in that section, the rectangle should be covered with hexagons going up and right until the whole area is covered.

In addition, to fulfill the corrections of the boundaries, the center of the first hexagon should be placed at the line of the lowest side of the rectangle and at a distance  $d$  from the left side of the rectangle as it is shown in Figure 3.2:

$$d = R \cdot \sin\left(\frac{\pi}{6}\right) \quad (3.2)$$

In addition, it should be noticed that, although we want to minimize the wasted area at the four edges, it is only possible to do it for two edges, the other ones will depend on the dimensions of the rectangle.

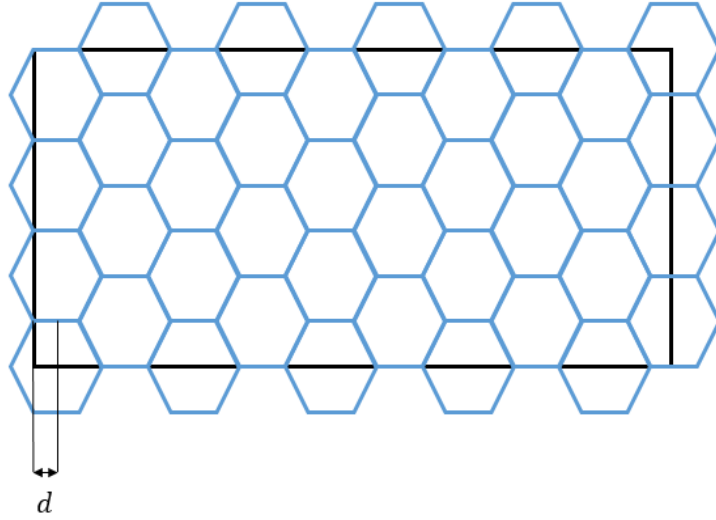


Figure 3.2: Hexagon pattern

Once the hexagons have been placed, the last step is to circumscribe the circles to the hexagons as it is shown in Figure 3.3. As it can be observed the whole area has been covered as it was required in Section 3.1 and the area wasted due to overlapping has been minimized thanks to the pattern used.

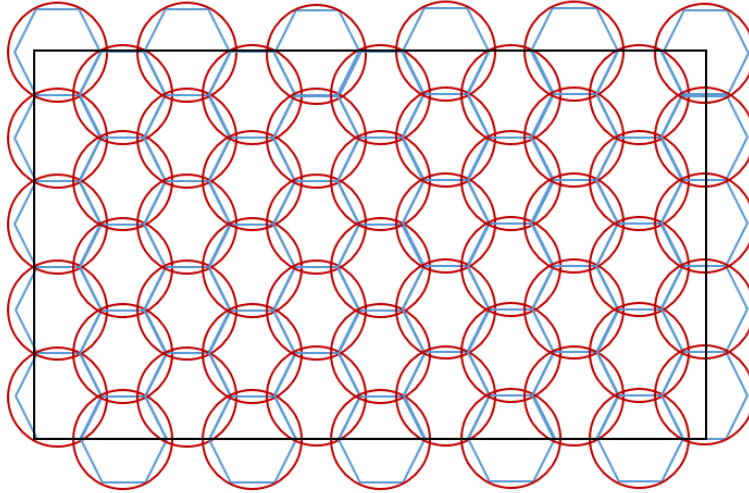


Figure 3.3: Circle area coverage

At this point, the method to select the set of waypoints needed to fully cover a simple area with a vehicle carrying the LIFESEEKER has been defined, meaning that it can be implemented for the case under study. The resulting waypoints after implement the area coverage method for the area defined in Section 3.1 are:

$$waypoints = [x, y, z] = \begin{bmatrix} 244.9 & 0 & 100 \\ 1714.6 & 0 & 100 \\ 979.8 & 424.3 & 100 \\ 244.9 & 848.5 & 100 \\ 1714.6 & 848.5 & 100 \\ 979.8 & 1272.8 & 100 \\ 244.9 & 1697.1 & 100 \\ 1714.6 & 1697.1 & 100 \\ 979.8 & 2121.3 & 100 \end{bmatrix} \quad (3.3)$$

These waypoints are shown in Figure 3.4. The number at each waypoint corresponds to the index of each element in Equation 3.3.

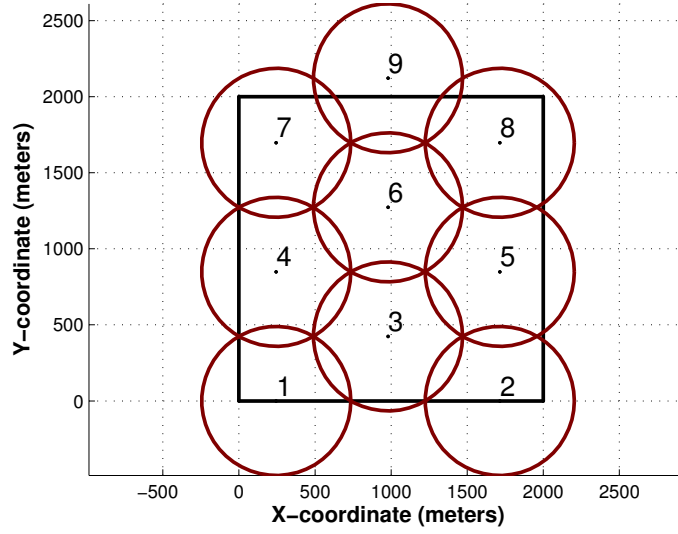


Figure 3.4: X-Y representation of the waypoint position

### 3.3 Path planning

Once the set of waypoints needed to fully cover the study area has been established, it is time to find the desired sequence to reach all of them while the distance travelled is minimized. For that, as it was described in Chapter 2, two different approaches are going to be implemented. These two problems will be named OTSP (Open Travelling Salesman Problem) and CTSP (Close Travelling Salesman Problem), respectively.

#### 3.3.1 Closed TSP statement

This problem ask the following question: *Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?*<sup>1</sup>

Let  $I$  be the set of waypoints with  $S$  points, including the starting waypoint, and let be  $d_{ij}$  be the distance between waypoints  $i$  and  $j$ . Moreover, let  $x_{ij}$  be a binary variable that is 1 if waypoint  $j$  is reached right after waypoint  $i$ , or 0 if not. Then, the TSP can be formulated mathematically in the following way:

$$\min \sum_{i \in I} \sum_{j \in I} d_{ij} \cdot x_{ij} \quad (3.4)$$

subject to:

$$\sum_{j \in I} x_{ij} = 1 \quad i \in I \quad (3.5a)$$

$$\sum_{i \in S} x_{ij} = 1 \quad j \in I \quad (3.5b)$$

$$\sum_{i \in S} \sum_{j \in S} x_{i,j} \leq |S| - 1 \quad S \subset I, |S| \geq 2 \quad (3.5c)$$

$$x_{ij} \in \{0, 1\} \quad i \in I, j \in I \quad (3.5d)$$

<sup>1</sup>Wikipedia. *Travelling salesman problem* — Wikipedia, The Free Encyclopedia. Online; accessed 13-September-2015. 2015. URL: [https://en.wikipedia.org/w/index.php?title=Travelling\\_salesman\\_problem&oldid=680796550](https://en.wikipedia.org/w/index.php?title=Travelling_salesman_problem&oldid=680796550)



The objective function 3.4 minimizes the total distance traveled. Constrains 3.5a and 3.5b ensure that the salesman arrives and leaves each city exactly once. Constraint 3.5c assures that there are no sub-tours, i.e. the route must be coherent. Finally, constrain 3.5d secures that the variables are binary.

### 3.3.2 Open TSP statement

According to [17], this problem can be described as follows: Consider a mission with  $n$  waypoints, including the starting waypoint. Subindex  $i$  and  $j$  refer to waypoints and take values between 2 and  $n$ , while index  $i=1$  refers to the starting waypoint. Also there is a shortest distance  $d_{ij}, i, j = 1, 2, \dots, n$  associated with the each pair of waypoints and also with the each waypoint and the starting one. The goal is to find a shortest route for a vehicle with unlimited capacity that reach all the waypoints so that the route ends after the final waypoint is reached. Mathematical programming formulation of OTSP requires two type of variables: the binary variables  $x_{ij}, i, j = 1, 2, \dots, n$  with a following notation:  $x_{ij} = 1$  if waypoint  $i$  precedes waypoint  $j$  in a route of the vehicle and  $x_{ij} = 0$  otherwise. Further on, the variables  $u_i, i = 2, 3, \dots, n$  represents the sequence value of the point  $i$ . The mathematical model for OTSP is:

$$\min \sum_{i=1}^n \sum_{j=1}^n d_{ij} x_{ij} \quad (3.6)$$

subject to

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 2, 3, \dots, n \quad i \neq j \quad (3.7a)$$

$$\sum_{j=2}^n x_{ij} \leq 1 \quad i = 2, 3, \dots, n \quad i \neq j \quad (3.7b)$$

$$\sum_{j=2}^n x_{1j} = 1 \quad i = 1, 3, \dots, n \quad (3.7c)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad i, j = 2, 3, \dots, n \quad i \neq j \quad (3.7d)$$

$$x_{ij} \in \{0, 1\} \quad i, j = 1, 2, \dots, n \quad i \neq j \quad (3.7e)$$

Equation 3.6 models the total distance covered. Constrains 3.7a ensures that the RPAS goes to all waypoints and constrain 3.7b ensures that the vehicle does not need to depart from every waypoints since the path ends after going to the last one. Constraint 3.7c ensure that the helicopter starts its travel exactly once and constrain 3.7d avoid sub-paths. The last constrain, 3.7e, secures that the variables are binary.

### 3.3.3 Genetic algorithms

Genetic algorithms are a search technique used to find good solutions to optimization problems and they are part of the evolutionary algorithms. The evolutionary algorithms try to simulate nature's evolutions, using techniques inspired of it, such as heredity, mutation, selection and crossover. The genetic algorithms used to solve the Travelling Salesman Problems are the ones developed by Joseph Kirk for the open<sup>2</sup> and closed<sup>3</sup>. These algorithms are based on the most general genetic algorithm, so let's first explain this one. For that, a few terms have to be explained before:

- Individual: is a possible solution to the problem. It is an string with the sequence of waypoints. Each individual is represented by a chromosome.
- Population: is a set of individuals.
- Fitness: is a characteristic of an individual that measures how good the solution of that individual is.

<sup>2</sup>Joseph Kirk. *Fixed Start Open Traveling Salesman Problem - Genetic Algorithm TSP*. Retrieved February 2015. 2008. URL: <http://www.mathworks.com/matlabcentral/fileexchange/21198-fixed-start-open-traveling-salesman-problem-genetic-algorithm>

<sup>3</sup>Joseph Kirk. *Travelling Salesman Problem - Genetic Algorithm TSP*. Retrieved February 2015. 2007. URL: <http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm>

- **Genes:** is the set of waypoints in the string of the individual.

A general genetic algorithm works as shown in Algorithm 1.

---

**Algorithm 1:** General genetic algorithm
 

---

1. **Initialization of population:** An initial (random) sampling of chromosome is generated.
  2. **Fitness:** Evaluate the fitness of each chromosome in the population.
  3. **New population:** Generate a new population by repeating the following steps until the new population is complete:
    - (a) **Natural selection:** Select two parent chromosome from a population according to their fitness.
    - (b) **Crossover:** The genes of these chromosomes are combined through crossover to form a new offspring. If no crossover is performed, offspring is an exact copy of parents
    - (c) **Mutation:** The genes of the new offspring are mutated.
    - (d) **Accepting:** Place the new offspring in a new population.
  4. **Replace:** Use new generated population for further run of the algorithm.
  5. **Evaluate:** If the end fitness condition is satisfied, Stop, and return the best solution in current population. If the condition is not fulfilled, return to step 2.
- 

Once the most general one has been described, the differences between this one and the used one are going to be explained:

- The main difference is that the used algorithm does not use crossover operator. This is because the author states that "*Crossover operator tends to be quite destructive (it makes large changes to a given route) and therefore rarely improves a decent solution*"<sup>3</sup>. This behaviour has been attributed by [18] to the feature that this operator rarely produces a valid tour when the search space is very extensive.
- The other difference is that this algorithm, instead of looking to fulfill a determined fitness condition, it performs the number of iterations that you desire according to your computational requirements.

The steps followed by this algorithms are described in Figure 3.8. The mutation operators used in this algorithm works in the following way:

- **Flip:** flip mutation takes a segment of genes of the individual and flips them (like a mirror).

```

1 2 3 4 5 6 7 8 9
      ↓
1 8 7 6 5 4 3 2 9
  
```

Figure 3.5: Flip operator between elements 2 and 8

- **Swap:** swap mutation exchanges the positions of 2 randomly selected elements from the individual. All other elements remain in their current positions.

```

1 2 3 4 5 6 7 8 9
      ↓
1 8 3 4 5 6 7 2 9
  
```

Figure 3.6: Swap operator between elements 2 and 8

- **Slide:** moves the position of one gen to another random position.

1 2 3 4 5 6 7 8 9  
↓  
1 3 4 5 6 7 8 2 9

Figure 3.7: Slide operator with element 2

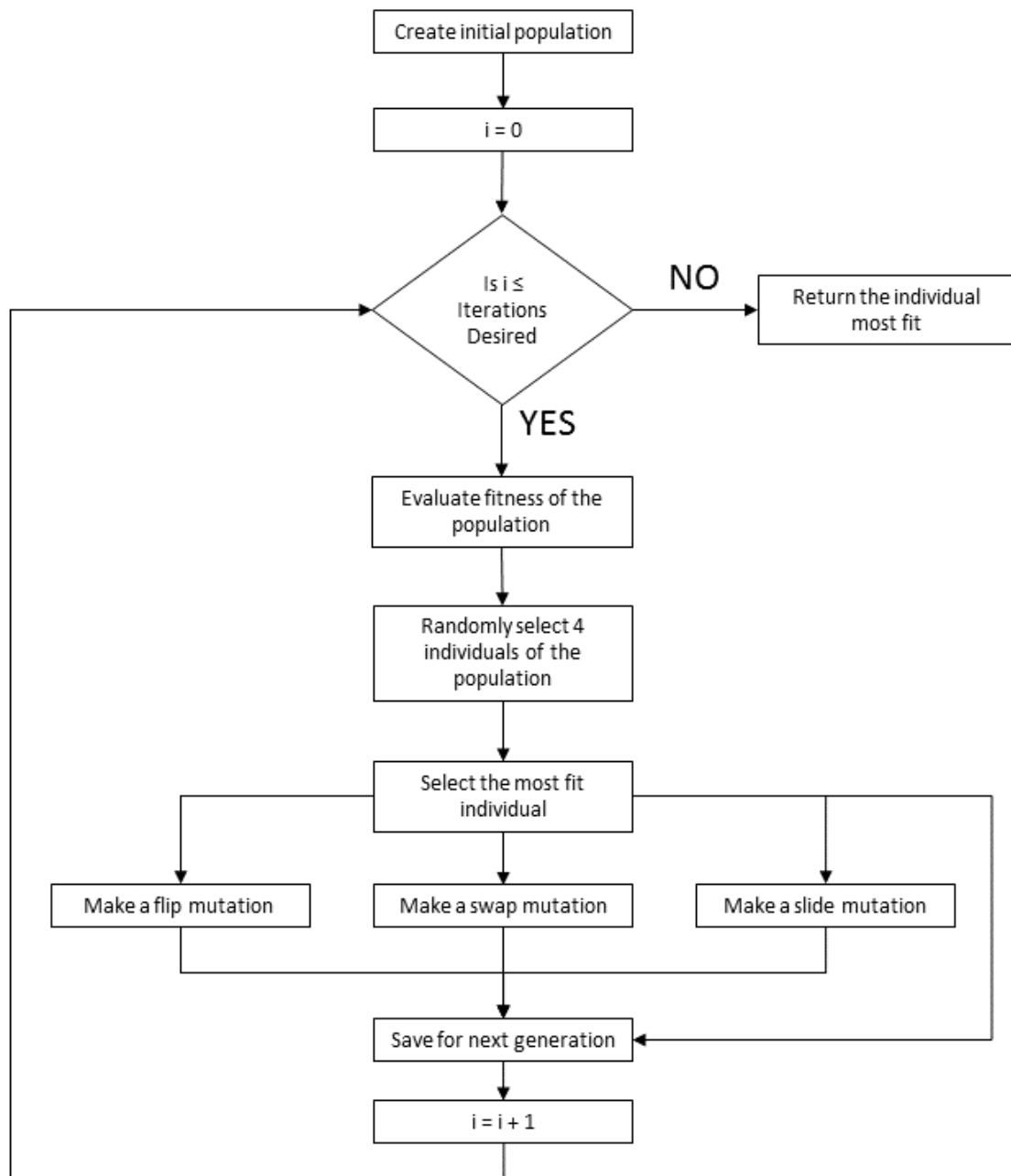


Figure 3.8: Genetic algorithm flowchart

### 3.3.4 Waypoint sequence

Once the method used has been defined, it can be applied to the set of waypoints obtained in Section 3.2.2. Both, open and close path planning are going to be obtained.

#### Close path planning

The sequence provided by the CTSP algorithm, taking as reference the waypoints defined in equation 3.3, is:

$$CTSP = [1 \ 3 \ 2 \ 5 \ 8 \ 6 \ 9 \ 7 \ 4 \ 1] \quad (3.8)$$

To find this solution, 1000 iterations and a population size of 1000 individuals has been used. The computational time used is 5.25 seconds and the total distance covered is 7636.75 meters. This sequence is illustrated in Figure 3.9. The red square in the figure represents the first waypoint.

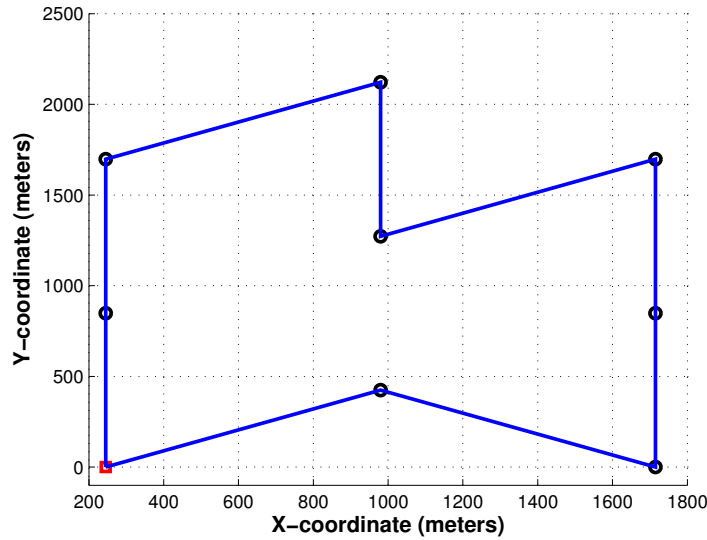


Figure 3.9: X-Y representation of the waypoint sequence for CTSP algorithm

#### OTSP Path Planning

The sequence of waypoints provided by the OTSP algorithm according to waypoints defined in equation 3.3 for an initial population size of 1000 individuals and 1000 iterations is:

$$OTSP = [1 \ 4 \ 3 \ 2 \ 5 \ 8 \ 9 \ 7 \ 6] \quad (3.9)$$

This sequence is illustrated in Figure 3.10. This algorithm takes a computational time of 5.56 seconds and the total covered distance is 6788.22 meters.

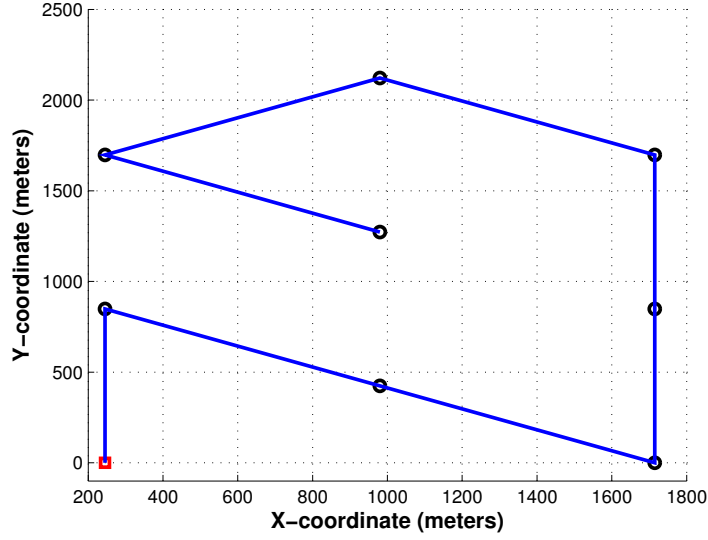


Figure 3.10: X-Y representation of the waypoint sequence for OTSP algorithm

### 3.4 Test environment

As it was defined in Chapter 2, the test environment is going to be composed of a waypoint tracking controller with a linear model of the Raptor 90 SE. It would be interesting that this controller could be used for other small-scale helicopter, because, as it was explained, the Raptor 90 SE cannot perform a real search and rescue mission. For that, the linear state space model has to capture the dynamic behaviour of a wide family of small-scale helicopters. With that purpose, the used model is going to be based on the parametrized model developed by Mettler in [19]. This model has been proved successfully for different small-scale helicopter controllers [20] [21] [22] [23]. Mettler's model, provides a generalized and physically meaningful solution to developing practical linear models for small-scale helicopters.

#### 3.4.1 Helicopter modelling

The basic linearized equations of motion for the helicopter's dynamics model are derived from the Newton-Euler equations for a rigid body with 6 degrees of freedom. The external forces are represented in stability derivative form. The relation between the body and internal axes is defined in appendix 6.1.

Following the work in [19], the equations of motion of a small-scale helicopter are derived and categorized in the following way.

##### Lateral and longitudinal fuselage dynamic equations

The translational and angular fuselage equations of motion are derived from the Newton-Euler equations:

$$\dot{u} = (w_0 q + v_0 r) - g\theta + X_u u + X_a a \quad (3.10)$$

$$\dot{v} = (-u_0 r + w_0 p) - g\phi + Y_v v + Y_b b \quad (3.11)$$

$$\dot{p} = L_u u + L_v v + L_b b \quad (3.12)$$

$$\dot{q} = M_u u + M_v v + M_a a \quad (3.13)$$

##### Heave dynamics

The heaving dynamics are defined in the following way:

$$\dot{w} = (-v_0 p + u_0 q) + Z_w w + Z_{col} u_{col} \quad (3.14)$$

### Yaw dynamics

The corresponding equations that define the yaw dynamics are:

$$\dot{r} = N_r r + N_{ped}(u_{ped} - r_{fb}) \quad (3.15)$$

$$\dot{r}_{fb} = -K_{r_{fb}} r_{fb} + K_r r \quad (3.16)$$

Where  $r_{fb}$  is the yaw rate feedback which represents an artificial yaw damping system that is used during flight tests to obtain the stability and control derivatives.

### Rotor/stabilizer-bar dynamics

The rotor equations of motion for the longitudinal (a) and lateral (b) flapping angles are:

$$\tau_f \dot{a} = -a - \tau_f q + A_b b + A_{lat} u_{lat} + A_{lon} u_{lon} \quad (3.17)$$

$$\tau_f \dot{b} = -b - \tau_f p + B_a a + B_{lat} u_{lat} + B_{lon} u_{lon} \quad (3.18)$$

And the stabilizer-bar equations for the longitudinal (c) and lateral (d) flapping angles are:

$$\tau_s \dot{c} = -c - \tau_s q + C_{lon} u_{lon} \quad (3.19)$$

$$\tau_s \dot{d} = -d - \tau_s p + D_{lat} u_{lat} \quad (3.20)$$

Where  $\tau_f$  and  $\tau_s$  are, respectively, the time constants of the rotor and the stabilizer bar.

### 3.4.2 State-space model

The state-space model needed to develop the PID controller is going to be based on the previously described equations of motion. In particular, the linear state space model developed in [24] is going to be used for the development of this project. The main reason for using it, is that, it has been already adapted for the Raptor 90 SE and the corresponding stability and control derivatives are already identified (see appendix 6.1).

This model represents the dynamic response of the vehicle perturbed state space vector from the reference flight condition. For this particular model, the reference flight condition is hover, whose equilibrium state variables are:

$$u_0 = v_0 = w_0 = p_0 = q_0 = r_0 = \theta_0 = \phi_0 = 0 \quad (3.21)$$

Although, theoretically, the model is limited to the neighborhood of a certain operation condition, in reality, it covers a relative wide area of the flight envelope. This was proved by comparing this model against a non-linear one in [24].

The state space model is:

$$\dot{x} = Ax + Bu_c \quad (3.22)$$

where  $x$  and  $u_c$  are:

$$x = [u, v, \theta, \phi, q, p, a, b, w, r, \psi] \quad (3.23)$$

$$u_c = [u_{lon}, u_{lat}, u_{col}, u_{ped}] \quad (3.24)$$

where A and B are:

$$A = \begin{bmatrix} X_u & 0 & -g & 0 & 0 & 0 & X_a & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & g & 0 & 0 & 0 & Y_b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ M_u & M_v & 0 & 0 & 0 & 0 & M_a & 0 & 0 & 0 & 0 \\ L_u & L_v & 0 & 0 & 0 & 0 & 0 & L_b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1/\tau_f & A_b & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & B_a & -1/\tau_f & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & Z_a & Z_b & Z_w & Z_r & 0 \\ 0 & N_v & 0 & 0 & 0 & N_p & 0 & 0 & N_w & N_r & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (3.25)$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A_{lon} & A_{lat} & 0 & 0 \\ B_{lon} & B_{lat} & 0 & 0 \\ 0 & 0 & Z_{col} & 0 \\ 0 & 0 & N_{col} & N_{ped} \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.26)$$

And the four controllers are:

- $u_{lon}$  = Main rotor longitudinal cyclic control command
- $u_{lat}$  = Main rotor lateral cyclic control command
- $u_{col}$  = Main rotor collective control command
- $u_{ped}$  = Tail rotor longitudinal control command

The model's order can be increased by adding the dynamics of the stabilizer bar and the yaw damping system. These subsystems give additional damping to the angular velocity dynamics ( $p$  and  $q$ ), but they constitute just additional sources of feedback of the angular dynamics, their presence does not influence the controller design [24]. Therefore their effect has been omitted.

### 3.4.3 PID controller design

Once the helicopter's model has been defined, the waypoint tracking controller that is going to be used to test the elaborated flight plan of previous sections can be developed. As it was explained in Chapter 2, this controller is going to be composed of four SISO PID feedback loops, so let's start by explaining the main features of a PID controller.

#### PID Theorecial background

PID controllers are control loop feedback mechanisms that calculate an error value as the difference between the desired set point and the actual one of a selected measurable variable. The controller attempts to minimize the error by adjusting the process through the use of a manipulated variable.

The PID controller has three separate constant parameters that acts in parallel to reduce the error: the proportional (P), the integral (I) and the derivative (D). The proportional generates a correction to the actual error, the integrator produce a correction proportional to the error integral, and the derivative part anticipates the error through the derivative of the error and generates a correction to reduce it in the future.

A scheme of a PID controller is shown in Figure 3.11.

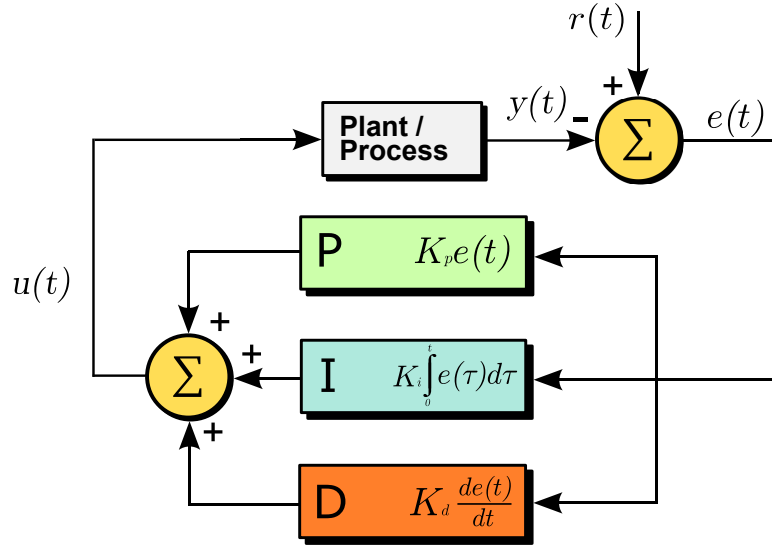


Figure 3.11: By TravTigerEE (Own work) [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons

### PID Controller Design

The design of cyclic feedback loops is based on the fact that the longitudinal and lateral velocity of the helicopter is produced from the pitch and roll tilt of the fuselage. Therefore the helicopter velocity will be proportional to the attitude of the helicopter [19].

The structure of the proposed controller can be observed in Figure 3.12 and is based on the one proposed by Ioannis and Kimon in [24]. It should be notice that this controller does not take into account the cross coupling effect that the helicopter could have but as it was explained in Chapter 2, this cross coupling is not a big issue for the Raptor 90 SE.

The SISO controller is formed by four closed loop which are independent of each other. Three of them are designated to control the helicopter towards a desired x,y,z position and the other one control the heading of the helicopter during that motion. Then, there are two different inputs, the required position,  $p_r^I$  and heading,  $\psi_r$ , that are controlled by the guidance logic block.

The inner loop commands the attitude angles of the helicopter by setting the desired angles  $\theta_{des}$  and  $\phi_{des}$ . This command is compared with the actual attitude of the helicopter and after that, they are multiplied by their corresponding gains to give the adequate input to the main rotor longitudinal and lateral cyclic controllers. These controllers produces the longitudinal and lateral tilting of the main rotor disc.

$$u_{lon} = -K_\theta \cdot (\theta - \theta_{des}) \quad (3.27)$$

$$u_{lat} = -K_\phi \cdot (\phi + \phi_{des}) \quad (3.28)$$

As it can be seen, instead of implement the difference between  $\phi$  and  $\phi_{des}$  as for  $\theta$  and  $\theta_{des}$ , they are added. This is because in the outer loop, if the vehicle wants to move in the y-direction, the difference between  $p_{y,r}^I$  and  $p_y^I$  is negative, and then,  $\psi_{des}$  is going to have the opposite sign of the one actually needed. Then by adding  $\phi$  and  $\phi_{des}$ , we are actually performing its difference.

$$u_{lon} = -K_\theta(\theta - K_x e_x^B - K_u e_u^B) \quad (3.29)$$

$$u_{lat} = -K_\phi(\phi + K_y e_y^B + K_v e_v^B) \quad (3.30)$$



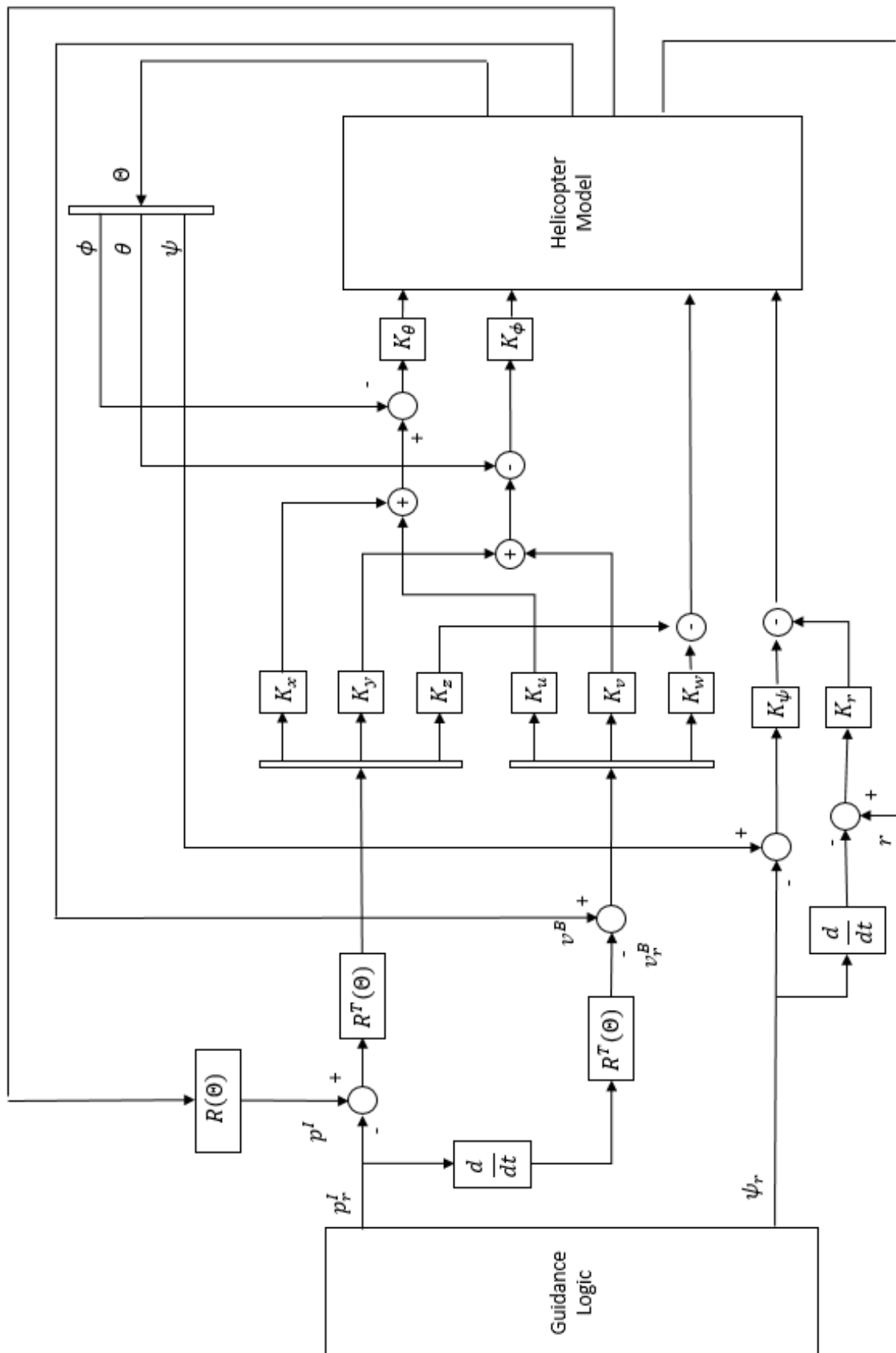


Figure 3.12: Block diagram of the PID waypoints tracking controller

The outer loop is responsible for generating the desired attitude angles needed in the inner loop. As it was explained before, these angles are proportional to the position and velocity errors. The controllers of this loop are  $u_{ped}$  and  $u_{col}$  and commands the thrust produced by the tail rotor generating a yaw moment on the helicopter and the rotor collective that controls the thrust of the main rotor. The tail rotor pedal control and the main rotor collective control command loops are more direct than the previous ones. The collective is only proportional to the z position and velocity and the pedal is only proportional on the yaw angle and rate.

$$u_{ped} = -K_{\psi}e_{\psi}^B - K_re_r \quad (3.31)$$

$$u_{col} = -K_z e_z^B - K_w e_w \quad (3.32)$$

Finally, the gains that make the controller to be stable are tuned manually until a good response of the controller is obtained. During the tuning, it should be taken into account that the controller should reduce the attitude error faster than the transitional one in order to operate fine. For that reason, the values should be chosen taking into account the different time scaling. Finally, the gains that lead to a good controller response are:

- $K_{\theta}=0.7566$
- $K_{\phi}=0.3252$
- $K_{\psi}=3$
- $K_r=0.35$
- $K_x=0.3256$
- $K_y=0.3252$
- $K_z=1.6018$
- $K_u=0.1628$
- $K_v=0.2493$
- $K_w=0.6060$

The remaining element of the controller to be explained is the guidance logic block. The purpose of this block is to identify the desired position and heading command that need the controller as input. These position and heading are going to be different depending whether the vehicle needs to make a rectilinear flight or a turn, then, the guidance logic block is going to feed the controller with different inputs depending on the flight maneuver that is needed to be implemented.

Summarizing, the guidance laws introduced by these block are going to be different for the two flight maneuvers to be executed:

- Rectilinear flight
- Turning maneuver

The aircraft is supposed to start in hover in the first waypoint with the needed heading to reach the second waypoint. Then, it makes a rectilinear path towards the second waypoint accelerating until it reaches the desired maximum velocity. When it reaches the waypoint, it performs the turn towards the next waypoint until it has the desired heading. At that point it continues with a rectilinear flight, and so on, until the mission is finished. An example of these two phases is showed in Figure 3.13.

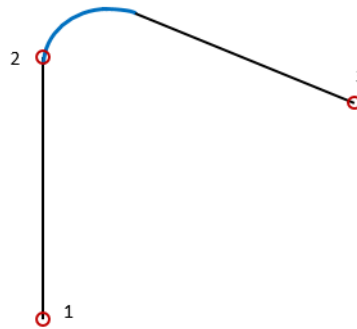


Figure 3.13: Example of the two phases of the guidance logic

Whenever a segment starts, the final conditions of the previous flight are the starting conditions of the next one (from now on, those reference conditions will be designated with the subindex "0"). Particularly  $\psi_0$  and  $p_0$  are the heading and position (x,y,z) at which the vehicle finishes the last guidance phase.

Due to this piece-wise function for the reference trajectory and heading, there are points of discontinuity that can lead to instantaneous transient jumps in the control inputs. To avoid it and improve the performance of the helicopter, the reference conditions can be processed by an appropriate low pass filter that attenuates the high frequency components of the signal, nevertheless, it has been observed that these discontinuities do not degrade too much the helicopter's performance, so it has not be implemented.

### Rectilinear flight

At this phase, the objective is to accelerate the vehicle towards the desired velocity or keep it if the vehicle already has it. Then the following exponential function will be implemented as the trajectory guidance law for desired position,  $p_r$ , during rectilinear flight:

$$p_r = p_0 + \begin{bmatrix} v_0 + (v_{max} - v_0) \cdot (1 - e^{-0.3 \cdot (t-t_0)}) \cdot \cos(\psi_r) \\ v_0 + (v_{max} - v_0) \cdot (1 - e^{-0.3 \cdot (t-t_0)}) \cdot \sin(\psi_r) \\ z_0 \end{bmatrix} \quad (3.33)$$

Where  $v_{max}$  is the maximum velocity magnitude that the vehicle is able to reach in order to minimize the time to find the missing person. Furthermore,  $v_0$  is the magnitude of the inertial velocity at previous phase:

$$v_0 = \sqrt{v_{x,0}^I + v_{y,0}^I + v_{z,0}^I} \quad (3.34)$$

Being  $v^I$  the components of the inertial velocity. During this rectilinear flight the required heading is going to be constant to the one at which the previous segments ends, designated as  $\psi_0$ .

$$\psi_r = \psi_0 \quad (3.35)$$

Finally, It should be noticed that  $z_r$  has been set constant because all the waypoints for the simple area have the same altitude.

### Turn maneuver

The turning maneuvers are going to be elaborated so that the vehicle makes an uniform circular motion. Then, the desired heading will be:

$$\psi_r = \psi_0 + \Omega \cdot (t - t_0) \quad (3.36)$$

Being  $\Omega$  the angular velocity of the turn, which is an input introduced by us that will be set to the maximum that the vehicle features allows with the corresponding  $v_{body}$ . And  $\psi_0$  is the heading at which the vehicle starts the turn.

On the other hand, to describe the turn, we are going to set two set of coordinates more, one in which its axes are parallel to the inertial axes and its origin is the actual position of the vehicle when it starts the turn ( $O_{x_A, y_A, z_A}$ ); and the other one is going to have same origin but its y-axes y going to be parallel to its heading ( $O_{x_B, y_B, z_B}$ ) (see Figure 3.14).

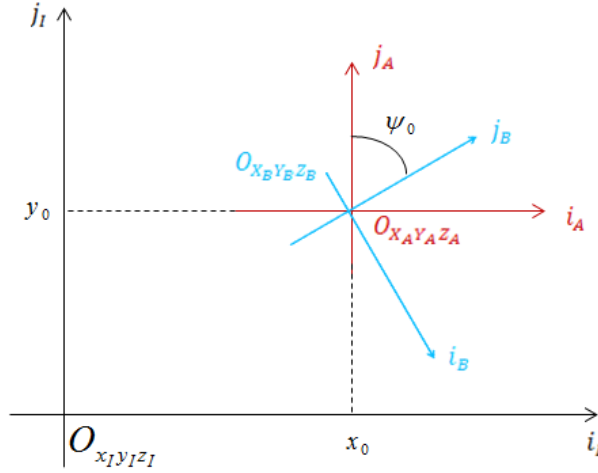


Figure 3.14: Set of coordinate systems

Then, the desired position of the vehicle with respect to the second set reference frame, when the turn is clockwise:

$$p_r^B = p_c + \begin{bmatrix} -R \cos(|\Omega| \cdot (t - t_0)) \\ R \sin(|\Omega| \cdot (t - t_0)) \\ 0 \end{bmatrix} \quad (3.37)$$

And when the turn is counterclockwise:

$$p_r^B = p_c + \begin{bmatrix} R \cos(|\Omega| \cdot (t - t_0)) \\ R \sin(|\Omega| \cdot (t - t_0)) \\ 0 \end{bmatrix} \quad (3.38)$$

Being  $p_c$  the turning center position, that will be  $[R, 0, 0]$  for the clockwise turn and  $[-R, 0, 0]$  for the counterclockwise turn.

Then,  $p_r$  with respect to  $O_{x_A, y_A, z_B}$  will be:

$$p_r^A = \begin{bmatrix} \cos(\psi_0) & \sin(\psi_0) & 0 \\ -\sin(\psi_0) & \cos(\psi_0) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot p_r^B \quad (3.39)$$

Finally, the desired position with respect to the inertial reference frame will be:

$$p_r = p_r^A + \begin{bmatrix} x_0 \\ y_0 \\ 0 \end{bmatrix} \quad (3.40)$$

### 3.4.4 Helicopter simulation

Once the test environment has been defined, the case under study can be simulated, but first, some parameters need to be defined. These parameters are the maximum velocity and the angular velocity of the turns:

- $v_{max} = 20 \text{ m/s}$
- $\omega = 0.5236 \text{ rad/s}$

The simulation has been done for the open and close path planning develop in previous sections. The trajectory followed by the vehicle for both plans can be seen in figures 3.15-3.18.

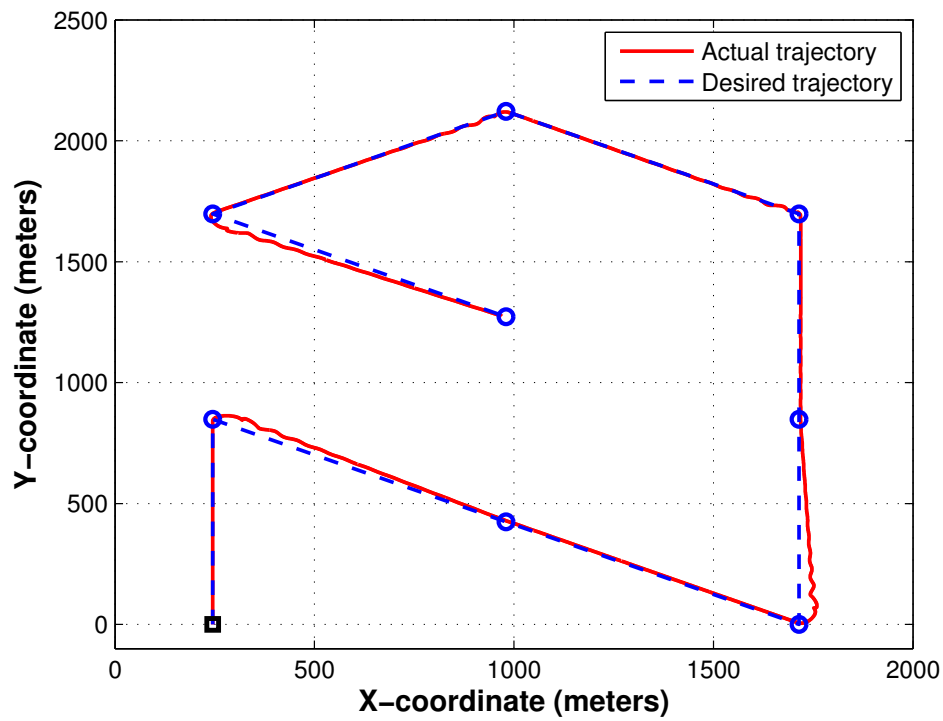


Figure 3.15: Inertial X-Y representation of helicopter's trajectory in the close path planning

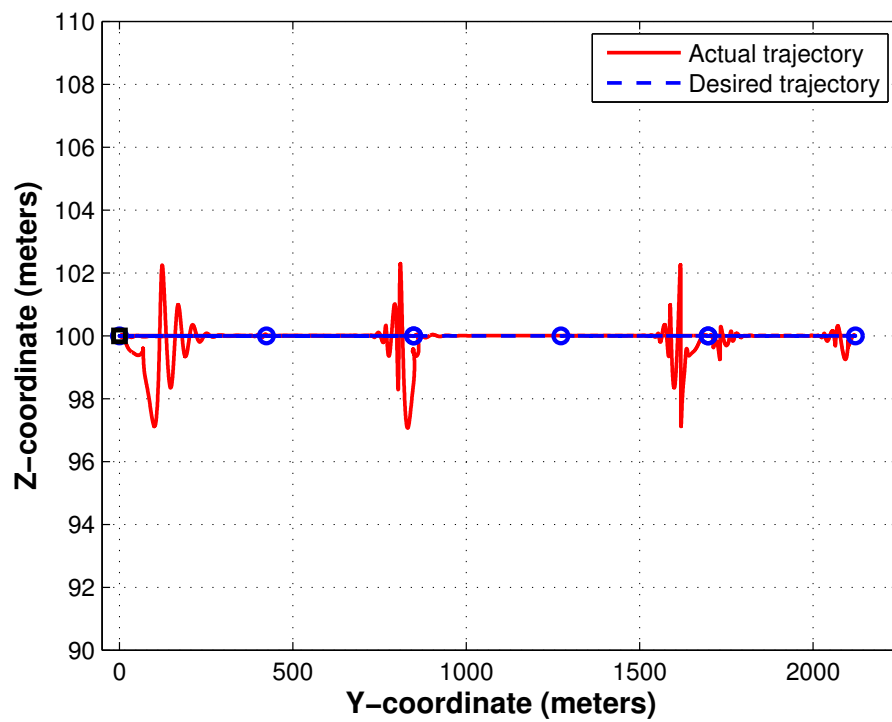


Figure 3.16: Inertial Y-Z representation of helicopter's trajectory in the close path planning

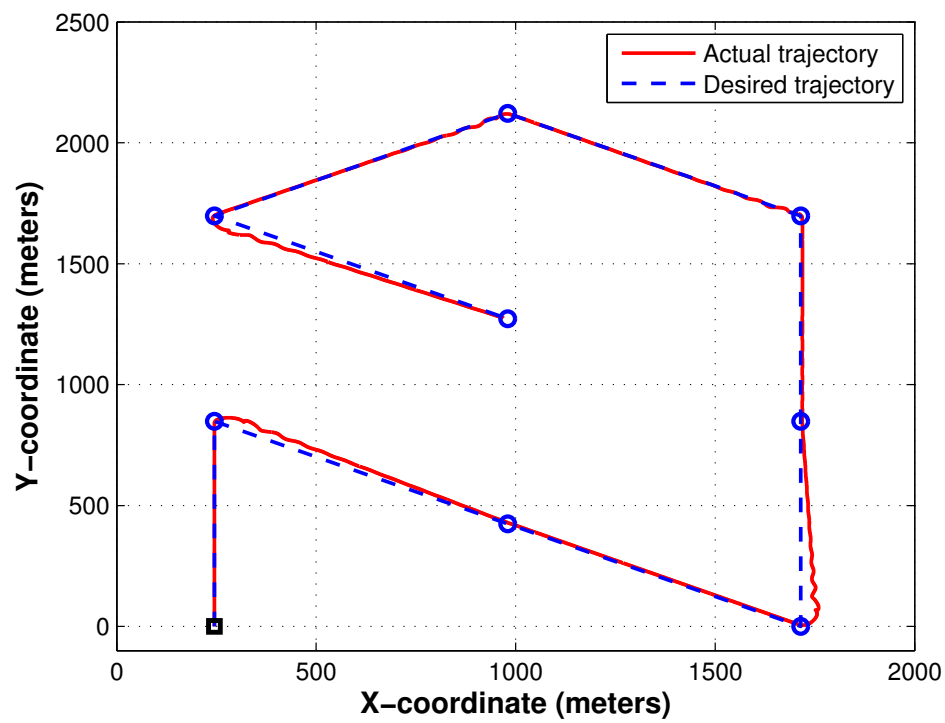


Figure 3.17: Inertial X-Y representation of helicopter's trajectory in the open path planning

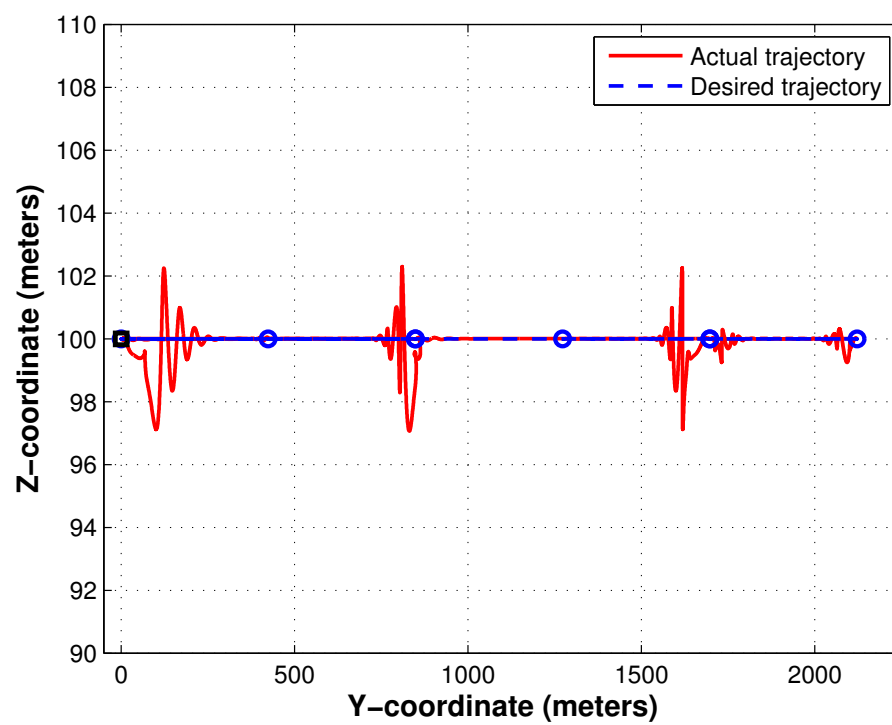


Figure 3.18: Inertial Y-Z representation of helicopter's trajectory in the open path planning

The time needed to perform both trajectories is 6 min and 23 s and 5 min and 40 s for the closed and open ones, respectively. As it could be expected, the one with lower distance to be travelled is the faster one. As can be observed from the times, the needed time is very low to cover an area of  $4 \text{ km}^2$ , and since the major time comes from the rectilinear flights it could be approximated that the time increases linearly with the travelled distance. Then, by the application of these algorithms the time needed to fully cover an area could be reduced significantly, since the main objective of the TSP is to minimize the distance travelled.

As can be seen from y-z representation in figures 3.16 and 3.18, there are some disturbances in the change of stages, this is produced due to the non-linearity of the guidance logic. But, since this disturbance is very small (around 2 meters from the desired altitude), it does not cause huge changes in the x-y trajectory.

Moreover, as can be seen from figures 3.15 and 3.17, there are some turns where the vehicle needs a higher radius to perform them. This is due to the abrupt change in heading that makes the helicopter to need more time to adapt its actual heading to the new one. Then, as the desired x-y trajectory depends on the heading, it makes that the vehicle performs a higher turn. In addition, the z-disturbance previously presented also degrade the vehicle's turn.

The changes in the heading can be observed in figures 3.19 and 3.20. As can be seen the helicopter adapts perfectly its heading to the desired one.

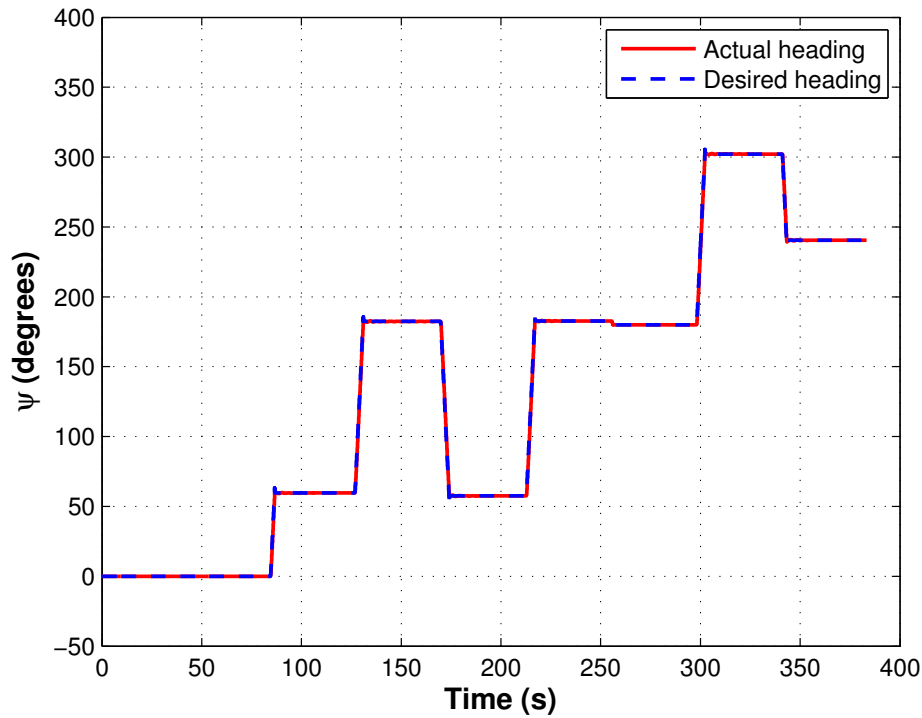


Figure 3.19: Heading angle of helicopter in the closed path planning

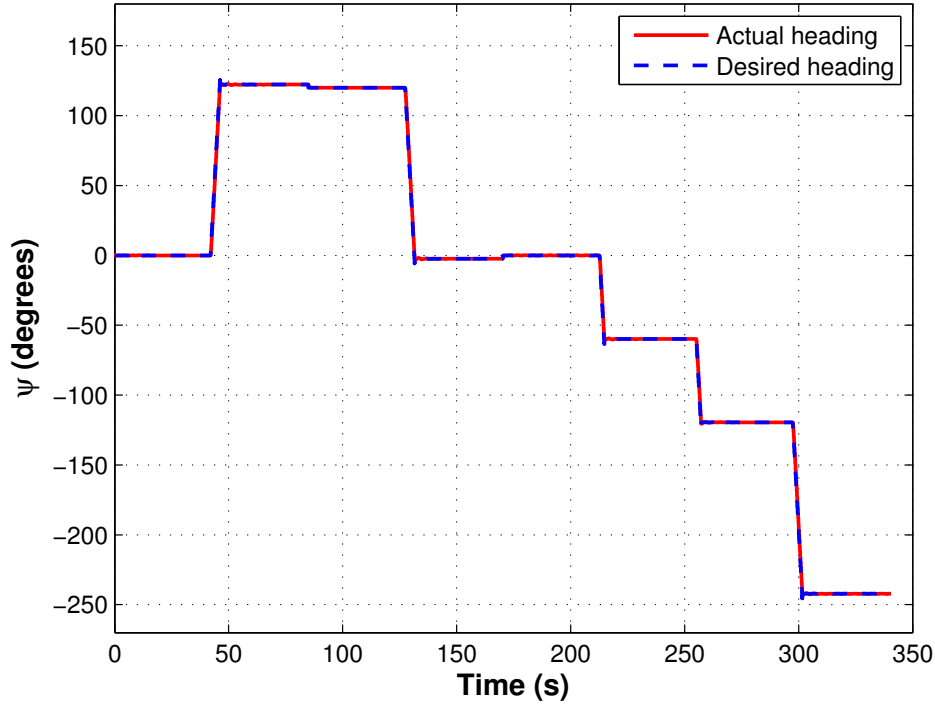


Figure 3.20: Heading angle of helicopter in the open path planning

To have a deeper knowledge of the helicopter performance during these trajectories, the first acceleration phase (that is equal for both path planning) and the first turn of the open path planning are going to be studied more deeply.

### Acceleration

One of the most important parameter to be focused during the first acceleration phase is the velocity. The reason is that the helicopter has to adapt it from zero to the desired one and it has to be done smoothly. This variable it is shown in Figure 3.21. As it can be seen, this requirement is fulfilled due to the guidance logic previously explained. Since the desired input velocity is set to be exponential, this avoid an abrupt change that could make the controller unstable. This variation in the velocity is produced by a change in the angle  $\theta$  that is shown in Figure 3.22.

Looking at the controllers Figure 3.23, it should be notice that for representation it has been normalized with its maximum experiment during the path planning. As can be seen from this figure, this change in the position is produced by variation in the longitudinal cyclic control that produces the needed inclination in the main rotor to move forward.

Moreover, a small variation in the altitude is produced. This is due to the change in the thrust vector produced by longitudinal cyclic control, which makes that the collective must be increased to overcome this variation and maintain the altitude (see Figure 3.24).

Although it is not really appreciable in Figure 3.23 due to the normalization, there is an small use of the pedals, which in principle should not be needed because we are moving forward. This occurs because of the previously mentioned change in altitude induces  $v_z^B$  to increase. This component generates a yaw rate due to the stability derivative  $N_w$ , which makes necessary the use of the pedal to corrects the error and maintain the needed heading.



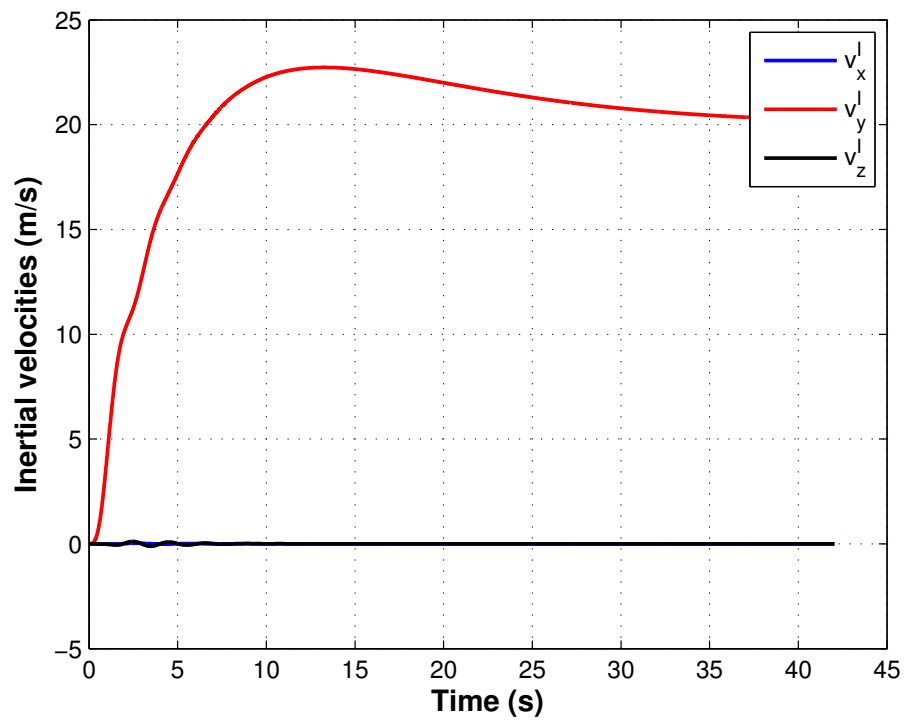


Figure 3.21: Inertial velocities of the helicopter between waypoints 1 and 2

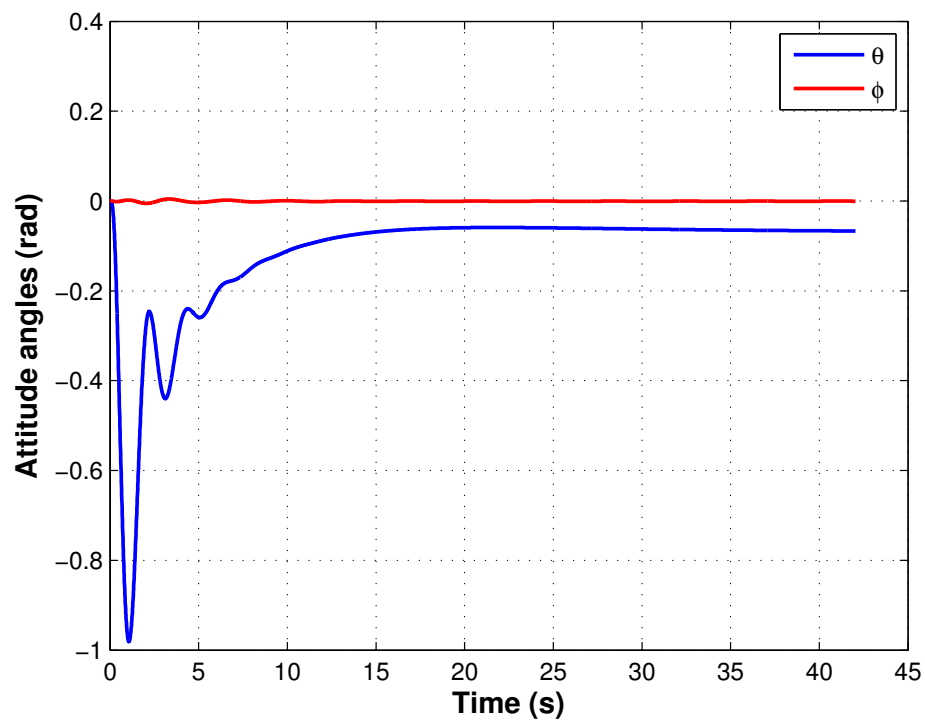


Figure 3.22: Attitude angles between waypoints 1 and 2

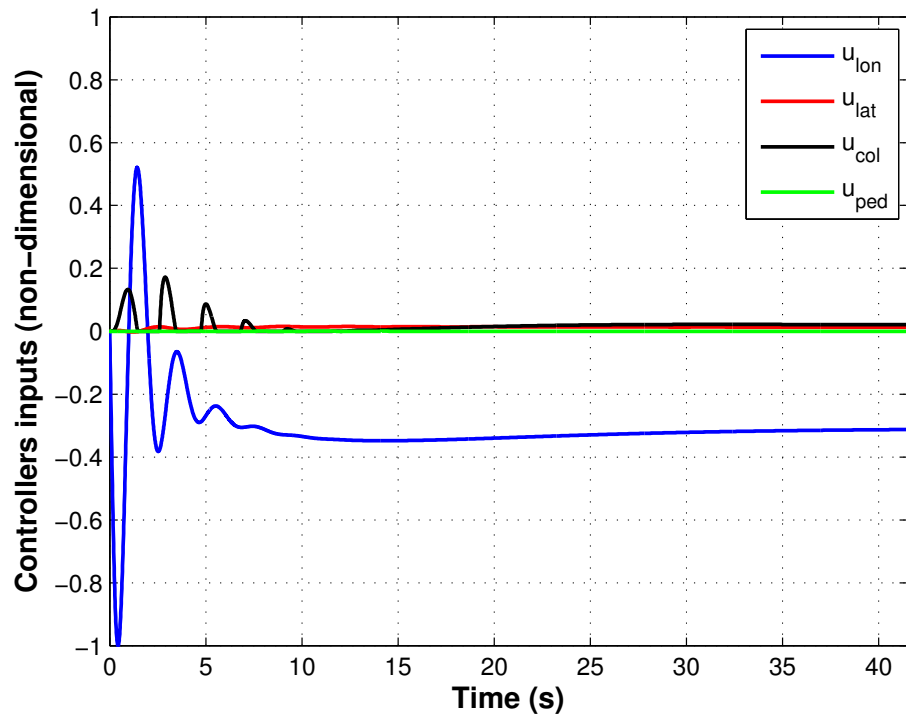


Figure 3.23: Helicopter's controller inputs between waypoints 1 and 2

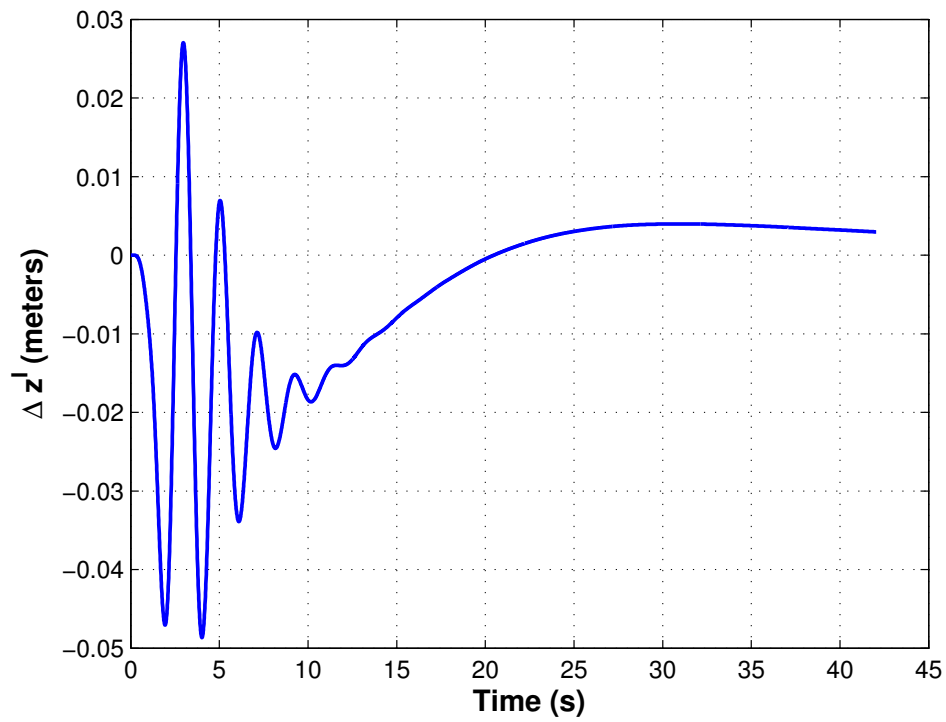


Figure 3.24: Helicopter's  $z^I$  variation between waypoints 1 and 2

## Turn maneuver

Once the vehicle reaches the first waypoint, the turning maneuver starts. As it can be seen from Figure 3.25, the inertial y-velocity component starts its reduction as x-velocity increases, performing the desired turn. The y-velocity component reaches negative values due to the turn the vehicle starts to move towards the x-axis. This changes are produced by the controllers inputs showed in figure 3.28, where it can be seen how the longitudinal and lateral cyclic commands are used to change the velocities. Also the collective command has a small increase to overcome the small changes in the altitude due to the non-linearity and the changes in the thrust direction.

From Figure 3.27 it can be observed how the yaw rate changes towards the desired one in a smooth way. This turn is also appreciable at the attitude angles shown in 3.26 that are produced due to its corresponding controllers that generates the flapping angles that finally generates the attitude angles.

To conclude, as it has been seen through this section, the RPAS is able to properly perform the flight plan established by TSP algorithms. Then, it can be conclude that the generated path planning can be used for small-scale helicopters such as the Raptor 90 SE.

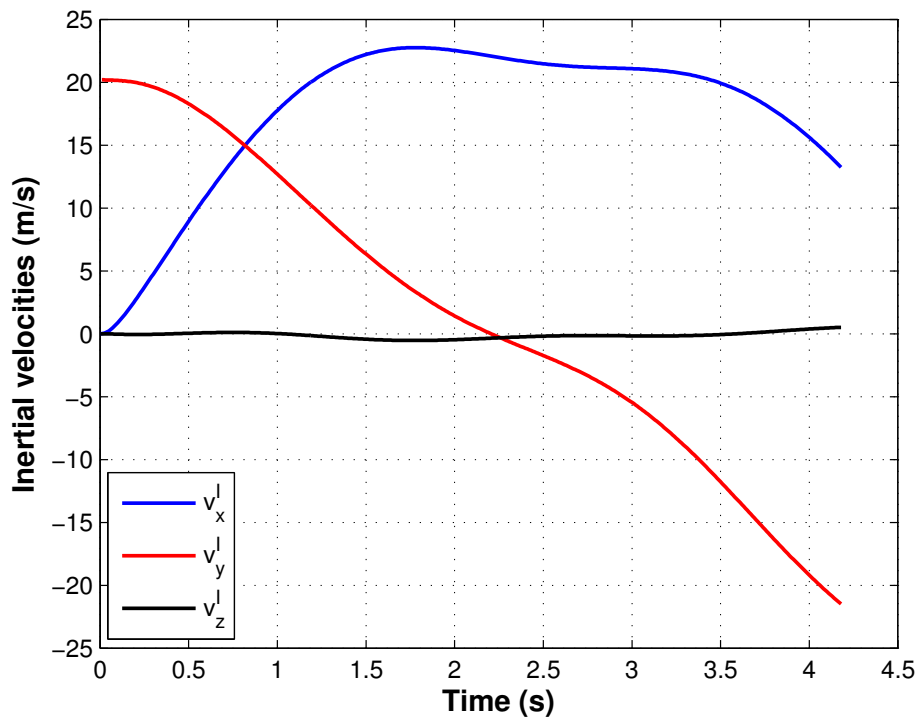


Figure 3.25: Inertial velocities of the helicopter during first turn in the open path planning

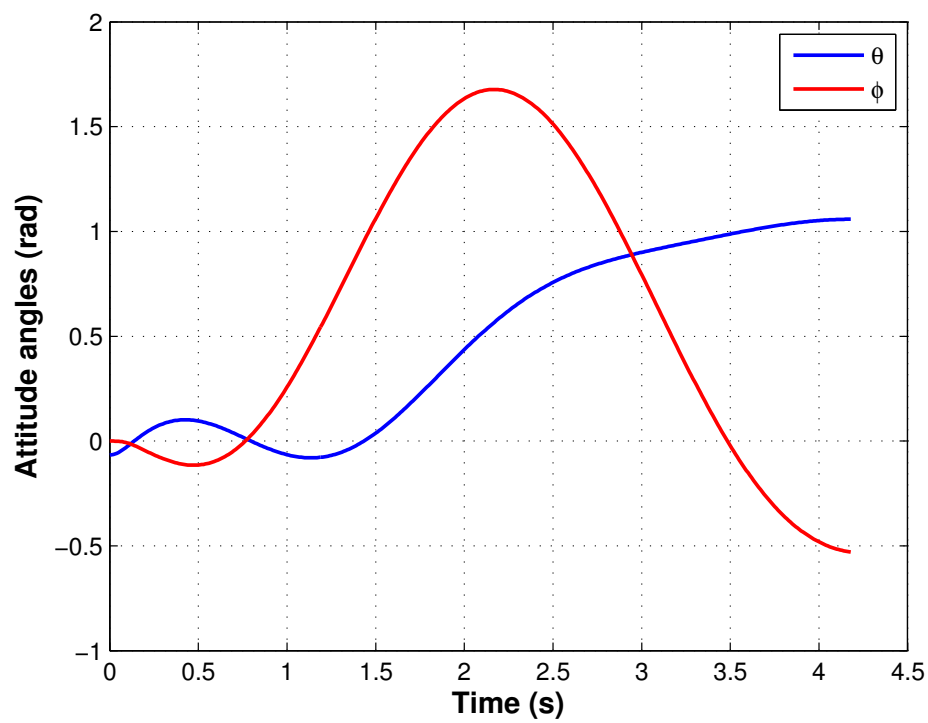


Figure 3.26: Attitude angles of the helicopter during first turn in the open path planning

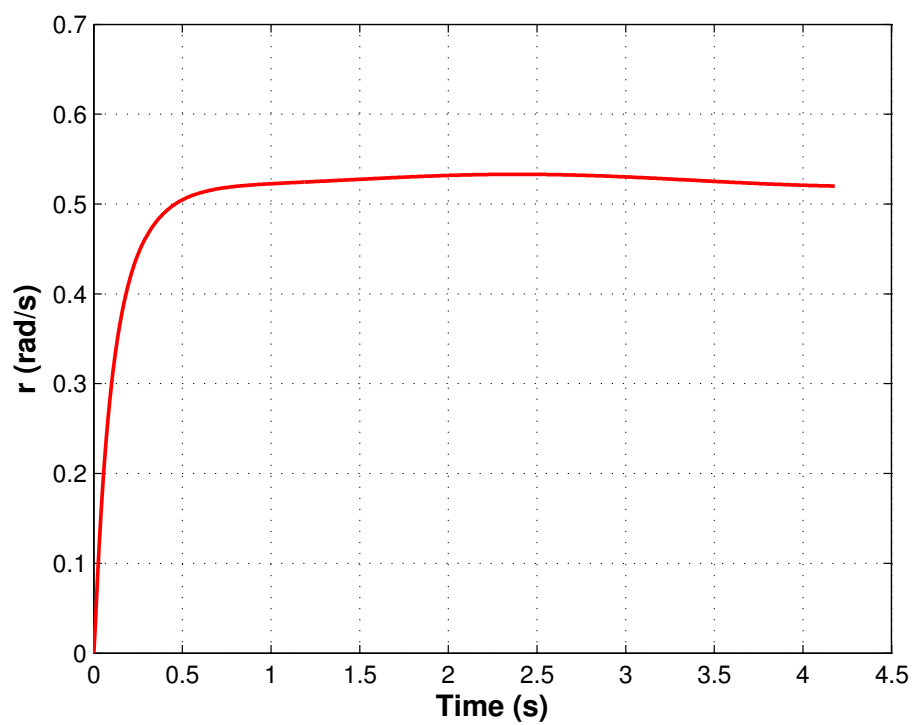


Figure 3.27: Yaw rate of the helicopter during first turn in the open path planning

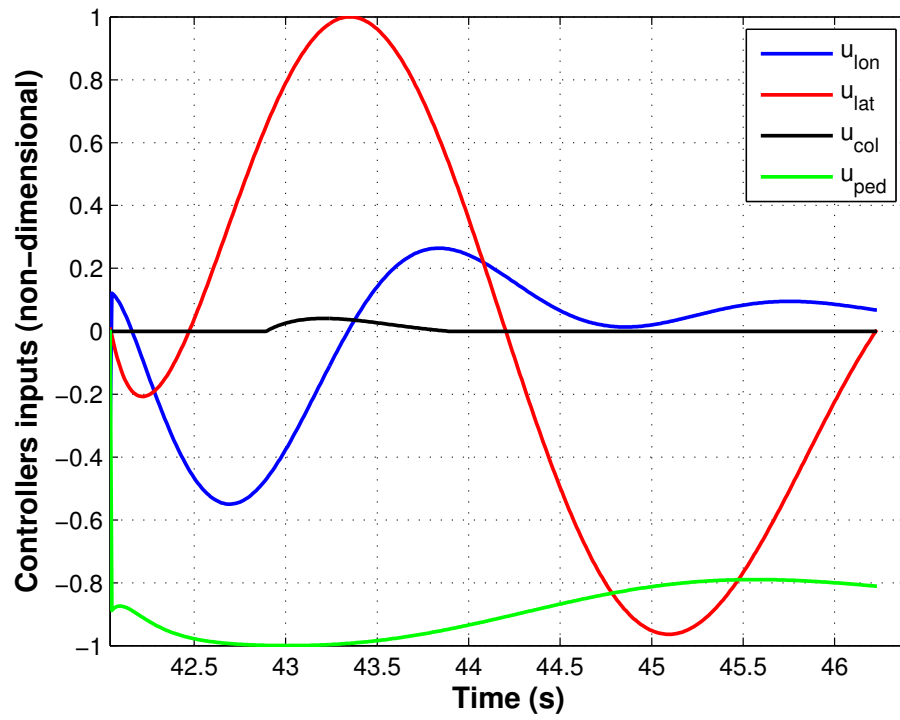


Figure 3.28: Helicopter's controller inputs during first turn in the open path planning



## Chapter 4

# Real case of study

### 4.1 Problem definition

Although the approximated area in Chapter 3 can fit well areas with small changes in altitude, it has been decided to implement another algorithm to find the minimum set of waypoints needed to fully cover a more complex area.

Those areas are going to be real scenarios from which its topographic map is available in digital format. For instance, the *Instituto Nacional de Geografia* (ING) provides, for free, with three different MDT (Terrain Digital Model): MDT5, MDT25 and MDT200. The numbers 5, 25 and 200 refers to the model mesh. These models use UTM (Universal Transverse Mercator) projection in the corresponding zone.

In particular, the area used for this problem is going to be an area of  $[15km \times 15km]$  of the Madrid MDT200 (see Figure 4.1). The radio-range coverage of the LIFESEEKER is set to its real value, 4000 meters, and, as for the simple area, the flight altitude of the vehicle is going to be set to  $z = 100m$  from the terrain, so that the distance between ground and the vehicle is always the same.

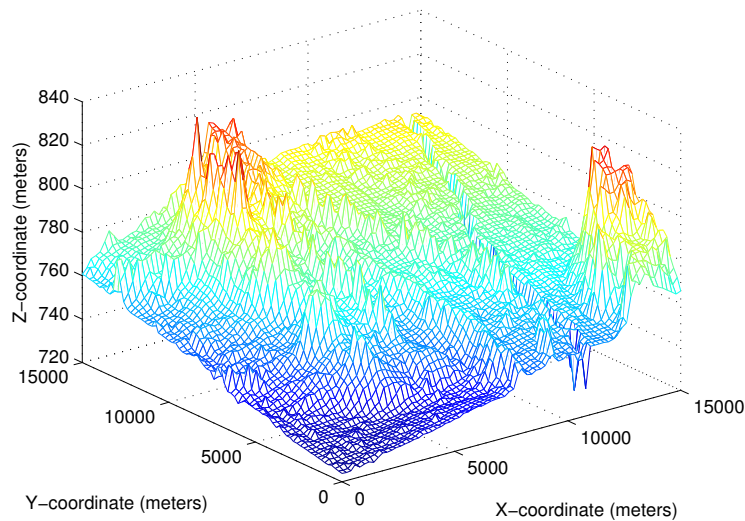


Figure 4.1: MDT200 Section of Madrid, Spain

### 4.1.1 Universal Transverse Mercator background

The UTM system conformal projection uses a 2-dimensional Cartesian coordinate system to provide a given location on the surface of the Earth <sup>1</sup> independently of the vertical position. But this system is not a single map, it divides the Earth into 60 different zones (see Figure 4.2), each being a 6° band of longitude, and uses the secant transverse Mercator projection in each zone. For instance, the Iberian Peninsula is located along zones 29, 30 and 31.

Moreover, each zone is divided in 20 latitude bands of 8°, that are denominated from *C* to *X* in the English alphabet and excluding the letters "i" and "o". The Iberian Peninsula is inside bands T and S.



Figure 4.2: By User:Stefan Kühn (Own work) [GFDL (<http://www.gnu.org/copyleft/fdl.html>) or CC-BY-SA-3.0 (<http://creativecommons.org/licenses/by-sa/3.0/>)], via Wikimedia Commons

To change from UTM coordinates to latitude, longitude it is necessary to know the East, North coordinates, the zone and the hemisphere of the band. The maps provided by the ING, gives explicitly the East, North coordinates but not the hemisphere and the zone. Nevertheless this can be obtained by knowing the autonomic community of the map that is being used.

## 4.2 Area coverage

Once the area has been selected, the area coverage problem can be defined. As it was explained in Chapter 2, this problem can be addressed as a set cover problem, where there are a set of potential waypoints defined by the terrain mesh, that can cover a determined number of points of the terrain, and there is a set of waypoints that can cover the whole area.

Then, if the terrain mesh is defined as  $Terrain_{mesh} = [x_{mesh}, y_{mesh}, z_{mesh}]^T$ , the set of potential waypoints can be defined as  $waypoints_{potential} = [x_{mesh}, y_{mesh}, z_{mesh} + z]^T$ . Therefore the error added due to this reduction in the set of potential waypoints will be proportional to the mesh of the terrain, being lower for MDT5 than MDT200.

But before start explaining the set cover problem and its resolution method, another important issue should be addressed, shadowed areas. Since now the topography can be much more complex with changes in altitude equals or greater to the selected flight altitude, it is possible that some shadowed areas exist.

A shadow area is an area that although ideally the sensor should cover, it doesn't because there is an object between the sensor and that area. For this case of study the object could be a mountain or a sudden elevation of the terrain for instance. To overcome this problem, in the selection of the points that can cover each potential waypoint a constrain is going to be imposed.

<sup>1</sup>Wikipedia. *Universal Transverse Mercator coordinate system* — Wikipedia, The Free Encyclopedia. Online; accessed 10-September-2015. 2015. URL: [https://en.wikipedia.org/w/index.php?title=Universal\\_Transverse\\_Mercator\\_coordinate\\_system&oldid=673304969](https://en.wikipedia.org/w/index.php?title=Universal_Transverse_Mercator_coordinate_system&oldid=673304969)



A line is going to be drawn between each point inside the set of each waypoint and the corresponding waypoint. If this line intercepts the mesh at any point different than the corresponding point where the line starts, that point should be dropped from the set because there is another point that make that point to be inside a shadow area. Then, there is no chance for shadow areas to be taken into account as part of the set, and we can go ahead with the set cover problem.

As it was introduced in Chapter 2, the set cover problem is going to be solved by implementing a greedy algorithm. In particular, the greedy algorithm <sup>2</sup> developed in [25], which is based in Chvatal algorithm[12] has been used. This algorithm has two small differences from the original one:

- In case of several choices at one step, the biggest one is selected.
- When solution is found, the chosen sets are checked to find a better solution, removing a set if is a subset of the union of the other sets.

#### 4.2.1 Set cover problem statement

According to [26], *Given a collection  $S$  of sets over a universe  $U$ , a set cover  $C \subseteq S$  is a subcollection of the sets whose union is  $U$ .* Then, the set cover problem can be defined as: Given  $S = \{S_1, S_2, \dots, S_n\}$ , find the smallest subset of  $S$  whose union equals the universe.

For our particular case,  $S$  is the collection of potential waypoints, each set  $S_i$  is formed by the points that covers the system at that waypoint and  $U$  is all the points in the MDT.

Mathematically, it can be formulated in the following way <sup>3</sup>:

$$\min \sum_{i=1}^n S \quad (4.1)$$

subject to:

$$\sum_{i:v \in S_i} x_i \geq 1 \quad \forall v \in U \quad (4.2a)$$

$$x_i \in \{0, 1\} \quad \forall i \in \{1, 2, \dots, n\} \quad (4.2b)$$

Equation 4.1 minimizes the number of sets. Constrain 4.2a ensures that every element of the universe is covered. Constrain 4.2b ensures every set is either in the set cover or not.

#### 4.2.2 Greedy algorithm

A *greedy algorithm* is an algorithm that follows the problem solving heuristic of making the logically optimal choice at each stage<sup>4</sup>, with the aim of finding the optimal solution. Nevertheless, in general it does not produce the optimal solution but an approximation to the optimal one in a reasonable time.

In general, greedy algorithms have five components <sup>5</sup>:

1. A candidate set, from which a solution is created
2. A selection function, which chooses the best candidate to be added to the solution
3. A feasibility function, that is used to determine if a candidate can be used to contribute to a solution
4. A solution function, which will indicate when it has been discovered a complete solution

<sup>2</sup>Fabio Gori. *Greedy algorithm for Set Cover problem*. Retrieved March 2015. 2010. URL: <http://www.mathworks.com/matlabcentral/fileexchange/29650-greedy-algorithm-for-set-cover-problem>.

<sup>3</sup>Wikipedia. *Set cover problem* — *Wikipedia, The Free Encyclopedia*. Online; accessed 10-September-2015.2015. URL: [https://en.wikipedia.org/w/index.php?title=Set\\_cover\\_problem&oldid=671504353](https://en.wikipedia.org/w/index.php?title=Set_cover_problem&oldid=671504353)

<sup>4</sup>Black, Paul E. (2 February 2005). "greedy algorithm". *Dictionary of Algorithms and Data Structures*. U.S. National Institute of Standards and Technology (NIST). Retrieved 17 August 2012

<sup>5</sup>Wikipedia. *Greedy algorithm* — *Wikipedia, The Free Encyclopedia*. Online; accessed 10-September-2015. 2015. URL: [https://en.wikipedia.org/w/index.php?title=Greedy\\_algorithm&oldid=675973911](https://en.wikipedia.org/w/index.php?title=Greedy_algorithm&oldid=675973911)

The greedy algorithm developed by [12] (Algorithm 2), is a very simple greedy algorithm that choose the set with the largest number of uncovered elements at each stage. This kind of greedy algorithm can be efficiently implemented in time that is linear in the size of the input [27], then, the computational time is going to be proportional to the size of the area to be covered and the type of mesh used.

---

**Algorithm 2:** Greedy algorithm for Set Covering (Chvatal, 1979)

---

**Input** : Family of sets  $C_1, \dots, C_n$  ( $R := \cup_{k=1}^n C_k$ )  
**Output:**  $J \subseteq \{1, \dots, n\}$ , s.t.  $\cup_{j \in J} C_j = R$   
 $U \leftarrow R$ ;  
 $J \leftarrow \emptyset$ ;  
**while**  $U \neq \emptyset$  **do**  
    Select  $\hat{i} \in \{1, \dots, n\} \setminus J$  s.t.  $|C_{\hat{i}} \cup U|$  is maximum;  
     $U \leftarrow U \setminus C_{\hat{i}}$ ;  
     $J \leftarrow J \cup \{\hat{i}\}$ ;  
**end**  
**return** ( $J$ )

---

As it was explained in Section 4.2, the algorithm used to solve our set cover problem is based on Algorithm 2, but it has two important additional features: in case of several options that provide the same number of uncovered elements at the same step, the largest set is chosen; and after the algorithm finished, the set cover is checked to remove sets that are subsets of others. The process of this algorithm is illustrated through the following toy example.

Suppose we have 6 different,  $\{C_1, C_2, \dots, C_6\}$ , whose union covers 10 different elements,  $\{r_1, r_2, \dots, r_{10}\}$  (see Table 4.1. A bullet in an entry  $(i, j)$  means that that element  $r_i$  is covered by the subset  $C_j$ . Then, the algorithm starts looking for the set cover. As the algorithm selects set with the biggest amount of uncovered elements, the first choice is  $C_1$ . At this point all the other sets provide the same amount of uncovered elements. Due to the modification introduced by Fabio Gori, under this situation, the algorithm would select the set that provide a larger number of elements, that is  $C_2$ . Finally, the only remaining set that provides more uncovered elements completing the solution is  $C_3$ . Then the cover set,  $J$ , will be  $J = \{C_1, C_2, C_3\}$ . At this point we will check if there is a set that is a subset of others sets in order to remove it, but for this toy problem it is not needed.

	$C_1$	$C_2$	$C_3$	$C_4$	$C_5$
$r_1$	•	•			
$r_2$	•			•	
$r_3$			•		•
$r_4$		•		•	
$r_5$	•				
$r_6$			•		
$r_7$		•			
$r_8$	•				•

Table 4.1: Input covering matrix

### 4.2.3 Waypoint set definition

Once the method to obtain the set of waypoints needed to fully cover a real area has been defined, the method can be applied to obtain the set for the real scenario described in Section 4.1. The resulting set of waypoints is:

$$waypoints = [x, y, z] = \begin{bmatrix} 0 & 3400 & 831 \\ 2600 & 3400 & 833 \\ 3400 & 3400 & 918 \\ 3600 & 11800 & 922 \\ 3800 & 10800 & 886 \\ 3800 & 8400 & 883 \\ 5600 & 5400 & 860 \\ 6400 & 2000 & 837 \\ 11200 & 11200 & 881 \\ 11200 & 13800 & 887 \\ 11600 & 7800 & 863 \\ 12200 & 2800 & 858 \end{bmatrix} \quad (4.3)$$

This set is shown in Figures 4.3, where each number corresponds to the index of each element in the array.

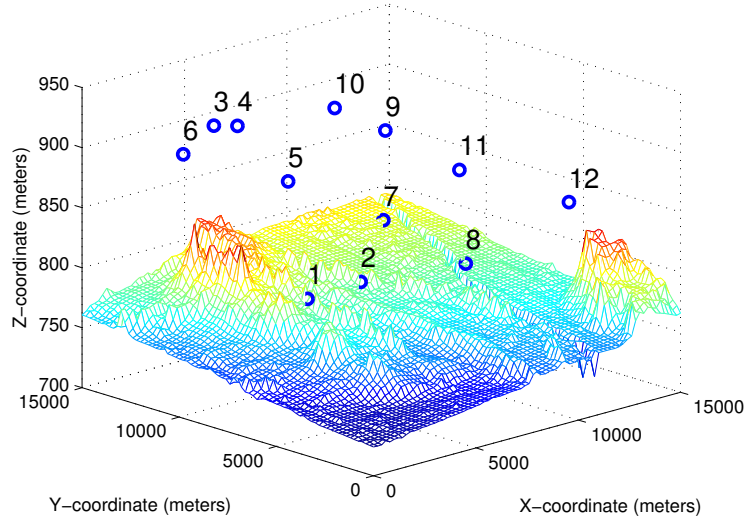


Figure 4.3: 3D representation of the waypoints needed to cover the area

As can be seen, there are some waypoints that are relatively close to each other with respect to the coverage radius of the LIFESEEKER. This is because of the shadowed areas previously described, that makes necessary to include extra waypoints to reach all the points of the terrain.

## 4.3 Path planning

In order to find the sequence to be followed to reach all the previously found waypoints the same algorithms that were implemented in Chapter 3 are going to be used.

For the open path planning, starting at 1 with 1000 population size and 1000 iterations, the resulting path (taking waypoints indexes in 4.3 as reference) is:

$$waypoints = [x, y, z] = [1 \ 2 \ 8 \ 7 \ 5 \ 4 \ 3 \ 6 \ 10 \ 9 \ 11 \ 12] \quad (4.4)$$

This path planning is shown in Figure 4.4

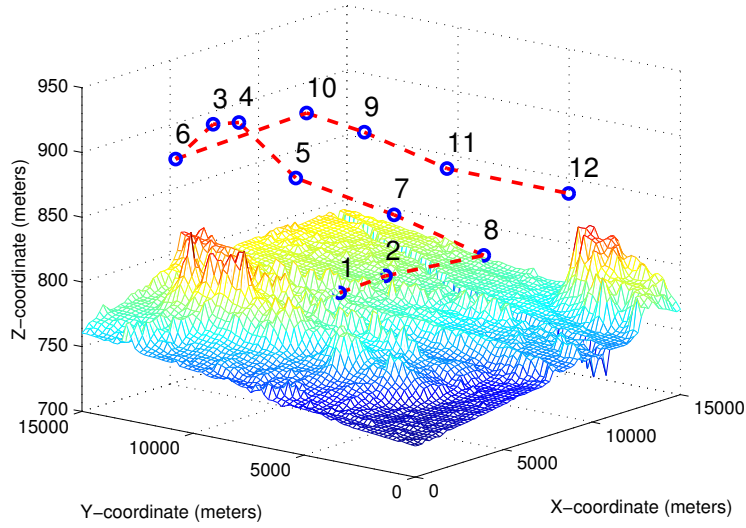


Figure 4.4: 3D representation of the open path planning

For the close path planning, starting at 1 with 1000 population size and 1000 iterations, the resulting path (taking waypoints indexes in 4.3 as reference) is:

$$\text{waypoints} = [x, y, z] = [1 \ 5 \ 4 \ 3 \ 6 \ 10 \ 9 \ 11 \ 12 \ 8 \ 7 \ 2 \ 1] \quad (4.5)$$

This path planning is shown in Figure 4.5

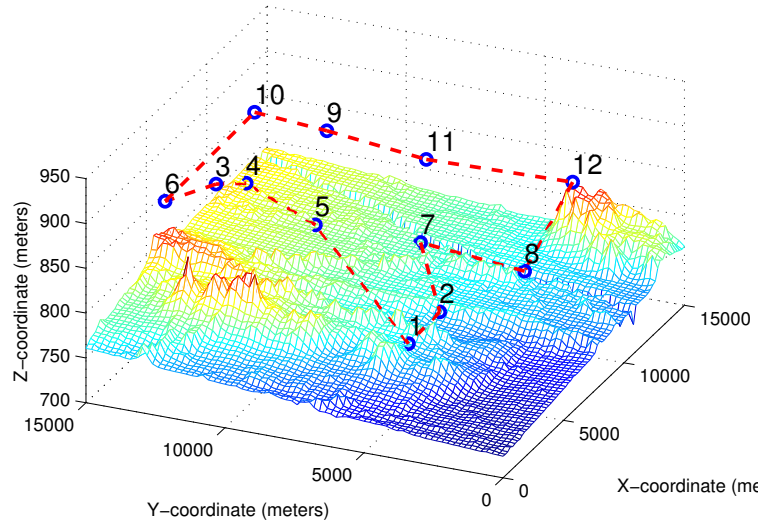


Figure 4.5: 3D representation of the close path planning

## 4.4 Test environment

As the type of path generated through the TSP algorithm were already proved to be followed by the Raptor 90 SE and no other helicopter's vehicle has been obtained, no relevant data is going to be obtained by implementing the simulation again to these new paths. The main reason is that the turns are going to be performed in the

same way and the only thing that changes is that the rectilinear paths are longer and include a z component, but these changes in altitude are not too high either, so obviously are going to be performed well by the RPAS.

In addition, the Raptor 90 SE cannot actually perform these flight plans due to its limitations in the range of the transmitter signal, the endurance and the MTOW. Then, this simulation has been omitted.



## Chapter 5

# Commercial application

### 5.1 An application for the real world

To conclude the project, it has been decided to implement a tool that can be used in a real search and rescue application by the involved teams in the mission.

The developed tool is a GUI that enables the user to implement all the described algorithms in this project: find the waypoints to cover an area and generate the path planning needed to reach all of them. Both functions can be used for a real scenario or a simple area as it has been done in Chapters 3 and 4.

The developed GUI can be observed in Figure 5.1. This GUI is able to elaborate a path planning to fully cover a real area according to user parameters.

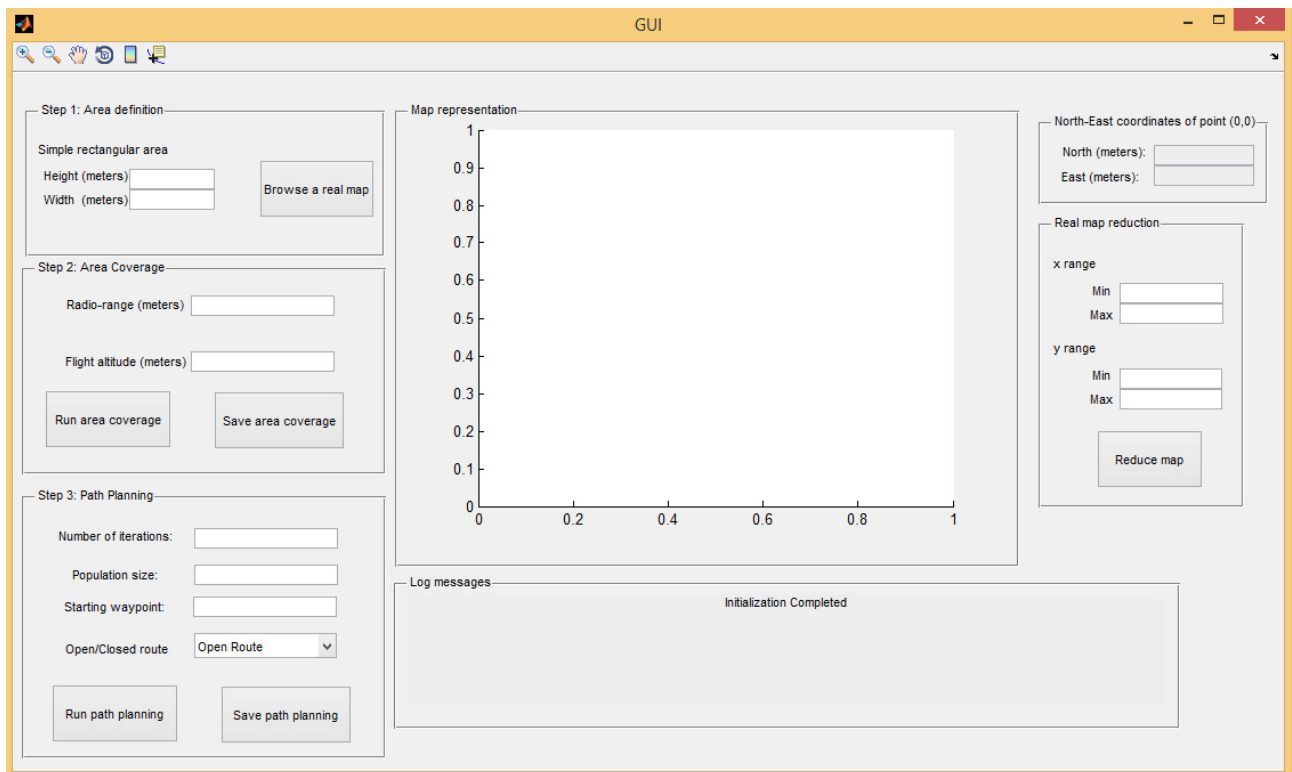


Figure 5.1: GUI

The complete set of functions are described in the following sub-sections.

### 5.1.1 Area definition

There are two options available to define the area to be covered. Select a real map as the one described in Section 4.1 through the browse button or to define a simple area as the one described in Section 3.1. Both commands are shown in Figure 5.2.

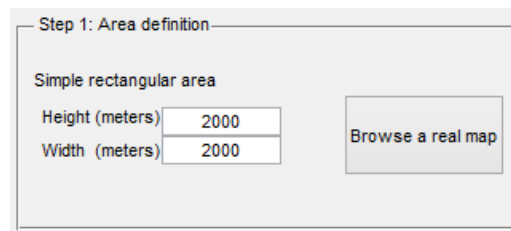


Figure 5.2: GUI Area definition panel

It is important to remark that the program only supports real map files with the format of those that can be downloaded from the webpage of the ING, but this functionality can be adapted to allow other formats.

Once the user select a valid area, it is showed at the "Map" panel (see Figures 5.3 and 5.4). In addition the North-East coordinates of the point (0,0) of the map represented at the figure are shown at the panel shown in Figure 5.5. This information allows the user to change from UTM coordinates to latitude, longitude coordinates by itself. For the case of simple area, these values are set to zero.

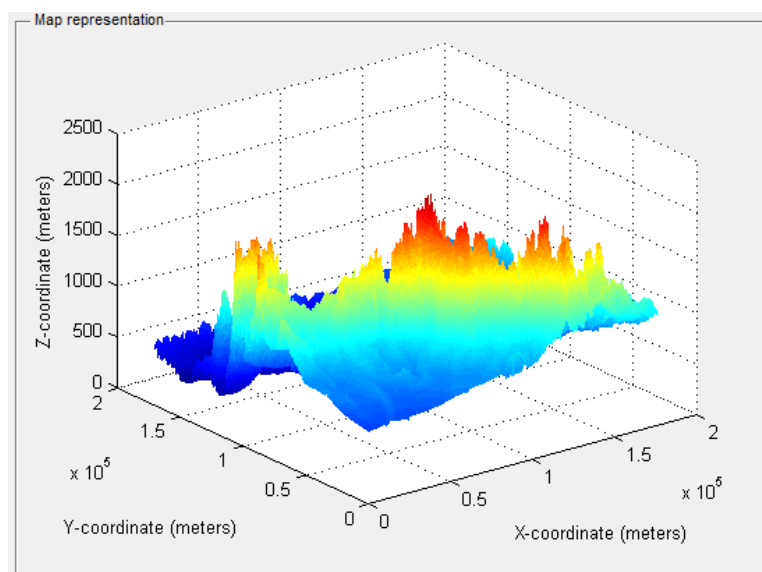


Figure 5.3: GUI Map panel when a real map is introduced



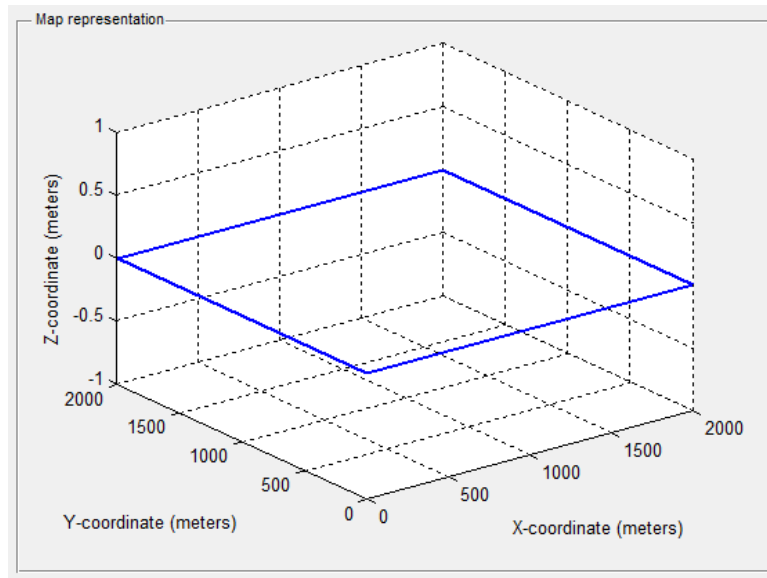
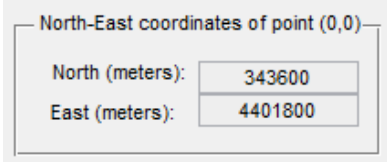


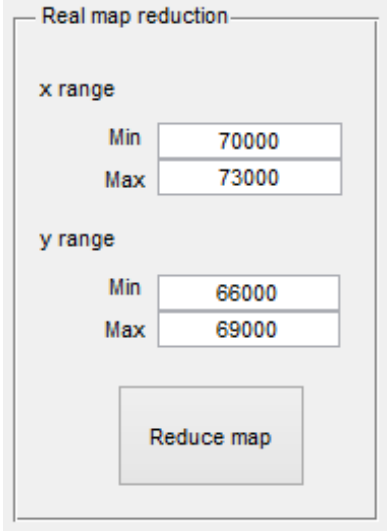
Figure 5.4: GUI Map panel when a simple map is introduced



A panel titled "North-East coordinates of point (0,0)" containing two input fields. The first field is labeled "North (meters):" and contains the value "343600". The second field is labeled "East (meters):" and contains the value "4401800".

Figure 5.5: GUI UTM North-East map coordinates panel

Furthermore, the map reduction panel (see Figure 5.6) allows the user to select a specific range of the chosen real map. This tool has been designed to allow the rescue teams to delimit the searching area.



A panel titled "Real map reduction" containing two sections for range selection. The first section is labeled "x range" and contains two input fields: "Min" with the value "70000" and "Max" with the value "73000". The second section is labeled "y range" and contains two input fields: "Min" with the value "66000" and "Max" with the value "69000". At the bottom of the panel is a button labeled "Reduce map".

Figure 5.6: GUI Real map reduction panel

The user must introduce the specific range values in the boxes and internally the data is modified. Moreover, the map is actualized to the chosen range (see Figure 5.7).

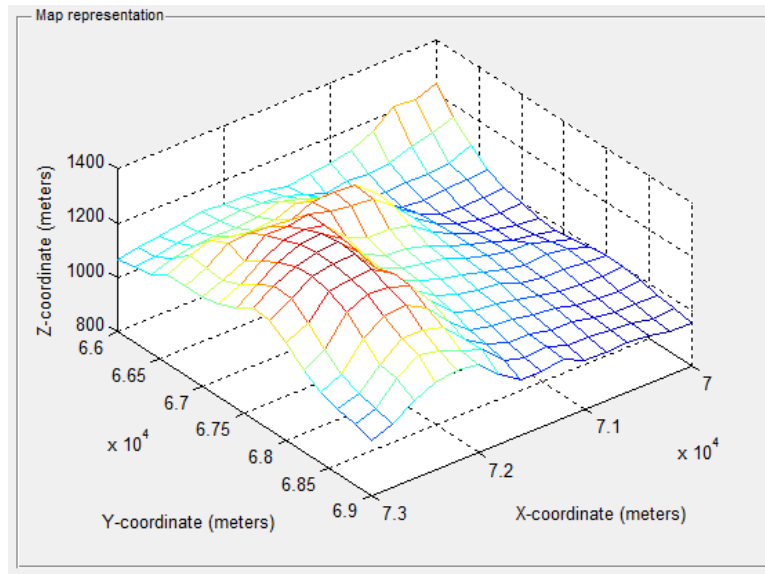


Figure 5.7: GUI Map panel when real map is reduced

### 5.1.2 Area coverage

Once the map has been established, the user can perform the area coverage algorithm explained in Chapters 3 and 4. However, first, the radio-range coverage of the system being used and the desired flight altitude has to be introduced (see Figure 5.8).

Figure 5.8: GUI Area coverage panel

Then, by clicking the button "Run area coverage", the waypoints that are needed to fully cover the area are calculated and showed at the map panel (see Figure 5.9). The program identifies by itself what algorithm has to be used between the two described in Chapters 3 and 4.

In addition a label is assigned to each one in order to allow the user to select the desired starting waypoint. Those waypoints can be saved in the desired directory by clicking on "Save area coverage".

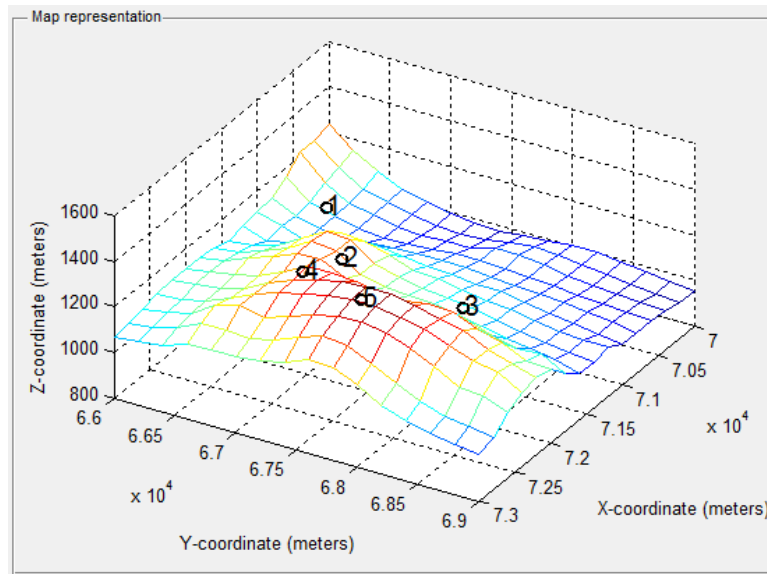


Figure 5.9: GUI Map panel when the area coverage algorithm finish

### 5.1.3 Path planning

When the waypoints has been established, the user can run the path planning by clicking on the button "Run path planning". But first it has to introduce the number of iterations and the population size desired for the genetic algorithm execution, and the desired starting waypoint (see Figure 5.10).

Figure 5.10: GUI Path planning panel

The obtained sequence is plotted on the Map panel (see Figure 5.11) and it can be saved through the "Save path planning" button.

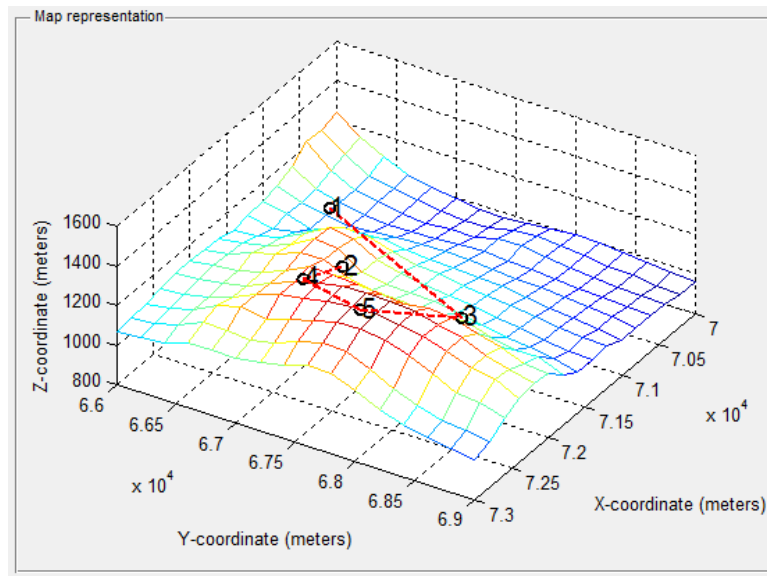


Figure 5.11: GUI Map panel when the path planning algorithm finish

#### 5.1.4 Log messages

During the whole execution of the GUI, the Log messages panel shows the user the internal evolution of the different functions as well as the produced error due to user wrong commands (see Figure 5.12). It also presents the resulting length of the generated flight plan.

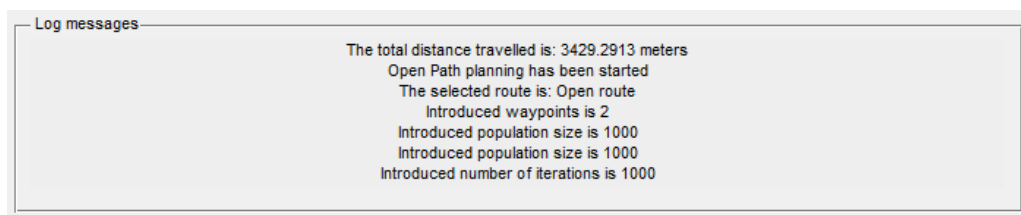


Figure 5.12: GUI Log messages panel after several commands introduced by the user

## Chapter 6

# Conclusions

To conclude, several conclusions can be inferred from the project and the goals established at the beginning:

- The first objective was to generate the path planning for a benchmark problem. But in order to reach this goal, it was divided in several sub-objectives:
  - The first one was to generate a set of waypoints that fully cover a simple area. This goal was achieved successfully by the implementation of the work described in Section 2.2. Then, by implementing the honeycomb structure with the boundary constraints, the full area was covered while the overlapping was minimized.
  - Second sub-objective was to generate the adequate sequence to cover a set of waypoints with the objective to minimize the travelled distance. To make this path planning, the Travelling Salesman Problem was introduced and solved by the implementation of a genetic algorithm due to its particular features in computational terms. The two proposed alternatives (open and close path plannings) were successfully described in order to cover the possibility that the RPAS has not enough endurance to complete the path planning and return home.
  - The third one was to prove that the generated path planning can be performed by a small-scale helicopter by the implementation of a test environment. The results were satisfactory, the test environment concludes that it was good enough to represent the desired flight plan and the helicopter model provides us a good representation of the flight's characteristics. Finally, figures in Chapter 3 displayed the flight performance, which coincided with the expected one.
- The second objective was to develop the algorithms needed to generate another path planning for a real scenario. This was done successfully by defining another area coverage problem, the set cover problem, which was solved by a greedy algorithm. In addition, this area coverage also takes into account the possible shadow areas that could be, which is a big issue to take into account in order to avoid false positive results.
- Finally, the last objective was to automatize all these algorithms through the implementation of a tool that could be used for real search and rescue missions. This was done through the development of a Graphical User Interface that enables a user, without a deep knowledge in the subject, to generate its own path planning for simple areas or real scenarios. As it was presented in Chapter 5, this GUI was developed with all the features that an arbitrary user could need for the development of path plannings in search and rescue missions using the LIFESEEKER or another similar system.

Summarizing, it can be concluded that the project has been successfully completed by achieving all the goals established at the beginning.

Before finishing, I would like to present my own conclusions inferred from the realization of this project. As far as I am concerned, this work has given me a huge knowledge in a very different subject:

- I have increased my knowledge on the RPAS sector.
- I have learnt about the flight mechanic and the equations of motion that governs the movement of an helicopter.

- I have developed a knowledge in optimization algorithms by the implementation of different algorithms that I had never studied before.
- I have enhanced my programming skills by learning how to program a Graphical User Interface.

## 6.1 Future work

Although the project has been concluded, there is still work to do about this matter. The open issues that could be enhanced are the following ones:

- All the algorithms could be optimized in computational terms by means of parallel computation. This is a computational method where several computations are carried out by different processors at the same time.
- In the area coverage problem another variables should be taken into account in order to find out the points actually covered by the sensor. For instance the signal attenuation under different atmospheric conditions could be taken into account.
- The sequence of the waypoints generated could take into account other variables such as the wind, the number of turns or if there are areas with more probability that the missing person is there.
- The data of an RPAS which is being used nowadays in search and rescue missions should be used to test the flight plans in order to simulate them in a more real environment.
- The test environment could be enhanced by introducing a non-linear controller or using an existing and more developed flight simulator as the ones described in Chapter 2.

# Bibliography

- [1] International Civil Aviation Organization. *Manual on remotely piloted aircraft systems (RPAS)*. Ed. by International Civil Aviation Organization. Montreal: International Civil Aviation Organization, 2015.
- [2] Spain. *Ley 18/2014, de 15 de octubre, de aprobacion de medidas urgentes para el crecimiento, la competitividad y la eficiencia*. Oct. 2014.
- [3] Salvamento Martimo. *Informe Anual 2014 Sociedad de Salvamento y Seguridad Martima*. Tech. rep. Ministerio de Fomento, 2015.
- [4] Jesper Pedersen. “CRS Conference-ICT4Development”. In: *Use of UAVs in the NGO World*. Nairobi, Kenya, Mar. 2014.
- [5] Tarnai, Tibor and Gáspár, Zsolt. “Covering a Square by Equal Circles”. In: *Elemente der Mathematik* 50.4 (1995), pp. 167–170.
- [6] Jan B. M. Melissen and PC Schuur. *Improved coverings of a square with six and eight equal circles*. Vakgroep OMST, 1996.
- [7] Nurmela, Kari J et al. “More optimal packings of equal circles in a square”. In: *Discrete & Computational Geometry* 22.3 (1999), pp. 439–457.
- [8] Richard Kershner. “The Number of Circles Covering a Set”. English. In: *American Journal of Mathematics* 61.3 (July 1939), pp. 665–671. URL: <http://www.jstor.org/stable/2371320>.
- [9] On wireless network coverage in bounded areas. “Yu, Zuoming and Teng, Jin and Li, Xinfeng and Xuan, Dong”. In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. Apr. 2013, pp. 1195–1203. ISBN: 9781467359467.
- [10] Fisher, Marshall L and Kedia, Pradeep. “Optimal solution of set covering/partitioning problems using dual heuristics”. In: *Management science* 36.6 (1990), pp. 674–688.
- [11] Beasley, John E and Jörnsten, Kurt. “Enhancing an algorithm for set covering problems”. In: *European Journal of Operational Research* 58.2 (1992), pp. 293–300.
- [12] Chvatal, Vasek. “A greedy heuristic for the set-covering problem”. In: *Mathematics of operations research* 4.3 (1979), pp. 233–235.
- [13] Jacobs, Larry W and Brusco, Michael J. “Note: A local-search heuristic for large set-covering problems”. In: *Naval Research Logistics (NRL)* 42.7 (1995), pp. 1129–1140.
- [14] Ohlsson, Mattias, Peterson, Carsten, and Söderberg, Bo. “An efficient mean field approach to the set covering problem”. In: *European Journal of Operational Research* 133.3 (2001), pp. 583–595.
- [15] Dwivedi, Varshika et al. “Travelling Salesman Problem using Genetic Algorithm”. In: *IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012)* 1 (2012), p. 25.
- [16] Tatiana Tabirca, Laurence T. Yang, and Sabin Tabirca. “Smallest Number of Sensors for k-Covering”. In: *International Journal of Computers Communications & Control* 8.2 (Feb. 2013), p. 312.
- [17] Čičková, Zuzana, Brezina, Ivan, and Pekár, Juraj. “Open Traveling Salesman Problem with Time Windows”. In: *1st Logistics International Conference Belgrade, Serbia* (2013), pp. 28–30.
- [18] Bhatia, Karan. “Genetic Algorithms and the Traveling Salesman Problem”. In: *Computer Science and Engineering CSE292: New Age Algorithms, University of California at San Diego* (1994).
- [19] Bernard Mettler. *Identification modeling and characteristics of miniature rotorcraft*. Springer Science & Business Media, 2013.

- [20] Agus Budiyo and Singgih S Wibowo. “Optimal tracking controller design for a small scale helicopter”. In: *Journal of bionic engineering* 4.4 (2007), pp. 271–280.
- [21] Guowei Cai et al. “Modeling and control system design for a UAV helicopter”. In: *Control and Automation, 2006. MED’06. 14th Mediterranean Conference on*. IEEE. 2006, pp. 1–6.
- [22] David Hyunchul Shim, Hyoun Jin Kim, and Shankar Sastry. “Control system design for rotorcraft-based unmanned aerial vehicles using time-domain system identification”. In: *Control Applications, 2000. Proceedings of the 2000 IEEE International Conference on*. IEEE. 2000, pp. 808–813.
- [23] Jinok Shin et al. “Model-based optimal attitude and positioning control of small-scale unmanned helicopter”. In: *Robotica* 23.01 (2005), pp. 51–63.
- [24] Ioannis A Raptis and Kimon P Valavanis. *Linear and nonlinear control of small-scale unmanned helicopters*. Vol. 45. Springer Science & Business Media, 2010.
- [25] Gori, Fabio et al. “MTR: taxonomic annotation of short metagenomic reads using clustering at multiple taxonomic ranks”. In: *Bioinformatics* 27.2 (2011), pp. 196–203.
- [26] Young, Neal E. “Greedy set-cover algorithms”. In: *Encyclopedia of algorithms*. Springer, 2008, pp. 387–381.
- [27] Bar-Yehuda, Reuven and Even, Shimon. “A linear-time approximation algorithm for the weighted vertex cover problem”. In: *Journal of Algorithms* 2.2 (1981), pp. 198–203.



## Appendix A

Linear state-space model parameters of the Raptor 90 SE:

- $X_u = -0.03996$
- $Y_v = -0.05989$
- $M_u = 0.2542$
- $M_v = -0.06013$
- $M_a = 307.571$
- $L_u = -0.02440$
- $L_v = -0.1173$
- $L_b = 1172.48$
- $A_b = 0.7713$
- $g = 9.389$
- $B_a = 0.6168$
- $Z_a = 0$
- $Z_b = 0$
- $Z_w = -2.055$
- $Z_r = 0$
- $N_v = 2.982$
- $N_p = 0$
- $N_w = -0.7076$
- $N_r = -10.71$
- $1/\tau_f = 30.71$
- $A_{lon} = 4.059$
- $A_{lat} = -0.01610$
- $B_{lon} = -0.01017$
- $B_{lat} = 4.085$
- $Z_{col} = -13.11$
- $N_{col} = 3.749$
- $N_{ped} = 26.90$

## Appendix B

The linear velocity of the helicopter with respect to the inertial frame is denoted by  $v^I = [v_x^I \ v_y^I \ v_z^I]^T$ . The linear velocity of the helicopter with respect to the body frame is denoted by  $v^B = [u \ v \ w]^T$ . Both velocity can be expressed as:

$$\vec{v}^B = u\vec{i}_B + v\vec{j}_B + w\vec{k}_B \quad (1)$$

$$\vec{v}^I = v_x^I\vec{i}_1 + v_y^I\vec{j}_1 + v_z^I\vec{k}_1 \quad (2)$$

Using Euler angles, the unit vectors of the body fixed reference frame  $O_Bx_By_Bz_B$  can be expressed relative to the frame  $O_3x_3y_3z_3$  as:

$$\begin{bmatrix} \vec{i}_B \\ \vec{j}_B \\ \vec{k}_B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} \vec{i}_3 \\ \vec{j}_3 \\ \vec{k}_3 \end{bmatrix} = R_\phi^T [\vec{i}_3 \ \vec{j}_3 \ \vec{k}_3]^T \quad (3)$$

Similarly,  $O_3x_3y_3z_3$  is expressed relative to  $O_2x_2y_2z_2$ :

$$\begin{bmatrix} \vec{i}_3 \\ \vec{j}_3 \\ \vec{k}_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \begin{bmatrix} \vec{i}_2 \\ \vec{j}_2 \\ \vec{k}_2 \end{bmatrix} = R_\theta^T [\vec{i}_2 \ \vec{j}_2 \ \vec{k}_2]^T \quad (4)$$

Finally,  $O_2x_2y_2z_2$  is relative expressed to  $O_1x_1y_1z_1$ :

$$\begin{bmatrix} \vec{i}_2 \\ \vec{j}_2 \\ \vec{k}_2 \end{bmatrix} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{i}_1 \\ \vec{j}_1 \\ \vec{k}_1 \end{bmatrix} = R_\psi^T [\vec{i}_1 \ \vec{j}_1 \ \vec{k}_1]^T \quad (5)$$

By consecutive substitution of 3, 4 and 5 into 1 and 2, the following expression is obtained:

$$\vec{v}^B = [u \ v \ w] R_\phi^T R_\theta^T R_\psi^T [\vec{i}_1 \ \vec{j}_1 \ \vec{k}_1]^T \quad (6)$$

Denote  $R(\Theta)$  the product:

$$R(\Theta) = R(\psi)R(\theta)R(\phi) \quad (7)$$

Equation 6 can be rearranged into:

$$\begin{bmatrix} \vec{v}_x^I \\ \vec{v}_y^I \\ \vec{v}_z^I \end{bmatrix} = R(\Theta) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (8)$$

Where  $R(\Theta)$  is called the rotation matrix:

$$R(\Theta) = \begin{bmatrix} \cos(\theta)\cos(\psi) & \sin(\phi)\sin(\theta)\cos(\psi) - \cos(\phi)\sin(\psi) & \cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi) \\ \cos(\theta)\sin(\psi) & \sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi) & \cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi) \\ -\sin(\theta) & \sin(\phi)\cos(\theta) & \cos(\phi)\cos(\theta) \end{bmatrix} \quad (9)$$