Path Planning and Collision Avoidance Algorithms for Small RPAS

Juliana Maria Medeiros Alves

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Alexandra Bento Moutinho
Prof. André Calado Marta

Examination Committee

Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. Alexandra Bento Moutinho
Member of the Committee: Prof. Pedro Vieira Gamboa

June 2017

Resumo

O desenvolvimento de pequenas Aeronaves Remotamente Pilotadas para aplicações civis tem aumentado nos ultimos anos. Uma das desvantagens na sua integração em espaço aéreo civil é a falta de sistemas de prevenção de colisões. Enquanto que aeronaves remotamente pilotadas mais sofisticadas, nomeadamente militares, têm sensores e sistemas aviónicos topo de gama, drones de baixo peso e custo não têm qualquer tipo de segurança. Dado o aumento exponencial do número deste tipo de exemplares disponíveis no mercado, tipicamente operados por operadores sem certificação, há uma necessidade urgente de desenvolver sistemas de prevenção de colisão que possam ser facilmente integrados neste tipo de plataformas.

Dentro do vasto tópico de sistemas de prevenção de colisoēs, esta dissertação foca-se no problema de gerar caminhos óptimos para Aeronaves Remotamente Pilotadas sujeitas a restrições de manobrabilidade e segurança, e fazer o seu replaneamento quando em risco de colisão. Para atingir este objectivo é proposta uma solução dividida em duas fases. Na primeira fase técnicas clássicas de planeamento de trajectória são implementadas para gerar caminhos num ambiente estático conhecido. Algoritmos de procura em grelha, nomeadamente o algoritmo A* e a Colónia de Formigas, são usados para encontrar uma sequencia óptima de pontos num ambiente discreto. Para ter a certeza que o caminho respeita restrições de curvatura e segurança, uma optimização de curvas racionais de Bezier é implementada.

Enquanto a primeira fase é feita antes da execução da missão, a segunda fase, responsável por evitar obstáculos dinâmicos e estáticos que não eram previamente conhecidos, é desenvolvida com as limitações de implementação em tempo real em mente. O método dos campos potenciais é usado para evitar colisões em tempo real. É assumido que o veiculo está equipado com piloto automático e unidade de controlo de trajectória.

Casos de planeamento de trajectória entre um conjunto de pontos num ambiente bimensional e tridimensional com obstáculos estáticos, usando os algoritmos desenvolvidos na primeira fase, são apresentados. Vários casos de replaneamento de trajectória, usando os campos potenciais, na presença de obstáculos dinâmicos são testados.

Na primeira fase de planeamento, os melhores resultados são obtidos usando uma combinação do algoritmo A* e a Colónia de Formigas para optimizar a ordem dos pontos a visitar. Os campos potenciais são um algoritmo rápido e computacionalmente leve, sendo uma solução adequada para implementação em tempo real. É mostrado que os algoritmos têm uma performance razoável em vários cenários.

Palavras-chave: Aeronaves Remotamente Pilotadas, Prevenção de Colisões, Planeamento de Trajectória, Procura Heuristica, Optimizacao com Colónia de Formigas, Campos Potenciais, Curvas de Bezier.

Abstract

The development of Remotely Piloted Aerial Systems (RPAS) for civil applications has been rapidly growing over the past years. One of the main drawbacks in their integration into the National Airspace System (NAS) is the lack of a reliable collision avoidance systems. While sophisticated large RPAS, often military, have sensing technology and avionics on-par or even more advanced than manned aircrafts, on the other end of the spectrum of RPAS, the low-cost light-weight off-the-shelf drones, are deprived of almost any safety system. Given the exponential rise in the number of the latter platform types, typically operated by uncertified operators, there is an urgent need to develop low-cost collision avoidance systems that can be easily embedded in such platforms.

Among the vast topic of Sense and Avoid (S&A), this thesis addresses the problem of generating optimal paths for RPAS subject to maneuverability and collision avoidance constraints, and replanning them when in risk of collision. To achieve this task, a two-steps approach is proposed. In the first stage, classical path planning techniques are implemented to generate flyable paths in a known static environment. Grid based search algorithms, particularly the A* algorithm and Ant Colony Optimization (ACO), are used to find an optimal sequence of waypoints in a discrete environment. To ensure that the path is flyable and complies with curvature and safety constraints, an optimization of Rational Bezier curves is implemented.

While the first stage is performed before the mission execution, the second planning module, responsible for avoiding dynamic and static obstacles that were not previously known, is developed with the limitations of real-time implementation in mind. Potential Fields methods are used to adapt the initial path to avoid unexpected obstacles in real-time. It is assumed that the vehicle is equipped with an autopilot and a trajectory control unit.

Examples of path planning between a set of waypoints in a bidimensional and tridimensional environment with static obstacles, using the algorithms developed in the first stage, are presented. Several cases of path replanning, using the potential fields method, when encountering moving obstacles are also tested.

For the first path planning stage, the best results were obtained using a combination of the A* algorithm and ACO to optimize waypoint sequence. The Potential Fields method is fast and computationally inexpensive, being a feasible solution for real-time implementation. It is shown that the algorithms perform reasonably well in several scenarios.

Keywords: Remotely Piloted Aerial Systems, Obstacle Avoidance, Trajectory Planning, Heuristic Search, Ant Colony Optimization, Potential Fields, Bezier Curves.

Contents

Resumo iii

Abstract v

List of Tables ix

List of Figures xi

List of Symbols xvii

1 Introduction **1**

1.1 Motivation 2

1.2 Problem Scope 4

1.3 Objectives 5

1.4 Thesis Outline 5

2 Background **7**

2.1 Problem Overview 7

2.2 Algorithms Overview 10

2.2.1 Graph Search Algorithms 10

2.2.2 Ant Colony Optimization 11

2.2.3 Model Predictive Control 12

2.2.4 Mixed Integer Linear Programming 13

2.2.5 Potential Fields 13

2.2.6 Markov Decisison Processes 13

2.2.7 Velocity Obstacle 15

2.3 Final Remarks 16

3 Pre-Flight Path Planning **17**

3.1 Problem Statement 17

3.1.1 Configuration Space 18

3.1.2 Constraints 19

3.1.3 Cost Functions 20

3.2 Path Search 23

3.2.1 A* Algorithm 23

3.2.2 Ant Colony Optimization 28

3.3 Path Smoothing . . . 36

4 **Real-Time Path Planning** **41**

4.1 Safety Zones . . . 41

4.2 Rules of Air . . . 41

4.2.1 Avoidance Logic . . . 42

4.3 Collision Detection . . . 43

4.4 Potential Fields . . . 44

5 **Results** **51**

5.1 Pre-Flight Path Planning . . . 52

5.1.1 Minimum Distance Paths between Two Waypoints . . . 52

5.1.2 Minimum Energy Paths between Two Waypoints . . . 57

5.1.3 Minimum Distance Paths with Three Waypoints . . . 61

5.1.4 Path Planning in a Digital Elevation Model . . . 64

5.2 Path Replanning . . . 66

5.2.1 Head-on Scenario . . . 66

5.2.2 Converging Scenarios . . . 67

5.2.3 Missed Waypoint Scenario . . . 70

5.2.4 Replanning with Static and Moving Obstacles . . . 71

5.3 Final remarks . . . 72

6 **Conclusions** **73**

6.1 Achievements . . . 74

6.2 Future Work . . . 74

Bibliography **75**

List of Tables

2.1 Information provided by obstacle detection module for static obstacles. 9

3.2 WA* results for fixed-wing platform. 26
3.3 WA* results for multirotor. 27
3.4 Map characteristics. 32
3.5 ACO default values. 32
3.6 ACO selected parameters. 35
3.7 Results for minimum distance paths. 39

5.1 RPAS parameters. 52
5.2 ACO parameters. 52
5.3 Map characteristics. 52
5.4 Potential fields parameters. 52
5.5 Computational time and cost function values for the resulting Bezier curve. 64
5.6 Potential fields and vehicle parameters. 66
5.7 Replaning scenario parameters. 66
5.8 Head-on scenario results. 66
5.9 Right approach scenario results. 68
5.10 Left approach scenario results. 68
5.11 Climb scenario results. 69
5.12 Missed waypoint scenario results. 70
5.13 Replan with static and dynamic obstacles results. 71

List of Figures

1.1 EuroHawk at ILA Berlin Air Show. 1
1.2 Bayraktar Mini UAV of the Turkish Land Forces. 2
1.3 Proposed airspace model for RPAS operations. 3
1.4 Main RPAS subsystems and functions. 4

2.1 Flight mission with global and local trajectories[10]. 7
2.2 Hierarchical planning architeture. 8
2.3 Example of path planned by RRT[22]. 12
2.4 Ant Colony Optimization[23]. 12
2.5 Potential field representation[41]. 14
2.6 Path planned through the potential field method[41]. 14
2.7 (a) Collision cone, (b) Velocity obstacle[47]. 15

3.1 Pre-flight path planning module. 18
3.2 Grid Resolution 18
3.3 Expansion rules for multirotor platforms. 19
3.4 Expansion rules for fixed-wing platforms. 19
3.5 Free body diagram. 20
3.6 Bi-dimensional grid with start and goal states. Obstacles are represented by blue cells. . . 24
3.7 WA* path for w=1 and w=10. 27
3.8 Kinematic model representation. 31
3.9 Variation of α for ACS(left) and MMX(right). 33
3.10 Variation of β for ACS(left) and MMX(right). 33
3.11 Variation of number of ants for ACS(left) and MMX(right). 33
3.12 Variation of q_0 for ACS(left) and MMX(right). 34
3.13 Variation of ρ for ACS(left) and MMX(right). 34
3.14 Variation of $\tau/\tau_{min}/\tau_{max}$ for ACS(left) and MMX(right). 34
3.15 ACO flow chart. 35
3.16 Rational Bezier curves. 37
3.17 Path generated by optimized Bezier curve division. 39
3.18 Flowchart for the pre-flight path planning algorithm. 40

4.1 Safety zones defined around the obstacle. 42
4.2 Rules of the Air: (a)Head-on encounter, (b) Converging encounter, (c) Overtaking. . . . 42
4.3 Resolution for avoidance of static obstacles. 43
4.4 CPA representation. 44
4.5 Attractive potential field for a waypoint. 45
4.6 Attractive potential field for a path ($\alpha = 0.5$). 45
4.7 Simple repulsive potential flow. 46
4.8 Fading repulsive potential flow. 46
4.9 Swirling potential flow. 47
4.10 Example of obstructed waypoint. 48
4.11 Path replan flow chart. 49

5.1 Block diagram of tested system. 51
5.2 Minimum distance paths for the A* and ACO algorithms. 53
5.3 Planned paths for the Potential Fields algorithm. 54
5.4 (a) Processing time of the several algorithms and (b) path cost(distance) for the fixed-wing case represented in Figs. 5.2(a) and (b) and Fig. 5.3(a). 55
5.5 (a) Processing time of the several algorithms for the multirotor platform and (b) path cost (distance) for the fixed-wing and multirotor cases represented in Figs. 5.2(c) and Fig. 5.3(b). 56
5.6 Minimum energy paths for the A* and ACO algorithms, for both fixed-wing and multirotor platforms. 57
5.7 (a) Processing time of the several algorithms and (b) path cost (energy) for the case represented in Figs. 5.6(a) and(d). 58
5.8 (a) Processing time of the several algorithms and (b) path cost, evaluated according to distance for the case represented in Figs. 5.6(b) and(e). 59
5.9 (a) Processing time of the several algorithms and (b) path cost, evaluated according to distance for the case represented in Figs. 5.6(c) and(f). 60
5.10 Minimum distance paths. (a),(b): A* with random waypoint order;(c),(d): ACS and MMX with random waypoint order; (e),(f): A* with optimal waypoint order given by ACO. . . 61
5.11 (a) Processing time of the several algorithms and (b) path cost(distance) for Figs. 5.10 (a), (c) and (e). 62
5.12 (a) Processing time of the several algorithms and (b) path cost(distance) for Figs. 5.10 (b), (d) and (f). 63
5.13 Minimum energy path planned for a fixed-wing platform: (a) 3D view, (b) 2D view. . . . 64
5.14 Minimum distance path planned for a fixed-wing platform: (a) 3D view, (b) 2D view. . . 65
5.15 Head-on encounters. 67
5.16 Right approach converging conflicts. 67
5.17 Left approach encounters. 68
5.18 Climb encounters. 69

5.19 Converging encounter with missed waypoint. 70

Acronyms

ACAS Airborne Collision Avoidance System

ACO Ant Colony Optimization

ADS-B Automatic Dependent Surveillance-Broadcast

ATC Air Traffic Control

BVLOS Beyond Visual Line of Sight

CPA Closest Point of Approach

DEM Digital Elevation Model

FCS Flight Control System

GNSS Global Navigation Satellite System

MDP Markov Decisison Process

MILP Mixed Integer Linear Programming

MPC Model Predictive Control

NAS National Airspace System

RPAS Remotely Piloted Aircraft System

S&A Sense and Avoid

TCAS Traffic Collision Avoidance System

UAV Unmanned Aerial Vehicle

VLOS Visual Line of Sight

List of Symbols

Greek Symbols

α	Weight of pheromone value	
β	Weight of heuristic value	
Δt	Time step	s
$\Delta x, \Delta y, \Delta z$	Grid resolution	m
η	Heuristic value	
γ	Flight path angle	°
ψ	Heading angle	°
ρ	Pheromone evaporation rate	
ρ_{air}	Air density	kg/m^3
τ	Pheromone value	
θ	Pitch angle	°

Roman Symbols

$\mathbf{D}$	Direction of motion	
$\mathbf{F}$	Force vector	N
$\mathbf{M}$	Total number of path points	
$\mathbf{O}$	Obstacle parameter vector	
$\mathbf{P}$	Path point	
$\mathbf{P_{obs}}$	Obstacle position	
$\mathbf{P_{rpas}}$	RPAS position	
$\mathbf{WP}$	Control Point	
$\mathbf{WP}$	Waypoint	

$\mathbf{S_{dir}}$	Swirlling direction	
B_c	Constrained Bezier curve	
B_i	Bezier curve	
B_o	Optimized Bezier curve	
C_D	Drag coefficient	
D	Drag	N
d_{CPA}	Distance to closest point of approach	m
$E_{supplied}$	Supplied energy	J
F_c	Cost function	
F_d	Distance cost function	m
F_e	Energy cost function	J
F_{at}	Atractive potential	
F_{rep}	Repulsive potential	
F_{tot}	Total potential force	
g	Gravitational acceleration	m/s^2
H	Obstacle height	m
h	Flight height	m
k	Curvature	
L	Lift	N
m	Mass	k
N	Total number of waypoints	
n	Order of Bezier curve	
p	Transition probability	
P_{close}	Closest point on path	
P_{next}	Next point on path	
R_a	Action radius	m
R_c	Collision radius	m
R_d	Detection radius	m

R_{curv}	Minimum turning radius	m
$R_{obstacle}$	Obstacle radius	m
R_s	Safety radius	m
S	Wing span	m
T	Thrust	
t_{CPA}	Time to closest point of approach	s
U	Mechanical work	J
v	RPAS velocity	m/s
V_{air}	Air velocity	m/s
V_{ground}	Ground velocity	m/s
V_{wind}	Wind velocity	m/s
W	Weight	N
w	Weight of weighted A* algorithm	
w_B	Weight of rational Bezier curve	
x, y, z	Cartesian coordinates	
$\mathbf{d}_o$	Relative distance to obstacle	m

Superscripts

c	Cross-track component
i, j, k	Computational indexes
max	Maximum
min	Minimum
t	In-track component

Chapter 1

Introduction

As it happens with many technologies, the development of Unmanned Aerial Vehicles (UAVs), also referred to as drones, was potentiated by the military field and can be dated back to World War I. As the utility of unnamed aircraft for ”dull, dirty and dangerous” missions became apparent, the use of UAVs has extended from the military field to the civil realm. Compared to manned aircraft, UAVs provide many benefits in terms of cost and safety. They allow for longer operation times, are easily adjustable to any type of terrain, some of the platforms are cheaper to operate, require less maintenance and have no need for an on-board human pilot. Nowadays, there is a wide range of available platforms which can go from vehicles heavier than some manned aircraft, as in Fig. 1.1, to miniature platforms that can be carried by an individual, as seen in Fig. 1.2.

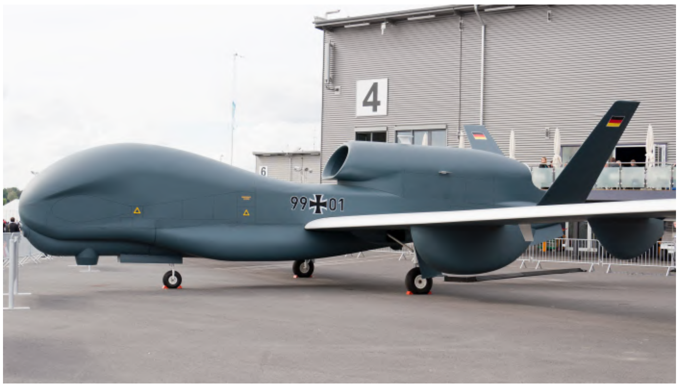


Figure 1.1: EuroHawk at ILA Berlin Air Show[1].

For UAVs to be integrated into non-segregated airspace, appropriate airworthiness standards must be defined. With that in mind, the classification of these vehicles is important from the regulatory point of view since the rules defined for drone operations will likely be different for distinct categories. To this day there are no consensus in the classification of UAVs for certification purposes. Their categorization generally follows a combination of parameters such as maximum take-off weight, range, endurance and altitude.

A Remotely Piloted Aircraft (RPA) is a subset of the UAV class, that refers to an aircraft that while it does not carry a human operator on-board is flown by a remote pilot. The pilot monitors the aircraft

Figure 1.2: Bayraktar Mini UAV of the Turkish Land Forces[2].

constantly and has direct responsibility for the safe conduct during flight. A Remotely Piloted Aircraft System (RPAS), besides the RPA, also includes the remote pilot station, the necessary communication links and any other elements needed for the flight operation, such as a launch platform. The RPA may possess some form of autopilot technology but is not a fully autonomous system, even though that today there are RPAS that can fly autonomously, not all of its systems are certified and can ensure the necessary safety standards, so supervision is always required. The majority of UAVs today belong to this category.

Current civil applications for RPAS include infrastructures and traffic monitoring, search and rescue operations, geographical surveys, environmental and meteorological observation, aerial photography, communication services, entertainment and cargo transport operations among many other. Many of these tasks can be achieved with low-cost light-weight and off-the-shelf drones, that usually have no significant security system and many of them are operated by uncertified operators. Given the exponential rise in the use of such platforms and considering that they share the same airspace with manned aviation, there is an urgent need to guarantee that such platforms have appropriate means of ensuring a safe flight.

1.1 Motivation

To allow the safe emergence of RPAS operations in today's airspace the current safety requirements for aircraft operations must not be compromised. In order to ensure a safe flight, the pilot in command is required to maintain vigilance, or be provided with sufficient information, to see and avoid any conflicting traffic while complying with the rules of the air. To this day there are no rules that relieve the pilot from this task, thus for RPAS to gain access to the National Airspace System (NAS), they will need to be equipped with an equivalent form of Sense and Avoid (S&A) technology[3]. When flying in controlled airspace, RPAS must comply to the rules of the air and respond to Air Traffic Control (ATC) instructions.

Nowadays regulations on the use of small drones vary among countries, however the majority allows only for Visual Line of Sigh (VLOS) operations. To achieve its full potential, RPAS are expected to operate in increasingly complex environments and be able to conduct Beyond Visual Line Of Sight (BVLOS) operations.

In Portugal the RPA can only perform flights during the day in VLOS and up to 120 meters, except in restricted areas, such as airports and military installations The RPA must also fly with identification lights, and the pilot is responsible for maintaining a secure distance from people and property, ant it must always give way to manned aircraft. Platforms with a weight bellow 250 grams cannot exceed a flight height of 30 meters[4].

Most commercial applications today occur in uncontrolled airspace which goes from the ground up to around 400m. With the growing demand for vehicles that can operate BVLOS, new air traffic systems must be developed. Amazon has proposed a segregated airspace model for operations bellow 500 ft[5], as seen in Fig. 1.3. In this model four main zones are defined: a low speed area for terminal operations and vehicles without sophisticated S&A functions, a high speed transit zone for certified vehicles, a no-fly-zone and a predefined low risk locations established by aviation authorities.

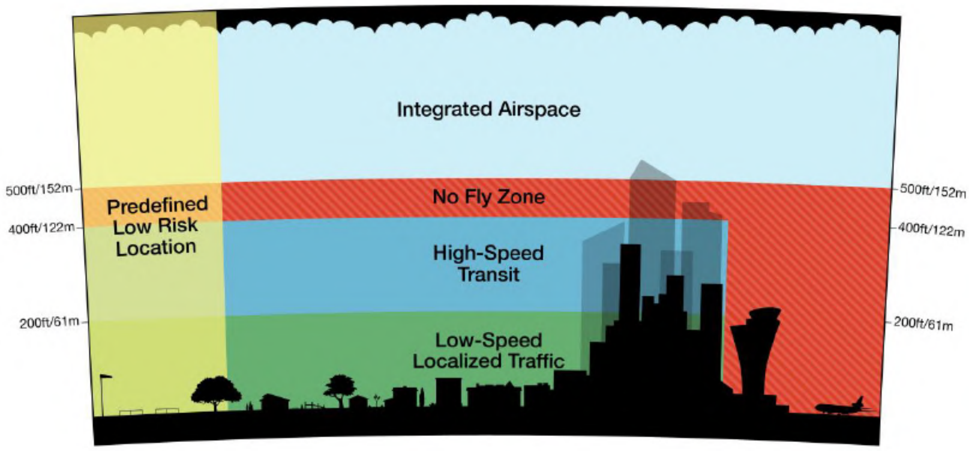


Figure 1.3: Proposed airspace model for RPAS operations[5].

NASA is also conducting research on a Unmanned Aerial Vehicle Traffic Management (UTM) system that could provide integration requirements for enabling safe, efficient low-altitude operations[6]. They propose two subsystems for the UTM system. The first would be portable and adaptable to different areas to provide support for agricultural operations and emergency scenarios. The other would be a persistent system for low altitude operations. This system relies on frequent communication and surveillance to track and monitor traffic.

The European Aviation Safety Agency (EASA) also published its concept of operation for drones[7]. It follows a risk based approach to propose three categories of operations. The OPEN category, which does not require authorization by the aviation authorities would allow the operation of vehicles up to 150 meters, under VLOS and outside reserved areas. The SPECIFIC category would encompass riskier operations and need approval of the national aviation authorities. To operate in this category some sort of S&A is required. Finally the CERTIFIED category, where the aircraft involved would be treated in the same manner as manned aviation.

The development of a reliable low-cost path planning and S&A systems that can encompass the wide range of operations that RPAS can conduct, and the diversity of existing platforms remains one of the major challenges in the research community. Even though several efficient planning and collision avoidance solutions have already been proposed, their application to the characteristics of unmanned vehicles is still a challenge. This thesis addresses the current challenge of the development of a reliable S&A systems that can be integrated in small low-cost platforms.

1.2 Problem Scope

To this day there are no certification standards for the use of unmanned aircraft but, in order to achieve its full potential, RPAS will have to be able to autonomously maintain a minimum safety level. Autonomy applies to many levels, as seen in Fig. 1.4, and refers to the vehicle ability of sensing, perceiving, analyzing, planning and acting to achieve the goals assigned by a human operator[8]. Among the several autonomy functions, the focus of this work will be on the path planning and collision avoidance functions.

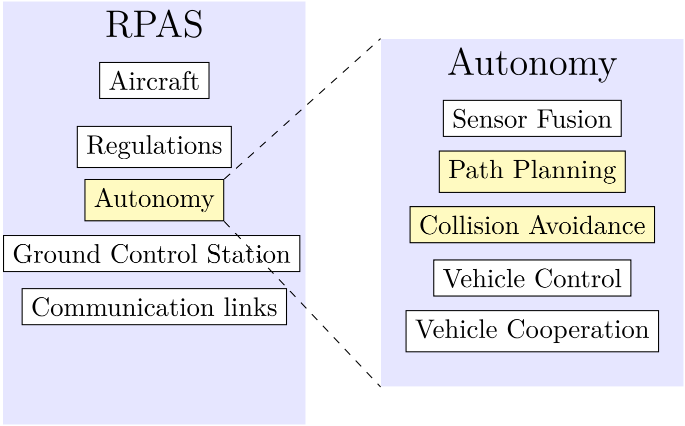


Figure 1.4: Some of the main RPAS subsystems and functions.

Sense and Avoid is a layered solution. First the Sense stage is responsible for the acquisition of the necessary information to detect threatening situations. Once this information is available, the avoidance stage follows, where when in risk of collision, an appropriate evasive maneuver is determined. It is assumed that exists a functioning sensing module that provides the necessary information. Even though cooperative systems for UAVs traffic management are being developed[6], not all platforms will be able to carry the necessary equipment to maintain communication so a solution for uncooperative vehicles will be considered.

The collision avoidance problem itself can be formulated in two ways: as a path planning problem or as a collision avoidance maneuvering problem [9]. In the first case, the problem is to plan a feasible path between two points while avoiding obstacles present in the way, while the latter addresses the issue of performing avoidance maneuvers as obstacles are detected. This can also be seen as a problem of replanning a given path. As many missions will take place in a known environment, the mission to execute can be planned previously to the flight execution, eliminating the need to recalculate the entire path and only performing local variations. This reduces the computational load of the operations to be performed in real-time (online), which is critical for small platforms with limited processing power.

1.3 Objectives

The goal of this work is to develop path planning and replanning algorithms that can be applied to small RPAS. The problem is split into several sub-goals to achieve:

- Develop a path planning algorithm to be implemented as an offline solution to find a safe and optimal path in a static environment;
- Develop a collision avoidance algorithm that can be implemented in real-time and avoid dynamic obstacles by replanning the reference path;
- Evaluate the performance of the proposed algorithms in representative case scenarios to determine the feasibility and drawbacks of the proposed methods.

The developed solution is expected to be as generic as possible so that it can be applied to platforms with different dynamics without major modifications. The two main types of small RPA platforms considered are multirotors and fixed-wings, moving in an urban or orographic environment. Two main situations are addressed: planning between a set of waypoints in a two or three-dimensional static environment considering the optimization of a given cost function and some platform constraints, and replanning the initial path when an uncooperative moving obstacle, or an unknown static obstacle, is detected.

1.4 Thesis Outline

This thesis consists of six chapters, organized as follows:

- **Chapter 1** introduces the study topic, provides the motivation for the research and defines the research goals of the work;
- **Chapter 2** presents an overview of S&A systems and current implementations. The proposed system framework is presented and the main techniques used in path planning and collision avoidance are reviewed;
- **Chapter 3** addresses the RPA path planning problem. Two graph search algorithms, A* and ACO, are used to generate a collision free path as a set of waypoints. Once the optimal path has been determined, Bezier curves are used to produce a smooth flyable path that accounts for the vehicle turning radius and safety constraints;
- **Chapter 4** addresses the collision avoidance problem during the mission execution stage. A collision avoidance strategy is devised based on the potential fields method while considering the Rules of the Air;
- **Chapter 5** evaluates the algorithms performance in several scenarios;
- **Chapter 6** presents the conclusions taken from the obtained results, an analysis of the algorithms limitations and provides recommendations for future work.

Chapter 2

Background

In this chapter, a background on the scope of S&A systems is provided, as well as a review of some of the literature on the topic.

2.1 Problem Overview

Path planning and obstacle avoidance are crucial tasks of mission management and flight planning for RPAS. The developed solution is expected to account for both static and dynamic obstacles. Some of these obstacles are generally known a priori, so the proposed solution will follow a classic two layered architecture. In the first stage, the global planning module, which assumes a known static environment, determines a collision free path from a given start to goal configurations. This task is performed prior to the flight execution. The RPA then follows this path and as new threats are detected by the on-board sensors, the local planning module must replan the path while avoiding the new obstacles. Figure 2.1 represents a flight scenario where the global trajectory must be adjusted to avoid a collision with an unexpected intruder.

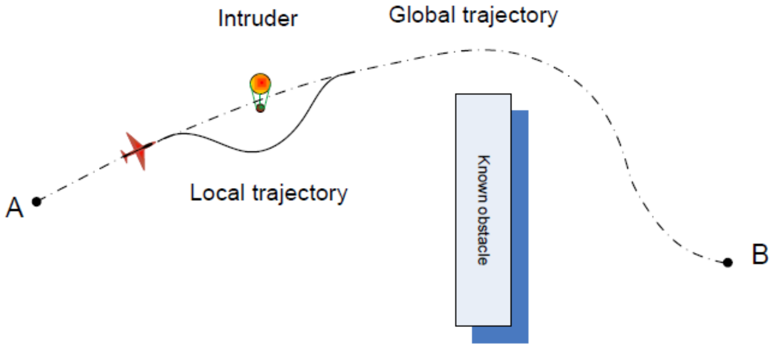


Figure 2.1: Flight mission with global and local trajectories[10].

The two-stage planning architecture allows the global and local solutions to compensate each other in terms of planning horizon, considered obstacles, vehicle and mission constraints. The generated path consists of a series of waypoints, which are uploaded to the Flight Control System (FCS) providing the reference for the tracking system, where the low-level control modules deal with the task of moving the RPA to a specific position. The reference path can be tracked using either path following or path tracking techniques. The objective of path following is to minimize the position error, forcing the control output to follow a path with no time dependence. Path tracking requires the vehicle to follow a time and space reference trajectory, not being as flexible as the path following approach.

When in-flight, the issue of avoiding new obstacles can be handled in different manners. The pre-computed reference path can be replanned when the obstacle is detected and a new collision free path is provided to the path tracking/following module. Instead of replaning the path, collision avoidance can be included in the lower level control module and, as an alternative to defining reference positions, the avoidance is made by providing directly control inputs, or velocity and attitude (pitch, roll, yaw) commands to the autopilot.

This work will focus on the high-level planning modules assuming that lower level controllers are available to handle vehicle dynamics and guide the vehicle to the reference waypoints. The scheme in Fig. 2.2 illustrates the proposed collision avoidance system framework.

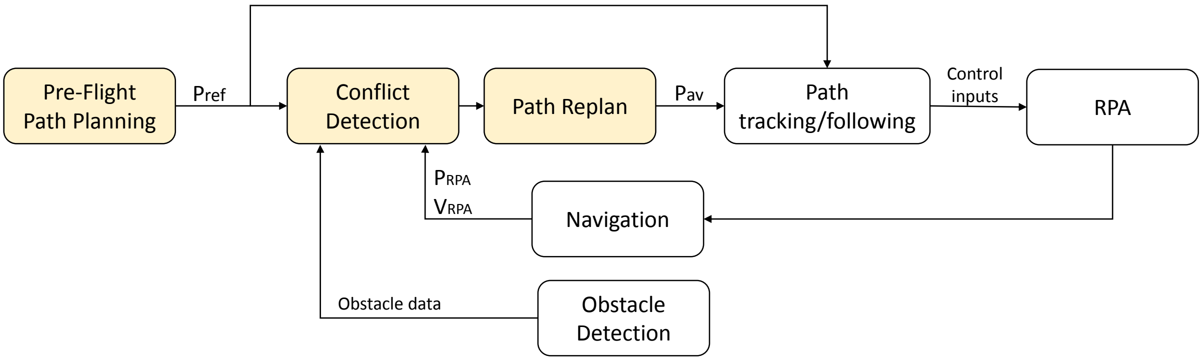


Figure 2.2: Hierarchical planning architeture.

The **pre-flight path planning** module receives a set of waypoints during the mission planning stage, a known obstacle database and a terrain map. If obstacles are found in the segments that connect the given waypoints, a free path P_{ref}, that satisfies certain optimization criteria, such as minimum distance, is provided in the form of a waypoint matrix.

During mission execution the **path tracking/following** is responsible for following the reference path. While the path is executed, the **navigation** system provides the current RPA position (P_{rpa}) and velocity (V_{rpa}) to the conflict detection module, which also receives information from the **obstacle detection** system. In this work, the type of sensors used will not be specified, but it is assumed that there is a working method of sensor fusion to obtain the necessary information about the environment. To develop the collision avoidance algorithms, this information will be provided in the form of a table. For the purpose of collision avoidance, a safety volume is defined around the obstacles. Due to its simplicity and ability to encompass a wide variety of obstacle types, a cylindrical model is used to represent obstacles.

Obstacles	X	Y	Z	$R_{obstacle}$	$H_{obstacle}$	$\|d_o\|$
1	.	.	.	.	.	.
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2.1: Information provided by obstacle detection module for static obstacles.

In Tab. 2.1, the information provided for known static obstacles is presented. This information includes the obstacle base position (X,Y,Z), the protection volume defined by a radius $R_{obstacle}$, and height $H_{obstacle}$. From these values, and knowing the RPAS position from the navigation module, the distance to the obstacle, $\mathbf{d}_o$, is determined. This distance will be used to rank the table to prioritize closest obstacles. This ranking is not necessary for the offline planning stage but it will be important during real-time implementation. For moving obstacles, that will be detected during the mission execution, the same information is provided and the intruder velocity is also included. For this type of threats, the relative distance is not enough to evaluate the urgency with which the threat must be addressed hence the time to collision will also have to be considered. When new obstacles are detected in the sense table, the **conflict detection** algorithm will determine if there is a collision risk. If a collision is eminent, the data from the navigation and obstacle detection modules is passed to the **path replan** segment and a new avoidance path P_{av}, is computed.

The effectiveness of the collision avoidance system will depend on how fast and accurately the aircraft can detect obstacles. The information the RPAS needs about the environment to do so can be obtained from on-board sensors, transmitted from ground control stations, other aircrafts or can also be given by a pre-loaded map or data set. Depending on the way the information is obtained, the sensing process will be either cooperative or non-cooperative. The cooperative category requires a communication link between agents. Using devices such as transponders, aircrafts can exchange information like heading, speed, position and assigned waypoints. Currently, Traffic Collision Avoidance System II (TCAS) is the available and required implementation of Airborne Collision Avoidance System (ACAS) for manned aircrafts that complies with ICAO standards[11]. This system works by interrogating the transponders of surrounding aircrafts and from their reply the aircraft tracks the intruders range, bearing and altitude. This system provides two types of alerts: traffic advisories and resolution advisories. Traffic advisories are used to indicate a possible threat to the pilot. Resolution advisories are recommendations of maneuvers to provide separation from other aircrafts and avoid possible collisions. Advisories such as climb or descent are presented on a display and accompanied by an auditory warning. TCAS II requires two antennas (one on top and other on the bottom of the vehicle) and is able to identify traffic 14 miles to the front of the aircraft. When considering small platforms with strict payload limitations, this system cannot be applied.

Automatic Dependent Surveillance-Broadcast (ADS-B) is another widely used cooperative surveillance technology. Unlike TCAS, ADS-B does not require an interrogation signal to transmit its information. The aircraft obtains its position from a Global Navigation Satellite System (GNSS) constellation and then simultaneously broadcasts it to ground stations and properly equipped aircrafts. This system allows a greater precision in the position determination. Besides its position information of velocity, identification and intent is also broadcasted. This system is an attractive solution for RPAS use since it provides reliable information and is an element of the NextGen Air Transportation system and the Single European Sky ATM Research meant to replace TCAS[12]. This system however is not effective when considering ground obstacles, such as terrain or buildings.

Aircraft not equipped with any of these devices rely on non-cooperative sensing to detect obstacles. Among the non-cooperative technologies, the sensing principle can be active or passive. Active sensors, such as radars and lasers, emit a signal that is reflected by the obstacle allowing its detection. Passive sensors depend on the reception of signals emitted by the obstacle itself. They include electro-optical, infra-red and acoustic technologies.

There are many design factors to be considered during the development of a collision avoidance and path planning system. This work will address path planning between several waypoints at constant and varying height, considering a Digital Elevation Model (DEM) to provide terrain data and cylindrical obstacles. The replanning stage will take into account non-cooperative moving obstacles, while maintaining maximum proximity to the reference path and considering the Rules of the Air, which define the guidelines for the procedures to be followed in air navigation.

2.2 Algorithms Overview

In this section an overview of the main contributions to the collision avoidance and path planning problems is presented and the benefits and drawbacks of each method are highlighted.

2.2.1 Graph Search Algorithms

Graph search algorithms are one of the most popular methods used in robot path planning. In this solution the search space is represented by a set of cells on a weighted graph. The robot configuration is represented by the graph nodes while the edges connecting the graph represent the cost of moving between nodes. High weight values on cell edges can be used to represent obstacles. Most of the graph search algorithms are based on the Dijkstra algorithm [13]. This algorithm works by evaluating the cost of moving from one node to its neighbors and connecting the nodes with the lowest moving cost. Best-first-search[14] is another algorithm which works in a similar way to Dijkstra's, but it uses an estimate of the distance from the current node to the goal, choosing the node that is closer to the final position, thus reducing the search time. The A* algorithm[15] builds on the latter two by combining the moving cost between nodes and the cost to reach the goal to drive the search. The problem with these algorithms, due to its graph structure, is that they usually calculate paths with unnecessary heading changes. Generally post-processing techniques must be applied to obtain a flyable path. Dynamic A* improves on the A* algorithm making it applicable to dynamic environments by allowing the edges cost to be dynamically increased or decreased. It has proven to be applicable to real-time scenarios[16]. Theta*[17] is meant to deal with the problems of heading constraints in a graph structure by considering different connections between graph nodes. Paths obtained by T* are smoother than those determined by A*. Both algorithms were successfully applied to the UAV path planning problem[18]. A major downfall when applying typical graph search algorithms to path planning problems is lack of inclusion of the vehicle kinematics. In [19] node connection rules are defined to include the vehicle kinematics. A small fixed wing UAV is considered, and the grid size in the x and y directions is defined by the minimum turning radius, while the size in z is defined by the maximum climb angle. In [20] a new algorithm, Kinematic A* is presented for UAV path planning. KA* includes a simple kinematic model of the vehicle to evaluate the moving cost between cells in a 3D environment. Vehicle movements are limited by the minimum turning radius and maximum rate of climb. These algorithms are attractive because they are easy to implement and can be adapted to different case scenarios by allowing distinct heuristics to guide the search. However, the main drawback is the lack of correlation between aircraft dynamics and the planned path. Depending on the search space size, the computational load can be heavy.

Rapidly Exploring Random Tree (RRT)[21] is a popular search algorithm when dealing with high dimensional spaces. Like the Probabilistic Roadmap (PRM), is a sampling based motion planning algorithm. The general idea is to connect points sampled randomly from the search space. In PRM, a roadmap is computed a-priori, i.e. a graph that represents a set of collision free paths, and then the shortest path that connects initial and final states is calculated. This algorithm is probabilistically complete which means that the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap. RRT is also probabilistically complete. The algorithm is initialized with a graph that contains only the initial state as a single vertex. At each step, a random sample from the search space is generated and an attempt is made to connect this new sample to the nearest vertex on the tree. This connection is limited by a growth factor (generally distance). If the connection is successful the new sampled point is added to the graph. By increasing the probability of sampling states in a specific area, the search is guided towards the goal state. The higher this probability the greedier the algorithm. In [22] a greedy version of closed-loop RRT is used to plan collision free paths. The method is cooperative and collision with other aircrafts is predicted based on the RPAS current flight route and the intruders ADS-B data. A path planned through the RRT algorithm is presented in Fig. 2.3.

2.2.2 Ant Colony Optimization

The ACO algorithm is a metaheuristic approach inspired by the foraging behaviour of ants in the real world[23]. Ants are known to find the shortest path from their nest to the food source. They accomplish this by leaving a pheromone trail along their path. This trail is detected by other ants which will tend to follow the trail with higher pheromone levels, as seen in Fig. 2.4. This solution has been applied to solve several problems, such as network routing in telecommunications networks, the Travelling Salesman Problem, Job Scheduling and robot path planning [24] [25] [26] [27] [28]. The ACO formulation has also been applied to the UAV path planning problem [29] [30]. These solutions, however, are only applied to two-dimensional environments which is not suitable for many applications of flying vehicles. ACO has shown good results in planning paths but the computation time is prohibitive for online applications.

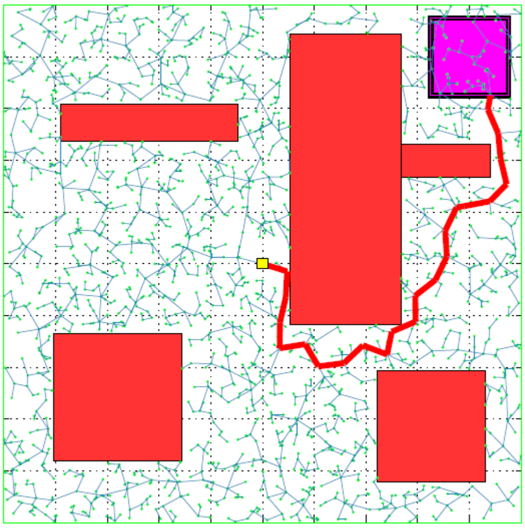


Figure 2.3: Example of path planned by RRT[22].

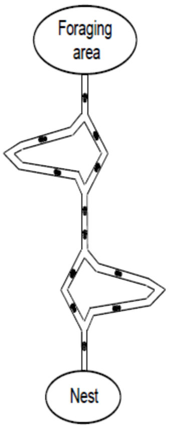
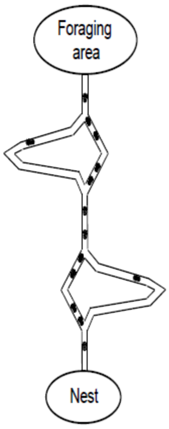



Figure 2.4: Ant Colony Optimization[23].

2.2.3 Model Predictive Control

Model Predictive Control (MPC), also known as Receding Horizon Control (RHC), is a method in which the trajectory optimization problem is solved at each time step through feedback control[31]. This solution was initially devised for systems with slow dynamics like chemical processes. MPC techniques allow the inclusion of vehicle kinematics and dynamics on the optimization problem to produce feasible trajectories. Recently, this method has been used to solve multiagent control problems[32]. MPC strategies are based on a finite-horizon optimization of a given model. At each time step, the model state is sampled and a numerical optimization process is used to compute the control input that minimizes some given cost. The method requires a dynamic model of the system, information on past control inputs and a cost function to be optimized over a defined prediction horizon. In [33] predictive control is used for the control of decentralized cooperative UAVs in a three-dimensional environment. A set of optimal points are calculated to ensure that the vehicle arrives at the required position without colliding with static and moving obstacles. The computation time was proved acceptable for real-time implementation. In

[34] MPC was used for collision free formation flight. Obstacle avoidance is achieved by defining a cost function based on the UAVs velocity orientation and relative distance between the vehicle and the obstacle, associated with a priority strategy.

2.2.4 Mixed Integer Linear Programming

MILP is an evolution of MPC algorithms. MILP is a technique that addresses the collision avoidance problem using linear programing[35]. A MILP standard problem has the following form:

$$\begin{aligned} \min \quad & f(x) = c^T x \\ \text{subject to} : \ & b_L \leq Ax \leq b_U \\ & x_L \leq x \leq x_U \end{aligned}$$

where $A \in \mathrm{I\!R}^{\mathrm{m \times n}}; \mathrm{c, x, x_L, x_U} \in \mathrm{I\!R}^{\mathrm{n}}; \mathrm{b_L, b_U} \in \mathrm{I\!R}^{\mathrm{m}}$. A subset of variables $x_I \subset x$ are restricted to be integer values, $x_I \in \mathbb{Z}^{n_I}$. To solve the MILP problem, a set of constraints to model the UAV dynamics must be specified. Additionally constraints related to collision avoidance, such as separation distance, must be provided. Once the problem is formulated several commercial solvers are available to determine the solution. This approach however is computationally expensive and generally used for offline planning, but some attempts to decrease solution time have showed this method reliability as an online path planner[36]. It also scales poorly with the problem size. This formulation has been widely applied to the collision avoidance problem of UAVs [37] [38]. MILP works on the planning level not on the control level.

2.2.5 Potential Fields

The Artificial Potential Field methods[39] are an approach inspired by physical potential fields. These methods are generally used for reactive collision avoidance systems [40]. In this approach, the robot is treated as a charged particle moving through a field induced by attractive and repulsive forces. Waypoints to visit are associated with attractive forces and repulsive forces are placed around obstacles. The magnitude of the repulsive force will depend on the distance between UAV and the obstacle. The sum of the attractive and repulsive forces yields a global potential field which leads the robot towards the goal by following the lowest potential value. An example of a potential field representation and the originated path can be seen in Fig. 2.5 and Fig. 2.6. These algorithms give smooth flyable paths avoiding both static and dynamic obstacles. When compared to other approaches, using this method is not computationally expensive, so it can be used in real-time. It is also a common approach for formation flights with collision avoidance capabilities[42].

2.2.6 Markov Decisison Processes

In this approach, the collision avoidance problem is formulated as the optimal control of a stochastic system. A MDP is a discrete dynamic programing process where the state of the system changes according to the current state and the chosen action[43]. A MDP is defined by a tuple $(S, A, P^a_{ss'}, R^a_{ss'}, \gamma)$ where S represents the state space, A is the action space, $P^a_{ss'}$ indicates the probability of transiting from state s

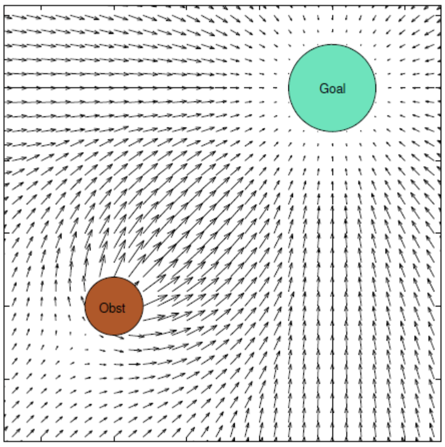


Figure 2.5: Potential field representation[41].

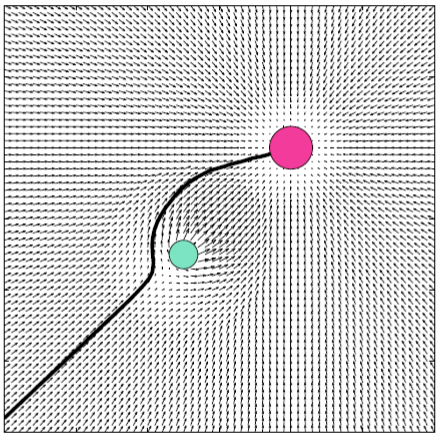


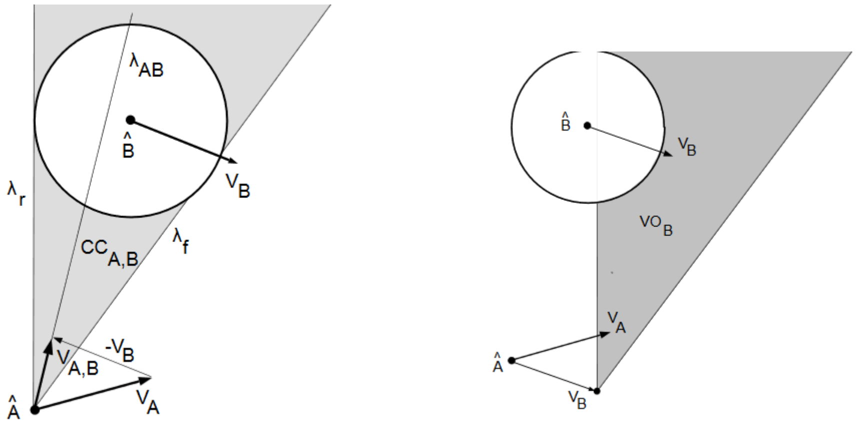
Figure 2.6: Path planned through the potential field method[41].

to s' haven taken action a and receiving the reward $R^a_{ss'}$, and $\gamma \in [0,1]$ is the discount factor used to favor early rewards against future ones. The problem is solved through dynamic programing by evaluating the expected reward for each combination of states and actions. When considering uncertainties that may arise from the available sensors and intruder behavior, a Partially Observable MDP (POMDP) can be used[43].

The MDP approach allows to account for all possible encounter scenarios and their likelihood when solving the model, so it is an attractive approach because it allows the automatic generation of the collision avoidance logic by defining the system goal and operating environment. If a proper MDP model has been developed, it can also be adapted to different RPAS models and sensor arrangements. For a collision avoidance system, the state must contain at least the 3D positions and velocities of at least two aircrafts and, in order to build an optimal policy, all possible combinations of actions-state must be considered. Large state spaces are the main issue when solving MDPs, as some formulations would require too much computational complexity to solve the problem. Examples of POMDPs applied to RPAS collision avoidance problem can be found in [44], where a two-dimensional formulation assuming a discrete state space is presented for offline planning. In [45] the problem is extended to three-dimensional continuous-state models. To this day, this approach is only suited for offline planning. This optimization process is also the solution being considered for the Next-Gen Airborne Collision Avoidance System (ACAS-X) being developed by the FAA that is meant to eventually replace the Traffic Collision Avoidance System (TCAS). Several variants of this system are being considered to extend collision avoidance protection to users and aircraft classes that currently not benefit from TCAS. The version ACAS-Xu will be specifically designed for RPAS [46].

2.2.7 Velocity Obstacle

Velocity Obstacles are a successful approach to collision avoidance in a dynamic environment. Velocity Obstacles represent the set of robot velocities that would result in a collision with an intruder moving at a given velocity at some future point in time. If the obstacle is static, the set of velocities is referred to as collision cone. A representation of the collision cone can be seen in Fig. 2.7(a), where the collision cone $CC_{A,B}$ induced by an agent B traveling at V_B, in an agent A traveling at V_A, is constructed by setting a safety distance λ_{AB} around the intruder and constructing a cone by defining two lines, λ_r and λ_f, that touch the safety circle at a single point point. A geometric representation of a velocity obstacle VO_B can be seen in Fig. 2.7(b), where the apex of the collision cone is at V_B. The idea to generate an avoidance maneuver is to choose an avoidance velocity vector that is out of the velocity obstacle. To make sure that the resulting avoidance maneuver is dynamically feasible, the set of avoidance velocities is intersected with the admissible velocities, defined by the robot's acceleration constraints. The trajectory is then chosen by searching a tree of feasible velocities at each time step [47]. This method was initially developed for ground vehicles but have since been applied to flying robots[48]. In [49], a tridimensional extension of the Velocity Obstacle method is presented to reactively generate an avoidance maneuver for UAVs, by changing the vehicle velocity vector based on the encounter geometry. In [50] these solution is applied to solve conflicts between UAVs by determining a reference velocity which results in a maneuver compliant with the Rules of the Air. Velocity Obstacles are also used to generate cooperative maneuvers to solve conflicts between UAVs [51]. This approach is made to deal with dynamic environments and is computationally inexpensive, being used to solve conflicts in real-time.



(a) (b)

Figure 2.7: (a) Collision cone, (b) Velocity obstacle[47].

2.3 Final Remarks

Path planning in a dynamic tridimensional environment is a challenging task. A generic path planning and collision avoidance strategy should be able to encompass many different scenarios and types of platforms.

The A* algorithm is regarded as one of the most practical an efficient path planning algorithms. It is easy to implement and can be adapted to different scenarios by allowing different heuristics. Some kinematic constraints of the vehicle can also be taken into account and its real-time implementation has been proven, so this method is considered as a first approach to our path planning problem. As it will be seen, this method has some limitations namely when wanting to optimize the order of a set of waypoints. A better choice for this sort of problems would be ACO, so this solution will also be considered. For the second stage, regarding collision avoidance in real-time, the concept of Potential Fields is investigated as its low computational requirements make it an attractive choice for implementation in platforms with limited processing capabilities.

Chapter 3

Pre-Flight Path Planning

In this chapter, the problem of planning an optimal path for a given RPAS platform in a static environment is addressed. As the focus is on the higher level of the planning process, two simple models for the RPAS platforms will be considered and it is assumed that low-level controllers are available to handle vehicle dynamics. The two types of RPAS platforms considered are multirotor and fixed-wing. The cost of the path will be evaluated according to two criteria: distance and energy. First, the configuration space is represented as a weighted graph and two graph search algorithms are studied: A* and ACO. Since a path planned in a discrete environment is of no practical interest for a vehicle that moves in a continuous one, Bezier curves are used to produce a smooth path.

3.1 Problem Statement

The fundamental problem of path planning consists of finding a sequence of actions for an agent that can take it from one location to another while avoiding any obstacles on the way. This work addresses the problem of generating an optimal path for a single vehicle operating in a static environment that can easily be tracked by low level controllers. It is generally also required that the planned path achieves the goal while optimizing a given cost.

The waypoints given to the path planner which are the reference points of obligatory passage are denoted as $\mathbf{W}\mathrm{P}_i \in \mathbb{R}^n$ for $i \in \{1, \dots N\}$ where N is the total number of waypoints, and $n = 2/3$ defines the problem dimension. The points generated by the coarse path planner between initial and final waypoints that will later be used to the path smoothing process are denoted as control points, $\mathbf{C}\mathrm{P}_i \in \mathbb{R}^n$ for $i \in \{1, \dots M\}$ where M is the total number of path points.

The minimum safety distance between RPAS and obstacles is defined as $R_{safe} \in \mathbb{R}$ and accounts for the vehicle dimensions and an extra safety distance for perturbations that may occur during path execution. The obstacles are defined as represented in Tab. 2.1, $\mathbf{O_j} \in \mathbb{R}^n$ for $j \in \{1, \dots P\}$ where P is the total number of obstacles.

The inputs and outputs of the pre-flight path planning module are depicted in Fig. 3.1. Path planning involves three main sub-problems: environment modeling, path search and path execution.

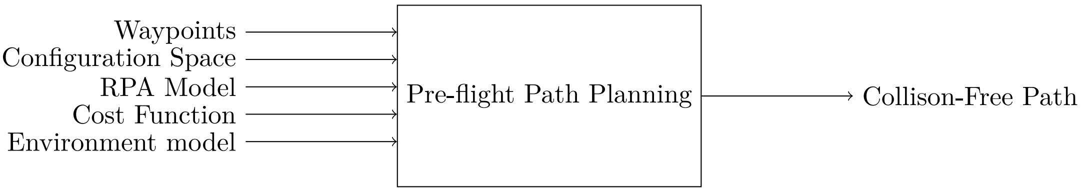


Figure 3.1: Pre-flight path planning module.

3.1.1 Configuration Space

A key concept of path planning is the representation of the physical world where the RPAS will operate. The environment model includes several natural conditions such as terrain, weather and obstacles. The configuration space is a data structure that allows the specification of the possible robot positions (location and orientation) and obstacles location. Many representations of configuration spaces can be used, the main ones being Voronoi diagrams, regular grids and quadtrees[39]. In this work, the configuration space will be defined as a regular grid. This is a conceptually simple representation, easy to construct and by properly defining the grid resolution it is possible to find kinematically feasible paths. This representation is also convenient as an action space can be independently built and a set of common actions can be applied to any of the states in the configuration space (which does not happen for instance in a direct graph structure). The physical features of the terrain can be easily obtained through Digital Elevation Model(DEM). A DEM is a regularly spaced matrix of elevation points. The spacing of latitude and longitude points is defined by the map resolution. When deciding on the grid size some limitations of the RPAS must be considered. Looking at Fig. 3.2 is possible to deduce for the grid resolution,

$$\Delta x = \Delta y = \frac{R_{curv}}{\sqrt{2}}, \tag{3.1}$$

where R_{curv} is the minimum curvature radius.

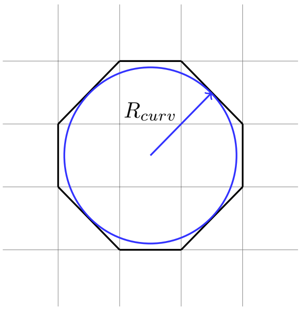


Figure 3.2: Grid Resolution

Fixed-wing platforms are not allowed to climb at an angle superior to the maximum climb angle, γ_{max}, hence the resolution along the vertical plane is defined according to this limit

$$\Delta z = \Delta x \times \arctan(\gamma_{max}). \tag{3.2}$$

According to the minimum and maximum allowed flight heights, the number of steps along the vertical plane are given by

$$H = \frac{h_{max} - h_{min}}{\Delta z}. \tag{3.3}$$

3.1.2 Constraints

Some of the kinematic constraints of the vehicle, minimum turning radius and maximum climb angle, were already included in the definition of the search space. Other constraints in the vehicle maneuverability can be included in the process of node expansion during the search process through the graph. Distinct expansion rules are defined for multirotor and fixed-wing platforms. For a non-holonomic vehicle, or multirotor platforms, any of the 26 neighboring nodes in a regular grid can be reached, as illustrated in Fig. 3.3.

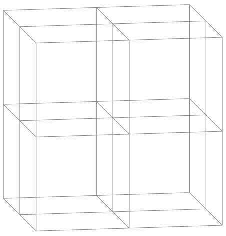


Figure 3.3: Expansion rules for multirotor platforms.

Fixed-wing platforms have a forward only motion and cannot make sharp turns or climbs. To incorporate maneuverability restriction, a set of expansion rules is defined, as illustrated in Fig. 3.4.

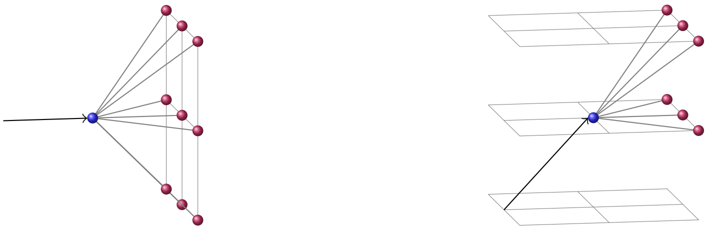


Figure 3.4: Expansion rules for fixed-wing platforms.

Other constraints are the minimum safety distance. Given the relative distance between the RPA position and the obstacle center $d_o = ||P_{rpas} - P_{obs}||$, the collision avoidance constraints is

$$d_o \geq R_s = R_{obstacle} + d_{min}, \tag{3.4}$$

where R_s is the obstacle radius plus the minimum allowed distance between vehicle and obstacles. The minimum distance, d_{min}, is defined by considering the RPA size and and extra distance for possible deviations that may occur during the execution of the path.

The mission constraints are the waypoints given to the path planner which the RPAS must visit,

$$WP = \{\mathbf{P}_1, \dots, \mathbf{P}_n\}, \quad \mathbf{P}_i = [x_i, y_i, z_i]. \tag{3.5}$$

3.1.3 Cost Functions

Depending on the mission objectives, different cost functions can be considered. RPAS have limited range and endurance so when planning paths a broadly used criteria is the minimum distance. Looking at the problem of limited on-board energy, another important objective would be to plan for least energy cost paths.

Minimum Distance

For the minimum distance paths, the cost function is simply given by the sum of Euclidean distance between all points. Considering a path $P = \{\mathbf{P}_1 \ldots \mathbf{P}_N\}$ of N waypoints the cost is given by

$$F_d = \sum_{i=1}^{N-1} \|\mathbf{P}_{i+1} - \mathbf{P}_i\| \,. \tag{3.6}$$

Minimum Energy

To formulate the energy minimization problem an energy balance is considered. Considering a point mass model for the RPA, its motion can be analyzed using the work and energy method[52]. The method of work and energy allows the simplification of problems involving forces, displacements and velocity of a particle. The integration of forces with respect to displacement result in the work energy equations. During a finite movement of the point of application of a force $\mathbf{F}$, the work done by the force along a displacement $\mathbf{s}$, from point 1 to point 2 is given by

$$U_{1\rightarrow 2} = \int_1^2 \mathbf{F} \cdot d\mathbf{s}. \tag{3.7}$$

During its displacement between two points, the RPA is subject to four forces: weight, lift, drag and thrust. The free body diagram is depicted on Fig. 3.5.

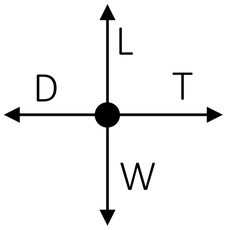


Figure 3.5: Free body diagram.

Weight The work done by the RPA weight, is dependent on height variations between points. The work done is negative as gravity acts in the downward direction. As the RPA is descending, positive work is being done by gravity, and when it is climbing the work is negative.

$$W_{1\to 2} = \int_{h1}^{h2} m\mathbf{g} \cdot d\mathbf{s} = -mg\Delta h \tag{3.8}$$

Lift Lift is perpendicular to the direction of motion. Any force perpendicular to the flight direction does no work, hence

$$L_{1\to 2} = \int_{1}^{2} \mathbf{L} \cdot d\mathbf{s} = 0. \tag{3.9}$$

Drag As the RPA moves through the air the resistance force that opposes its motion is called drag. The work done by drag is negative as its direction is always along and opposing to the flight path,

$$D_{1\to 2} = \int_{1}^{2} \mathbf{D} \cdot d\mathbf{s} = -D\Delta s = -C_D \frac{1}{2}\rho_{air} v^2 S \tag{3.10}$$

Its value is calculated according to eq. (3.10), where C_D is the drag coefficient, v the RPA velocity, S the platform wing area and ρ_{air} the air density.

Thrust Thrust is the force created by the engine that opposes drag. It corresponds to the supplied energy.

$$T_{1\to 2} = \int_{1}^{2} \mathbf{T} \cdot d\mathbf{s} = T\Delta s = E_{supplied_{1\to 2}} \tag{3.11}$$

According to the work and energy method if a force acts on an object causing its kinetic energy to vary then the mechanical work is given by

$$U_{1\to 2} = E_{k2} - E_{k1} = \frac{1}{2}m(v_2^2 - v_1^2). \tag{3.12}$$

If several forces act on the body, U represents the work done by the resultant force: $U = W+L+D+T$. Considering the forces acting on the RPA previously described, the equation for the supplied energy is

$$E_{supplied_{1\to 2}} = \frac{1}{2}m(v_2^2 - v_1^2) + D\Delta s + mg\Delta h. \tag{3.13}$$

The energy balance given by Eq. 3.13 is a statement about how energy is spent. This simplified model can be applied to either fixed-wing platforms or multirotors. In the latter case, the drag component tends to be negligible.

To determine the consumed energy of the vehicle between points, the velocity must be determined. To this point wind effects have not been considered, however in practical operations the presence of wind is generally the rule. Wind can have a strong influence on small RPA performance. The effect can be either adverse or beneficial. A headwind will reduce the RPA ground velocity and the covered distance while increasing the time required to reach the goal. A tail wind will have the opposite effect. In Eq. 3.13 v corresponds to ground velocity, V_{ground} , and Δs to the air displacement, which is dependent on V_{air} and V_{ground}, as

$$\Delta s = V_{air}\Delta t = V_{air}\frac{||AB||}{V_{ground}}, \tag{3.14}$$

where $||AB||$ is the travelled ground distance. To calculate the cost of transition between nodes, the absolute value of V_{air} and V_{ground} needs to be determined. Using the wind triangle,

$$\mathbf{V}_{ground} = \mathbf{V}_{air} + \mathbf{V}_{wind}. \tag{3.15}$$

To ensure that the RPA follows the desired ground track, the component of $\mathbf{V}_{ground}$ transversal to the ground path must be zero,

$$\mathbf{V}_{ground} \cdot \hat{T} = 0 \equiv (\mathbf{V}_{air} + \mathbf{V}_{wind}) \cdot \hat{T} = 0, \tag{3.16}$$

where $\hat{T}$ is the unit vector of the transversal component. If the wind field is known a priori, and the desired ground track is also known, $\hat{T}$ can be determined through

$$\hat{T} = \frac{\widehat{V_{ground}} \times \mathbf{V}_{wind}}{||\widehat{V_{ground}} \times \mathbf{V}_{wind}||}. \tag{3.17}$$

Once the vehicles flies at constant speed, the absolute value of V_{air} is known and its in-track component is calculated from

$$V_{air_t} = \sqrt{V_{air}^2 - V_{air_c}^2}. \tag{3.18}$$

Once the transversal component is known, $\mathbf{V}_{air}$ is given by the sum of its cross-track (c) and its in-track (t) components,

$$\mathbf{V}_{air} = V_{air_c}.\hat{n}_c + V_{air_t}.\hat{n}_t, \tag{3.19}$$

where $\hat{n}_c$ and $\hat{n}_t$ are respectively the unit vectors of cross-track and in-track components. The value of V_{ground} is then determined by

$$V_{ground} = (\mathbf{V}_{air} + \mathbf{V}_{wind}) \cdot \hat{V}. \tag{3.20}$$

Finally the cost function for minimum energy paths is given by

$$F_e = \sum_{i=1}^{N-1} E_{supplied_{i \to i+1}}. \tag{3.21}$$

This is a simplified way to calculate the energy cost, but if other models are available to compute the energy required to move between two points they can be easily included. In [53] a more accurate model is developed which considers the consumption of the propulsion system, avionics and mechanical energy. The work also covers the case of a hovering multirotor.

3.2 Path Search

A path planning problem in a discrete environment is defined by the following elements [39]:

- A state space X, which is a finite or countable finite set of states;
- A set of available actions, $u \in U(x)$ for each state $x \in X$;
- a state transition function f that produces a state $f(x, u) \in X$ for every $x \in X$ and $u \in U(x)$;
- An initial state $x_I \in X$;
- A goal state $x_G \in X$.

General graph search algorithm work by systematically searching the graph by applying the transition function and choosing the states that minimize the cost function while keeping track of the visited nodes so that no redundant exploration occurs.

3.2.1 A* Algorithm

The A* algorithm was initially described in 1968 by Peter Hart, Nils Nilsson and Bertram Raphael[15]. It is one of the most widely known and used algorithms in path finding and graph transversal. A* search is an informed method that improves on classic graph search methods by combining aspects of both uniform cost search and greedy best first search. Uniform cost search methods work by expanding the nodes with the lowest path cost first. The path cost $g(n)$ represents the cost of reaching node n from the initial state. This type of methods is called uniformed search strategy as the algorithm has no additional information about the problem, like the goal position for instance. A greedy best first method tries to find a solution as quickly as possible by expanding nodes that are closer to the goal state. This is done by evaluating nodes using the heuristic function, $f(n) = h(n)$, which gives an estimate of the cost to reach the goal. The A* algorithm works by evaluating nodes according to the cost function

$$f(n) = g(n) + h(n). \tag{3.22}$$

Since $g(n)$ denotes the cost to reach the node and $h(n)$ represents the cost of getting from the node to the goal, the cost function $f(n)$ represents the estimated cost of the cheapest solution going through n.

This method is known to be complete (it always finds a solution if one exists) and optimal (the solution found is the optimal one) if the heuristic function satisfies certain conditions. The first condition for optimality requires that $h(n)$ be a admissible heuristic. An heuristic function is said to be admissible if it never overestimates the cost of reaching the goal. Admissible heuristics are generally optimistic as they assume the cost of solving the problem to be less than it actually is. The most used example of an admissible heuristic is the straight line distance, as it is the shortest path between two points. The second condition implies that $h(n)$ must be consistent. This means that for every node n and every successor n' the estimated cost of reaching the goal from n is less than the step cost of getting form n to n' plus the estimated cost of reaching the goal from n' [14],

$$h_{n\rightarrow goal} <= g_{n\rightarrow n'} + h_{n'\rightarrow goal}. \tag{3.23}$$

During the search process, each node in the search space is in one of the following states:

- Unvisited: states which have not been explored yet. Initially all states are unvisited, except for the first one;
- Closed: states that have already been visited and each possible next state has also been visited;
- Open: states that have already been encountered but there is still adjacent states that have not been visited.

Once the search space, transition and cost functions and initial and final configurations are defined, the A* algorithm works as follows: two list are created, OPEN and CLOSED. The start node is added to the OPEN list and the search cycle begins. The node with lowest f cost is chosen from the OPEN list and checked to see if it is the goal node, if so the cycle is stopped and the search is over. If the goal is not reached yet the chosen node is expanded. Each neighbor node is then evaluated. If this node is already on the CLOSED list, the node is ignored. If the node is already on the OPEN list, it can be possible that the newly found path is better, so the cost of reaching the node is compared and if the node exists in the OPEN queue with a lowest cost the node is ignored. If the new node is not in OPEN or CLOSED, it is added to the OPEN list. This process continues until the goal is reached or there is no more nodes to evaluate and a path cannot be found. Each state also contains a pointer for the parent node, so that once the search is over the path can be reconstructed from goal to start.
An example of A*, with its pseudo-code represented in Algorithm1, can be seen in Fig. 3.6, where the cost function used is the number of traveled nodes. The obtained path that minimizes the cost function is: S-4-7-11-10-9-G. As it can be seen, this is only one of the shortest paths that can be obtained. Other possible solution would be: S-4-7-11-15-14-G. So the A* algorithm is guaranteed to find one of several possible optimal solutions.

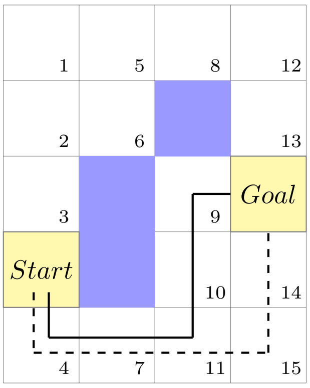


Figure 3.6: Bi-dimensional grid with start and goal states. Obstacles are represented by blue cells.

Algorithm 1: A* algorithm

```
Result: Optimal path from start to goal, $P = \{P_i, \dots, P_n\}$
initialize OPEN and CLOSED lists;
Put initial node on OPEN list;
while goal state is not reached or OPEN list is not empty do
    $best\_node \longleftarrow \min_{f(n)}(OPEN)$;
    if best_node is goal then
        $P \longleftarrow$ Reconstruct Path;
        return P;
    else
        remove best node from OPEN list;
        add best node from CLOSED list;
    end
    $Sucessors \longleftarrow$ Expand best_node;
    foreach $next \in Sucessors$ do
        $f(next) \longleftarrow g(best\_node) + g(best\_node, next) + h(next, goal)$;
        if $next \in CLOSED$ then
            if $f(next) \geq f(curent)$ then
                ignore this node
            end
        else if $next \in OPEN$ then
            if $f(next) \geq f(curent)$ then
                ignore this node
            end
        Set parent of next to best_node;
        $OPEN \longleftarrow next$;
    end
end
```

3.2.1.1 Weighted A*

In the example given previously, the algorithm could quickly converge to the right solution. However when considering a higher dimension planning graph, the number of nodes to explore increases exponentially. As such, for more complex problems, A* may take too long to find a feasible solution. In this case, an approximate solution that can be found faster may be more useful. To improve the algorithms performance, there is an approach called Weighted A*[54]. The WA* version makes use of a weighted sum of cost and heuristic to speed up the search process by reducing the number of explored nodes,

$$f(n) = w \times h(n) + g(n), \tag{3.24}$$

where w is a chose weight. For $w > 1$ there is a bias towards states that are closer to the goal. This weighted function improves the algorithms performance but optimality is no longer guaranteed as the heuristic is no longer admissible. The cost of the solution found is in the worst case w times greater than the optimal solution,

$$Cost(Solution) <= w \times Cost(OptimalSolution). \tag{3.25}$$

To evaluate the influence of w on the quality of the obtained solution, a set of tests were run over a large search space. The search space characteristics for this test are given in Tab. 3.1.

Map Size	Start	Goal	$\Delta x = \Delta y$	Δz	R_{safe}
$463 \times 350 \times 60$	[51,439,200]	[324,258,200]	30	10	100

Table 3.1: Map characteristics.

The values attributed to Δx and Δy correspond to a platform with minimum turning radius of $20m$ and the vertical spacing of $10m$ corresponds to a maximum climb angle of approximately 18°. A minimum safety distance of $100m$ from any obstacles or terrain is defined. Minimum vertical separation is simply evaluated by checking the terrain height in the DEM file as the nodes are evaluated. The horizontal separation is checked by evaluating the height of the surrounding nodes up to the defined distance. In these tests, the cost function for the search was simply the distance traveled from the start state to the current state and the heuristic used was the straight line Euclidean distance from the current node to the goal. The tests were run considering both fixed-wing and multirotor platforms. The average results for 30 simulations are presented in Tab. 3.2, for fixed-wing platforms and Tab. 3.3 for multirotor vehicles. The planned paths can be seen in Fig. 3.7.

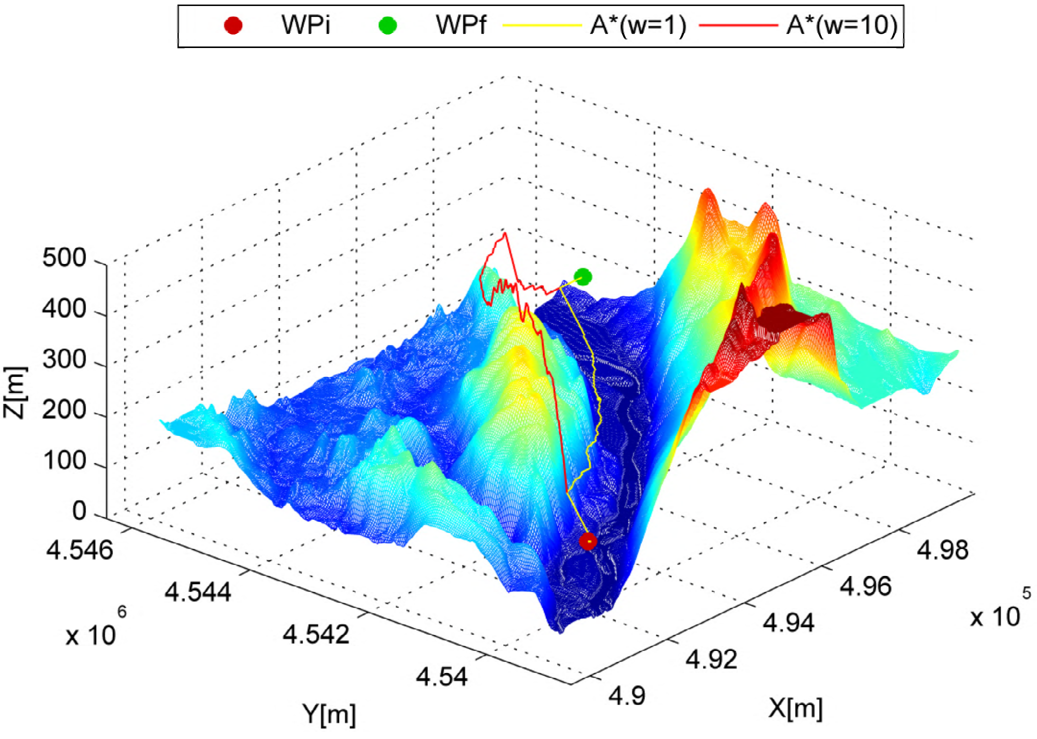
w	OPEN	CLOSED	Distance [m]	Time[s]
1	769422	49567	1.0446×10^4	9621
1.1	1872	275	1.0460×10^4	0.4036
1.2	1871	275	1.0460×10^4	0.3993
1.3	1871	275	1.0460×10^4	0.3828
1.4	1871	275	1.0460×10^4	0.3839
1.5	1871	275	1.0460×10^4	0.3836
1.6	1871	275	1.0460×10^4	0.3827
1.7	1871	275	1.0460×10^4	0.3884
2	1868	275	1.0460×10^4	0.3885
3	1868	275	1.0481×10^4	0.3563
4	1557	275	1.0501×10^4	0.3501
5	1469	275	1.0512×10^4	0.3457
10	1439	275	1.0517×10^4	0.3175
15	1435	275	1.0517×10^4	0.3172
20	1434	275	1.0517×10^4	0.3163
25	1432	275	1.0539×10^4	0.3160
30	1432	275	1.0539×10^4	0.3161
50	1429	275	1.0539×10^4	0.3160

Table 3.2: WA* results for fixed-wing platform.

From the obtained results it can be seen that WA* provides a faster execution time and suboptimal paths. The process of searching through the list is implemented with a for loop which explains the amount of time necessary to find an optimal path. With more efficient data structures better results can be achieved. Also when comparing the results obtained for fixed-wing platforms and multirotors, the effect of having more possible neighboring nodes to explore is notable by the running time and number of nodes present in the OPEN list. It is clear that even the smallest increase in value of w greatly reduces the number of explored nodes, biasing the search towards the goal. For a fixed-wing platform and $w = 20$,

w	OPEN	CLOSED	Distance [m]	Time[s]
1	942071	170338	1.0446×10^4	1916.9
1.1	3094	275	1.0460×10^4	1.1062
1.2	3079	275	1.0460×10^4	1.0598
1.3	3078	275	1.0460×10^4	1.0580
1.4	3078	275	1.0460×10^4	1.0585
1.5	3078	275	1.0460×10^4	1.0588
1.6	3078	275	1.0460×10^4	1.0530
1.7	3078	275	1.0460×10^4	1.0573
2	3088	275	1.0460×10^4	1.0621
3	2878	275	1.0484×10^4	1.0213
4	2721	275	1.0506×10^4	0.9552
5	2625	275	1.0520×10^4	0.9532
10	2581	275	1.0522×10^4	0.9048
15	2566	275	1.0523×10^4	0.9005
20	2567	275	1.0525×10^4	0.9081
25	2558	275	1.0543×10^4	0.8851
30	2554	275	1.0543×10^4	0.8860
50	2553	275	1.0543×10^4	0.8850

Table 3.3: WA* results for multirotor.



;

Figure 3.7: WA* path for w=1 and w=10.

the path cost is about 1.29% higher than the optimal path. Even though the increase in distance is not much higher, the path becomes less ”flyable” as it includes numerous height changes along the path.

During the offline computation stage we are not concerned with time issues, but for an online implementation the processing speed is key, so in those cases a compromise between path optimality and processing requirements may be necessary. A well informed heuristic is also key in the quality of the path obtained.

To this point paths have been planned only between two points. If more waypoints are to visited the planned path may not be optimal anymore. A* only allows to ”stitch” paths together in a predefined order.

3.2.2 Ant Colony Optimization

Ant Colony Optimization is a metaheuristic method derived from the observation of real ants' behavior when searching for food [23]. A metaheuristic is a higher level problem independent framework that provides some general guidelines on how to solve the problem at hand. This type of algorithms try to find the best solution out of all possible solutions. An important trait of ants' behavior is the way they communicate among each other. They use a trail of pheromones to mark paths on the ground from the nest to the food source. By sensing this trail other ants can follow the path. Higher concentration of pheromones in a given path will lead to a higher probability of other ants to follow that path. If an ant finds a path that does not lead to the desired source or if the path found is longer, pheromones will evaporate with time and the path is “forgotten”. The laying and following of pheromones is the main inspiration for ACO, where a set of artificial ants is used to find solutions for a given optimization problem. Using probabilistic rules to model the ants' behavior, a set of artificial ants is designed to move on a graph and find the optimal path between two nodes. The stochastic solution process is biased by the pheromones deposited along the path. The ACO metaheuristic, described in Algorithm2, consists of four main steps: initialization, where all the algorithms parameters are defined, solution construction, pheromone deposit and pheromone evaporation. This process is repeated until a termination criteria is met. Generally this criteria is a maximum number of iterations or CPU time.

Algorithm 2: ACO metaheuristic

```
Set initial parameters;
while not termination do
    Solution Construction;
    Pheromone Deposit;
    Pheromone Evaporation;
```

Taking as an example the well-known traveling salesman problem, the objective is to guide a salesman from its home to a set of costumer cities, visiting each city once and returning home while covering the shortest distance. The problem is represented as a weighted graph where each node represents the city and the weights connecting the nodes are equivalent to the traveled distance. The constraint to the problem is that every city must be visited one time. This is enforced by making each ant keep track of the cities they have visited in a *tabu* list. The ants cannot visit again any city in this list. The pheromone trail is the desirability to follow a good path. To obtain the shortest distance path, the distance between cities is used as heuristic information. A solution to the problem is constructed by putting all ants on the initial city and keep them adding cities to their *tabu* list until all cities have been visited. Pheromones are then updated, given more emphasis to the solution which provides a better cost. To this day, several variation of the ACO algorithm have been proposed, the main ones being: the ant system, ant colony system and MIN-MAX ant system [23].

3.2.2.1 Ant System

The ant system was the first ACO algorithm to be proposed by Dorigo in 1992 [23]. The two main steps of the ant system algorithm are the ants' solution construction and pheromone update. While ants construct their solution, the transition probability of the k ant move from node i to node j is given by a random proportional rule

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}, \tag{3.26}$$

where η_{ij} is the heuristic value and α and β determine the influence of the pheromone trail and the heuristic information. After all ants have constructed their tour, the pheromone trail is updated. First the pheromone evaporation is implemented according to

$$\tau_{ij} \leftarrow (1-\rho).\tau_{ij}, \tag{3.27}$$

where ρ is the pheromone evaporation rate. This evaporation stage avoids the unlimited accumulation of pheromones on the arc edges and allows the algorithm to forget previous bad choices. Then a pheromone value is added on the nodes that the m ants have visited during their tours,

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{3.28}$$

where $\Delta\tau_{ij}^{k}$ is the amount of pheromone deposited on each arc. This value is related to the cost of the k ant tour and is defined by

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C_k}, & \text{if ant k travels on edge ij} \\ 0, & \text{otherwise,} \end{cases} \tag{3.29}$$

where C_k is the cost of the tour built by ant k. This allows ants with lower cost tours to deposit a larger amount of pheromones on its path.

3.2.2.2 Ant Colony System

The Ant Colony System (ACS) [23] was the first major improvement to be proposed over the original ant system. There are three main differences between both algorithms. The first change is the decision rule that ants use during the solution construction. The ACS uses the called pseudo-random proportional rule, where ants choose the node to visit according to

$$j = \begin{cases} argmax\{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}\}, & \text{if } q \leq q_0 \\ S, & \text{otherwise,} \end{cases} \tag{3.30}$$

where q is a random number uniformly distributed in $[0 \ldots 1]$, S is a random variable selected according to the probability distribution given by eq.(3.26) and $q_0 \in [0, 1]$ is a parameter that determines the importance of exploration versus exploitation. A high value of q_0 will concentrate the search around the best solution found so far will and a lower value will encourage the search of new paths. The second improvement is the addition of a local pheromone update rule. This means that pheromones are updated as ants move between nodes according to

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0, \tag{3.31}$$

where ξ is the local evaporation coefficient and τ_0 is the initial pheromone value. The local updating of pheromones is used to avoid a stagnation behavior of the algorithm. As an ant makes its tour the pheromone value on the arcs is reduced encouraging the following ants to choose new paths. The third difference is in the global update of pheromones once all ants have completed their solutions. In ACS, only the ant with the best tour so far contributes to the laying of pheromones.

3.2.2.3 Min-Max Ant System

The Min-Max Ant System (MMX) [23] differs from the original ant system in the following aspects:

- Like in ACS only the best ant adds pheromones to the trail;
- The range of pheromones values in the trail is limited to an interval $[\tau_{min}, \tau_{max}]$;
- The pheromone trail is initialized to the upper pheromone trail limit, τ_{max};
- The pheromone trail is reinitialized each time the system reaches a stagnation point, when no improved tour is produced for a certain number of iterations.

3.2.2.4 ACO for the path planning problem

In this work, both variants of the ACO algorithm, ACS and MMX, are applied to the RPAS path planning problem.

Graph Construction: the graph is constructed as a weighted three-dimensional grid. Each node represents the RPAS position and orientation in a Cartesian coordinate system (x, y, z, γ, ψ), where γ is the flight path angle and ψ the heading, as depicted in Fig. 3.8. The spacing between the nodes is proportional to the distance between them.

Constraints: they correspond to the possible moves the ants can make. Depending on the type of platform being considered different movements are allowed as defined in Sec. 3.1.2.

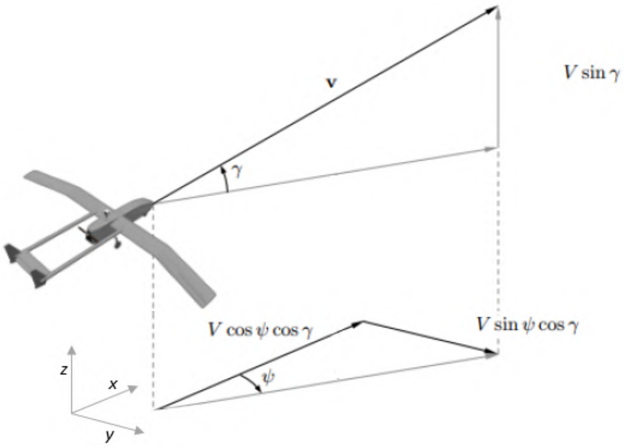


Figure 3.8: Kinematic model representation(adapted from[55]).

Pheromone trail:: the pheromone trail represents the desirability of visiting one node after the other. Generally, pheromones are deposited in the edges connecting the graph nodes, however when planning in a large three-dimensional grid it is infeasible to define each possible edge connecting nodes so in this implementation pheromones will be deposited in nodes instead of the edges.

Heuristic information: to ensure that ants reach the target point, the heuristic value is defined as the inverse of the distance between each node and the target. One of the purposes of the proposed path planning system is to find a path that can minimize energy expenditure. To do so the heuristic information must include information about energy expenditure between nodes. The heuristic information is computed according to

$$\eta_{ij} = \left(\frac{1}{d_{ij}}\right)^{c} \left(\frac{1}{e_{ij}}\right)^{1-c}, \tag{3.32}$$

where d_{ij} represents the Euclidean distance between node i and node j and e_{ij} represents the energy spend in the transition between nodes. The energy is calculated using Eq. 3.13 derived in Sec. 3.1.3. The value of constant c is set to 1 or 0 depending if the objective is to optimize distance or energy respectively. In Fig. 3.15 the flow chart for the ACO algorithm is presented. The process begins by setting the initial parameters. All ants are then placed in the initial waypoint and each ant chooses the next node according to Eq.(3.30) until the goal is reached. Each ant maintains a *tabu* list of all visited nodes so that no node is visited twice. Once all ants have completed their tours, the cost of each one is evaluated. Depending on the mission objective, this cost is a either measure of expended energy to complete the path or a measure of covered distance. The best path found so far is stored and pheromones are updated using either the ACS or the MMX rules. If the iterative condition is satisfied (maximum number of iterations), the iteration ends and the best solution is given.

3.2.2.5 ACO parameters

The ACO algorithm involves a number of parameters that need to be set appropriately in order to obtain optimal solutions.

The initial pheromone value τ_0 has a significant influence in the algorithms convergence speed. If the initial pheromone values are too low, the search is quickly driven to the first tours found by the ants. However, if the value is too high, the algorithm takes too much time to converge to the right solution as many iterations are lost until the evaporation of pheromones is enough for the added pheromones by the ants start driving the search. The value for the initial pheromones depends on the particular implementation of the algorithm. For minimum length paths, a suggested estimation [23] is $\tau_0 = m/C$, where m is the number of ants and C the length of a tour generated by the nearest neighbor heuristic.

The evaporation rate ρ, regulates the evaporation rate of the pheromone trail. A value too low results in the persistence of the pheromone information and can overshadow the pheromones deposited by ants. If the value is too high, the algorithm has a tendency to forget previous paths and concentrate on new information added to the pheromone matrix. A value given for ρ is generally 0.9.

The values of α and β are used to influence the relative importance of pheromone and heuristic information. The role of these parameters is biasing the search of the ants. Values for this parameters are generally given between 1-5.

Another parameter is the number of ants used. If a larger number of ants is chosen the algorithm will need less iterations to converge to the optimal solution. Good solutions are generally found for m=10.

Finding the appropriate parameters is not an easy task. Good parameter values for specific problems generally come from experimental research. As such, a study of parameter variation was conducted. The following parameters were evaluated: α, β, N of ants, q_0, ρ and τ. In each experiment, one parameter was varied while the others were set to the default values in Tab. 3.5. The results are presented for the case scenario in Tab. 3.4, the cost function used was the minimum distance and the curves are the result of the average of 20 runs for each parameter.

Map Size	$\Delta x = \Delta y$	$N^{o}Obstacles$	R_{safe}
50×50	10	10	20

Table 3.4: Map characteristics.

α	β	q_0	ρ	τ	τ_{min}	τ_{max}
1	1	0.9	0.9	1	0.1	10

Table 3.5: ACO default values.

For the parameter α, Fig. 3.9, there is not much difference in the convergence time between values but the worst results are obtained for the smallest and the largest values (0.1 and 10). They have a greater influence in the MMX algorithm preventing it from reaching the optimal cost.

The variation of β, Fig. 3.10, for values between 0.1 and 2, has no noticeable effects in the convergence time. In both the ACS and MMX algorithms values above 5 lead to a poorer performance.

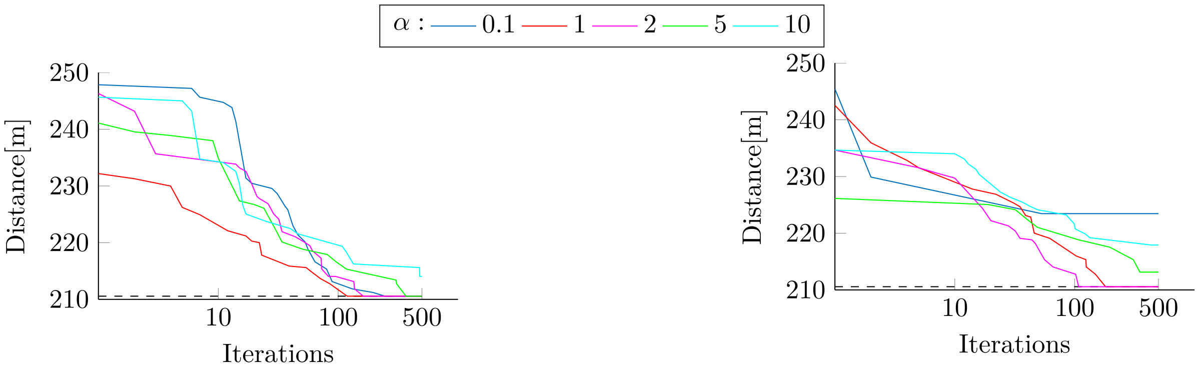


Figure 3.9: Variation of α for ACS(left) and MMX(right). The doted line represents the optimal cost.

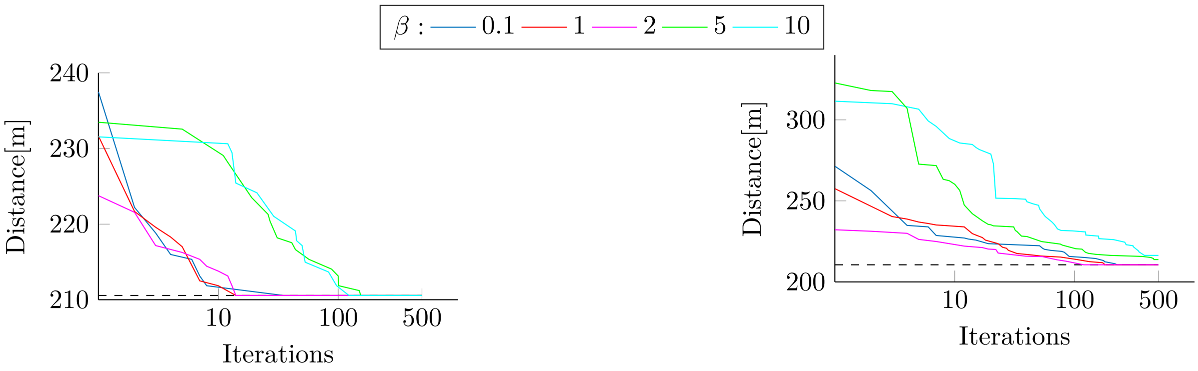


Figure 3.10: Variation of β for ACS(left) and MMX(right). The doted line represents the optimal cost.

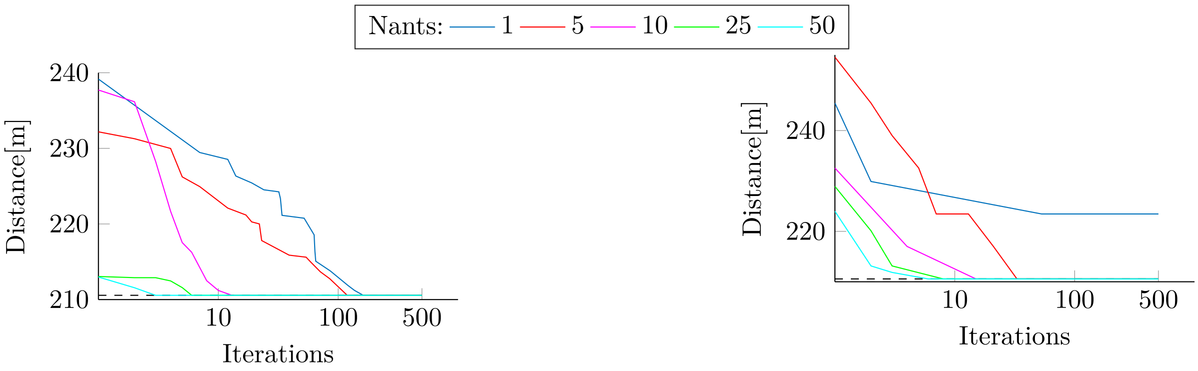


Figure 3.11: Variation of number of ants for ACS(left) and MMX(right). The doted line represents the optimal cost.

Varying number of ants, Fig. 3.11, greatly influences both algorithms performance. For smaller values the convergence time is much slower and in some cases is not even guaranteed. The larger the number of ants the fastest the optimal solution is reached.

The parameter q_0, Fig. 3.12, is important in the convergence time. A bigger value leads to a more frequent choice of the best path so the optimal solution is found faster. Small values have little guarantee of finding the optimal solution in time.

For the parameter ρ, Fig. 3.13, small values give worst solutions. The solution quality shows little

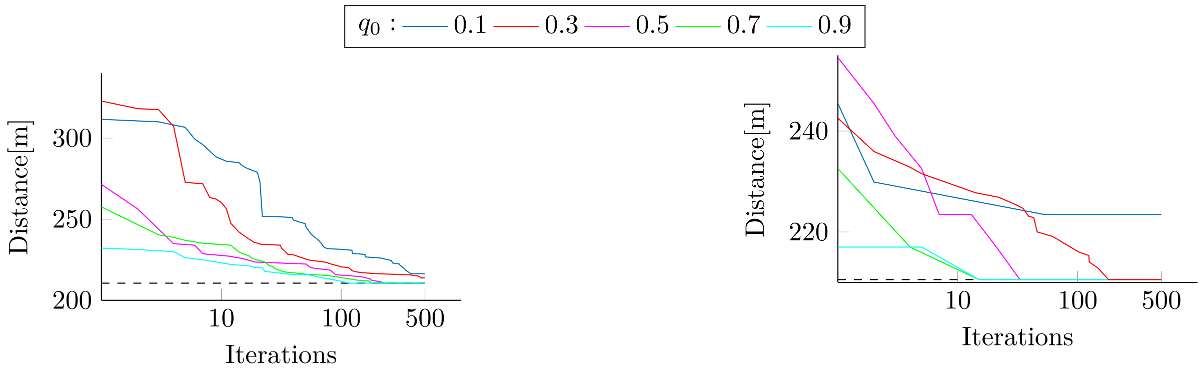


Figure 3.12: Variation of q_0 for ACS(left) and MMX(right). The doted line represents the optimal cost.

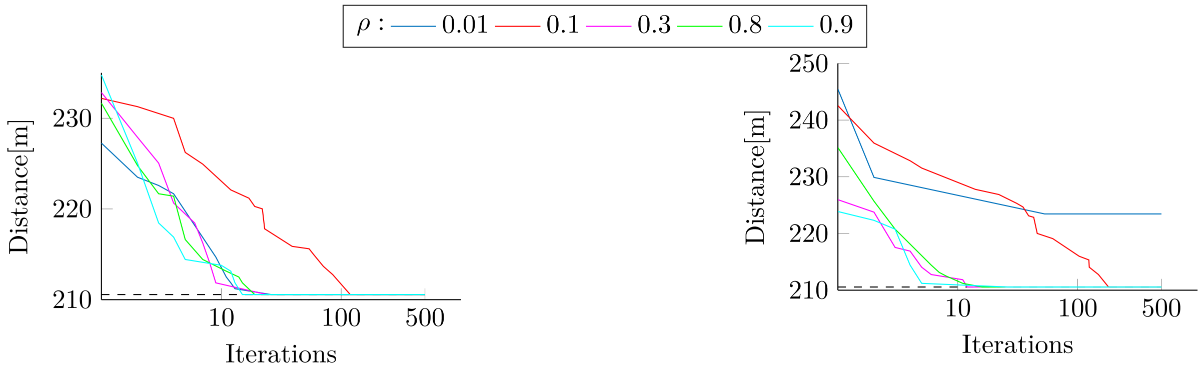


Figure 3.13: Variation of ρ for ACS(left) and MMX(right). The doted line represents the optimal cost.

difference for values above 0.1.

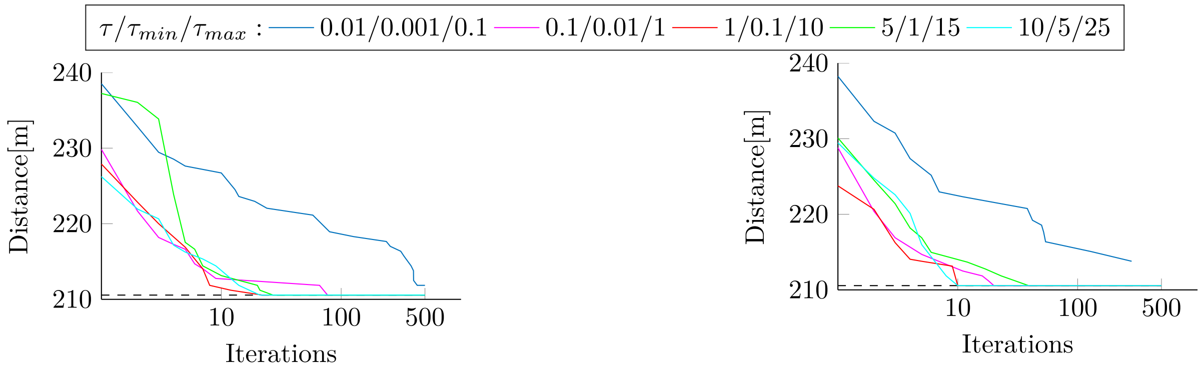


Figure 3.14: Variation of $\tau/\tau_{min}/\tau_{max}$ for ACS(left) and MMX(right). The doted line represents the optimal cost.

Small values of τ, Fig. 3.14, prevent the algorithms of converging to the optimal solution. Largest values show little difference between them.

The values selected for the ACO algorithms, presented in Tab. 3.6, were chosen considering the convergence to the optimal solution and the convergence time. The parameter α was maintained in its default value, as it gave the best compromise for both the ACS and MMX variants. The value of β was set to 2, as it provided the fastest convergence for both algorithms. The number of ants was set to

10 because it gave the best compromise between the number of iterations needed to reach the optimal solution, and the computational load placed in each iteration. The parameter of q_0 was chosen closer to one to ensure convergence and prioritize the following of optimal solutions instead of exploring new ones. The chosen values of ρ, τ, τ_{min} and τ_{max} were the ones that provided the fastest convergence for both ACS and MMX.

α	β	q_0	ρ	τ	τ_{min}	τ_{max}	Nants
1	2	0.9	0.9	1	0.1	10	10

Table 3.6: ACO selected parameters.

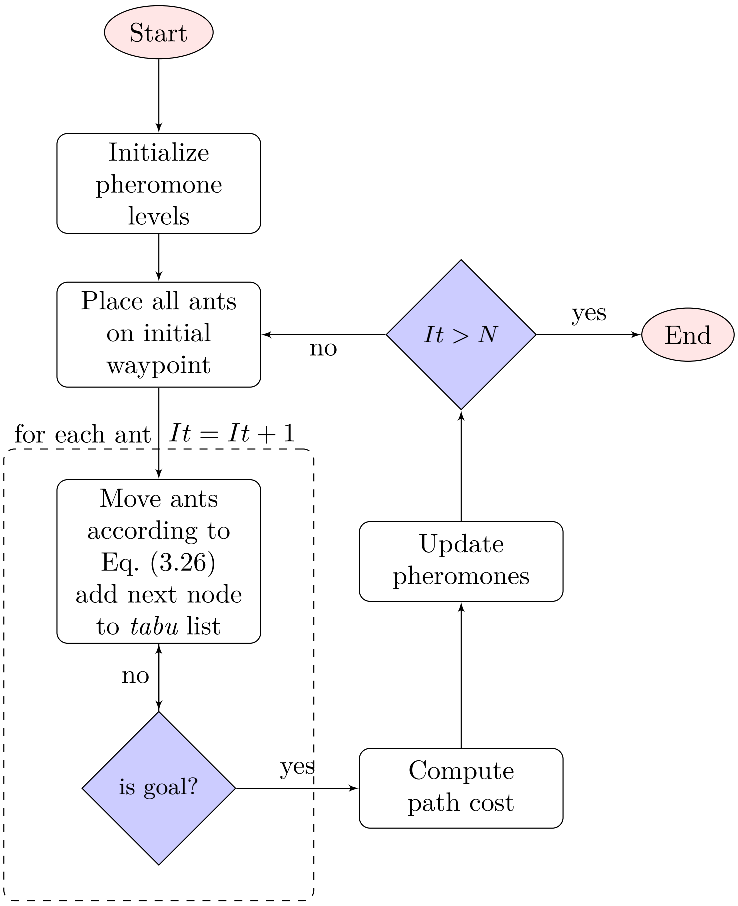

Start
Initialize pheromone levels
Place all ants on initial waypoint
no
$It > N$
yes
End
for each ant
$It = It + 1$
Move ants according to Eq. (3.26) add next node to *tabu* list
Update pheromones
no
is goal?
yes
Compute path cost

Figure 3.15: ACO flow chart.

3.3 Path Smoothing

The paths obtained with A* and ACO consist of straight line segments between waypoints. These paths cannot be exactly followed by RPAS with dynamic and kinematic constraints. Discontinuities in the curvature profile translate in instantaneous change in the bank and heading angle for a flying platform. These instantaneous changes require infinite control force. A continuous curvature path must be obtained for practical applications. If the velocity goes bellow the stall speed a fixed wing platform will stall. Also it cannot stop or suddenly change its heading. Rotary platforms can hover but cannot make sharp turns or stop immediately at command.

To provide a feasible path and minimize the errors in tracking performance, the geometry of the path must be modified. Spline based methods are widely used to compute feasible trajectories for UAVs[56][57][58][59][60]. Bezier curves are a type of parametric curves designed to provide a smooth path that passes exactly through the initial and final waypoints and is influenced by the other waypoints on the way. Bezier curves are defined by

$$P(t) = \sum_{i=0}^{n} B_i^n(t) P_i, \quad t \in [0,1], \tag{3.33}$$

where P_i are the control points and $B_i^n(t)$ is the Bernstein polynomial given by

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad i \in \{0,1,\dots,n\}. \tag{3.34}$$

This type of curves have the following properties[61]:

Property 1 (End point interpolation)**.** *The curve always goes through the first and last control points,* P_0 *and* P_n *and it generally does not pass on the other points.*

Property 2 (Convex hull property)**.** *The curve lies integrally inside the convex hull of the polygon formed by the control points.*

Property 3. *A Bezier curve with n+1 control points has order n.*

Higher degree Bezier curves give more design freedom and provide a higher degree of continuity. Rational Bezier curves are a special case of NURBS curves. These curves are generate by attributing an weight w_B to each control point, pulling or pushing the curve away from the point,

$$P_R(t) = \frac{\sum_{i=0}^{n} B_i^n(t).w_{B_i}.P_i}{\sum_{i=0}^{n} B_i^n(t).w_{B_i}}, \quad t \in [0,1], \tag{3.35}$$

they allow a better control over the curve shape. In Fig. 3.16 the effect of varying the weight value of the second control point on the shape of the curve can be observed.

The curvature of a parametric curve $P(t)$ can be calculated through

$$\kappa(t) = \frac{|P'(t) \times P''(t)|}{|P'(t)|^3}. \tag{3.36}$$

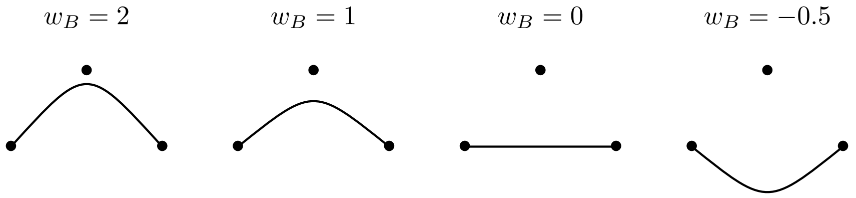


Figure 3.16: Rational Bezier curves.

The curvature depends on the first and second derivatives of the curve $P(t)$. A Bezier curve is given by a polynomial function which has continuous derivatives, hence Bezier curves always have continuous curvature[62]. The first derivative $P'(t)$ is obtained as[63]

$$P'(t) = \frac{a'(t) - b'(t)P(t)}{b(t)}, \tag{3.37}$$

where

$$a'(t) = n \sum_{i=0}^{n-1} B_i^{n-1}(t)(w_{i+1}P_{i+1} - w_i P_i), \tag{3.38}$$

$$b'(t) = n \sum_{i=0}^{n-1} B_i^{n-1}(t)(w_{i+1} - w_i), \tag{3.39}$$

and

$$b(t) = \sum_{i=0}^{n} B_i^n(t) w_i. \tag{3.40}$$

The second derivative is obtained in the same manner,

$$P''(t) = \frac{a''(t) - 2b'(t)P'(t) - b''(t)P(t)}{b(t)}, \tag{3.41}$$

where

$$a''(t) = n(n-1) \sum_{i=0}^{n-2} B_i^{n-2}(t)(w_{i+2}P_{i+2} - 2w_{i+1}P_{i+1} + w_i P_i), \tag{3.42}$$

and

$$b''(t) = n(n-1) \sum_{i=0}^{n-2} B_i^{n-2}(t)(w_{i+2} - 2w_{i+1} + w_i). \tag{3.43}$$

The properties of Bezier curves make them an attractive solution to generate flyable paths that go through the required waypoints. Using the waypoints generated by the global planner as control points a Bezier curve of continuous curvature is obtained. This curve, however, does not pass through all the points and it can lead to the violation of curvature and safety constraints. To overcome this problem, rational Bezier curves are used and the corresponding weights are optimized to originate a path within the required bounds. The problem optimizing the curve weights is formulated as

$$
\begin{aligned}
\text{minimize} \quad & F_c && (3.44)\\
\text{subject to} \quad & d_o \geq R_s && (3.45)\\
& P_R(t) = f(w) && (3.46)\\
& |k| \leq k_{max} && (3.47)\\
& w_{B_{min}} \leq w_{B_i} \leq w_{B_{max}} && (3.48)
\end{aligned}
$$

The cost function F_c to be minimized is given either by Eq. (3.6) or Eq. (3.13). If only the constraints are to be satisfied, F_c is set to zero. The constraint defined by Eq.(3.45) imposes a minimum distance between the robot and the obstacle. Eq.(3.47) ensures that, given the RPAS curvature limits, the path is flyable. The safety distance and curvature are calculated for each point on the Bezier curve, and the number of points on the curve differ from the number of optimization variables, so a constraints lumping method is performed to attribute to each optimization variable a constraint value for curvature and safety distance. The used method is the maximum constraint approach [64], where to each variable is attributed the maximum constraint value of the closest point.

A generic constrained optimization solver, *fmincon*, provided by MATLAB is used to solve the problem.

The reference waypoints given to the path planner are of obligatory passage for the platform. If more than two waypoints have to be visited, and due to Property 1, separate Bezier curves have to be created for each segment connecting the reference waypoints. When connecting the curves care must be taken to ensure that the joining point has no discontinuities. Two distinct Bezier curves are said to be C^k continuous at t_0 if

$$P(t_0) = Q(t_0), P'(t_0) = Q'(t_0), \ldots, P^{(k)}(t_0) = Q^{(k)}(t_0). \tag{3.49}$$

If two segments of control points are to be joined in a smooth way, they must be tangent in the last and first sections. High degree curves are generally not efficient to process and situations where a solution cannot be found can easily arise when planning a long range mission on an area densely populated with obstacles.

In Fig. 3.17 it can be seen an example of the minimum distance path planned by the A* algorithm for a fixed wing platform and the respective initial Bezier curve with unitary weights. In this situation, the solver is unable to find a feasible solution. To solve this problem, the curve is successively divided and each segment is optimized until the constraints are satisfied. The points of division, anchor points, must be placed in a position that guarantees curve continuity, this means that the preceding and following points must be in line with the anchor point. The results are presented in Tab. 3.7. It is noted that a significant amount of computational time is needed to optimize the several curve segments.

The flowchart for the overall pre-flight path planning Algorithm3 is shown in Fig. 3.18, where the input data corresponds to the reference waypoints, the configuration space, the cost function and the vehicle constraints.

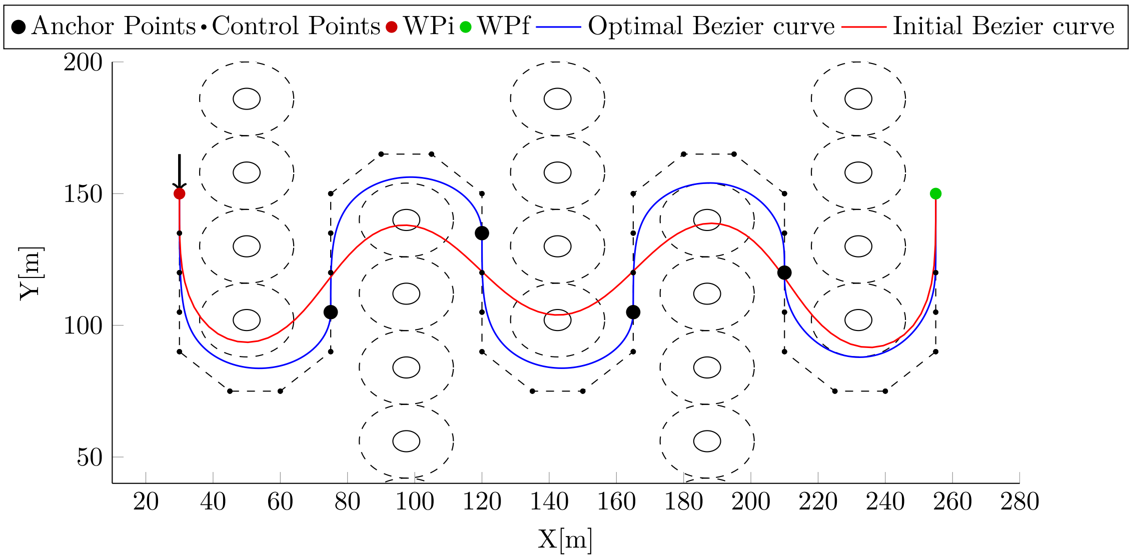


Figure 3.17: Path generated by optimized Bezier curve division.

	Distance[m]	Time[s]
$A*$	647.132	0.138
Initial Bezier curve	387.698	0.051
Optimal Bezier curve	529.891	535.034

Table 3.7: Results for minimum distance paths.

Algorithm 3: Pre-flight Path Planning.

```
Input: Constraints (sec. 3.1.2), cost function( 3.6,  3.13), waypoints( 3.5), obstacles( 2.1) and departure heading and flight path angle;
Output: Optimal path from star to goal, $P_{op}$;
Find control points $CPs = \{P_1, \dots, P_N\}$, using A* 1 or ACO 2;
Initialize weights and calculate initial Bezier curve, $P_Ri$, using eq. (3.35);
Current curve $\longleftarrow P_Ri$;
while Solution is not found do
    Find optimal weights, $w_o$, for the current Bezier curve;
    Current curve $\longleftarrow P_R(w_o)$;
    if constraints are satisfied then
        $P_{op} \longleftarrow$ Current curve;
        return $P_{op}$
    else
        Divide current curve
```

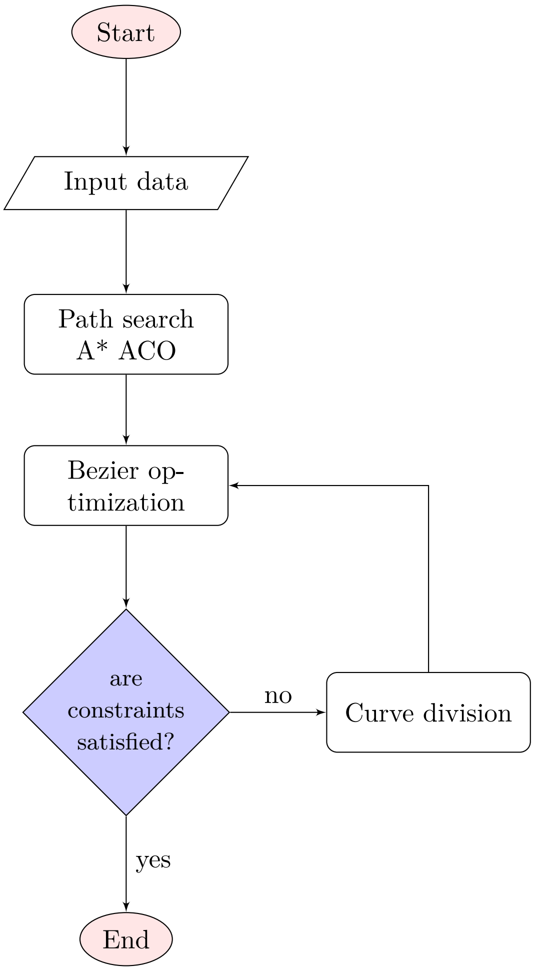

Start
Input data
Path search
A* ACO
Bezier optimization
are constraints satisfied?
no
Curve division
yes
End

Figure 3.18: Flowchart for the pre-flight path planning algorithm.

Chapter 4

Real-Time Path Planning

In this chapter the problem of replanning a reference path when new obstacles are detected is addressed and solved using the potential fields method. The new path is generated with considering the Rules of the Air during encounters with moving obstacles.

4.1 Safety Zones

The developed method to replan the original path, when a collision risk is detected, is based on previous works [65][66][67][68], with the deconfliction maneuver being dictated by the Rules of the Air. To develop the module for real-time path planning, some safety distances are defined. A collision is said to occur when the obstacle breaches the collision radius R_c, the collision radius is defined as equivalent to the vehicle dimensions. The safety radius R_s, defines the required safety distance that must be maintained, that distance should account for possible unpredicted deviations that occur during flight and be in the same order of the RPA size. Another zone to be defined is the action radius R_a, that is the distance from which the replanned path begins to depart from the original path given by the global planner. This distance should be proportional to the sum of the safety and collision radii, as the RPA will need to take action sooner to avoid bigger obstacles than smaller ones. The detection radius R_d, is dependent on the types of sensors available on the platform, this is the distance at which the object is considered by the path replanning system and should be equivalent to the detection range of the sensors.

4.2 Rules of Air

The Rules of the Air are a set of rules and procedures concerning the flight and maneuvering of aircraft in civil airspace. Regarding the avoidance of collisions for manned flight four main points are stated[69]:

1. On a head-on encounter, both aircraft should deviate to the right(Fig. 4.2(a));
2. On a converging scenario, the aircraft with the other on its right-hand side (starboard side) has to give way(Fig. 4.2(b));

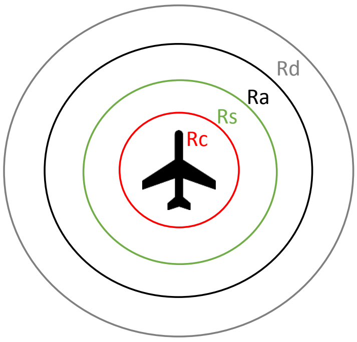


Figure 4.1: Safety zones defined around the obstacle.

3. In an overtaking event, the faster aircraft must overcome on the right hand side of the slower one(Fig. 4.2(c));
4. An aircraft should avoid passing over, under or in front of other.

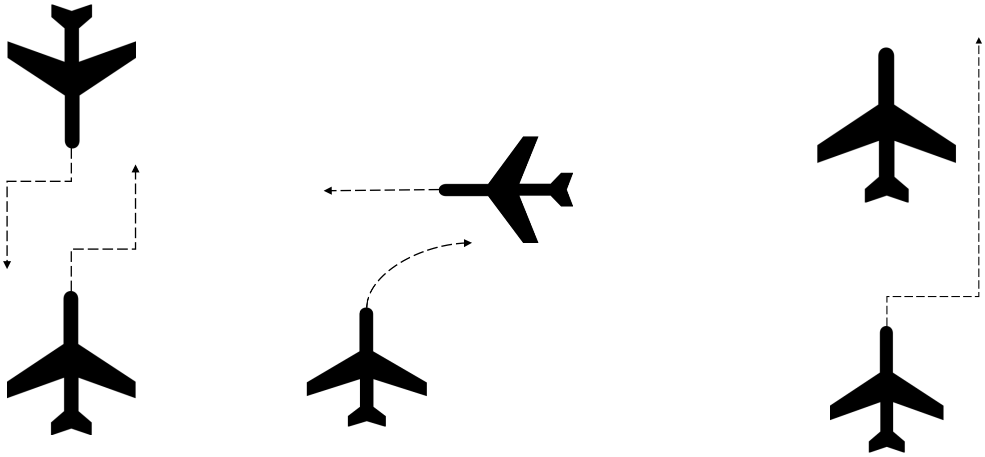


Figure 4.2: Rules of the Air: (a)Head-on encounter, (b) Converging encounter, (c) Overtaking.

To share the same airspace as manned aircraft, RPAS must consider this set rules. However they can only be fully applied when cooperative avoidance is considered. If the RPA has the right of way, it should maintain its course expecting that the other vehicle performs the avoidance maneuver. It is expected that RPA, especially the small platforms, must always give way to larger aircrafts. In this work uncooperative collision avoidance is addressed, so the RPA must always take action to avoid the collision with the safety sphere, while considering the Rules of the Air. It is assumed that the intruder aircraft will take no action and maintain its course.

4.2.1 Avoidance Logic

To comply with the Rules of the Air, when a moving obstacle is detected the type of encounter must be evaluated. Depending on the type of encounter different resolutions are adopted:

- When the intruder is found to be in a head-on collision course, or to the right of the RPA, the avoidance should be made by turning right;
- If the intruder is approaching from the left, and the RPA is on leveled flight, turning right will put the RPA in front of the intruder, to avoid this scenario the avoidance maneuver is made by turning left and going behind it;
- If the RPA is climbing, the avoidance is made by leveling the flight until the intruder is overcome;
- If the RPA is descending, the aircraft could be leveled off, but due to inertia this would be riskier than increasing the descent rate (unless the value is at its maximum).

If a static obstacle appears in the way, diferent paths are achieved depending on the direction of the avoidance. In this situation there are no rules commanding the vehicle to behave a certain way, so the following strategy is adopted:

- If any side of the obstcale is blocked, the rotation is set to the oposite direction;
- Considering the line joining the RPA position and the obstacle center, if the goal point in the path is to the left the rotation is made conterclockwise and vise versa, as depicted in Fig. 4.3. If the point or path direction is along the line the swirl direction can be arbitrarly chosen.

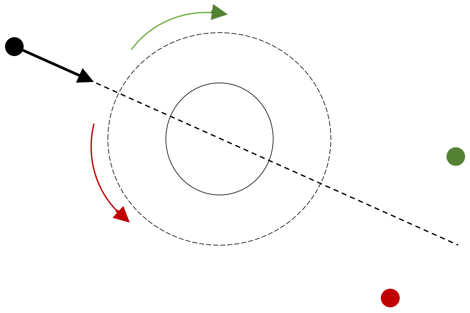


Figure 4.3: Resolution for avoidance of static obstacles.

4.3 Collision Detection

To detect possible collisions with moving obstacles, the concept of the Closest Point of Approach (CPA) is used[70].

This is the point at which the RPA is closest to the intruder.

If the closest distance is smaller than the safety distance then the RPA is in risk of collision and its path must be updated. To determine the CPA, it is assumed that the moving intruder will maintain its course of motion.

Considering two vehicles A and B, as depicted in Fig. 4.4, moving in a tridimensional space, according to $\mathbf{A}(t) = \mathbf{A}_0 + \mathbf{v}_A t$ and $\mathbf{B}(t) = \mathbf{B}_0 + \mathbf{v}_B t$, the distance between them at time t is $d(t) = \|\mathbf{A}(t) - \mathbf{B}(t)\|$. By calculating the minimum of d(t), the time to closest point of approach is given by

$$t_{CPA} = \frac{-\mathbf{d}_0 \cdot (\mathbf{v}_A - \mathbf{v}_B)}{|\mathbf{v}_A - \mathbf{v}_B|^2}, \tag{4.1}$$

where $\mathbf{d}_0 = \mathbf{A}_0 - \mathbf{B}_0$. Once the time to CPA is known the closest distance can be calculated through

$$d_{CPA} = \|\mathbf{A}(t_{CPA}) - \mathbf{B}(t_{CPA})\|. \tag{4.2}$$

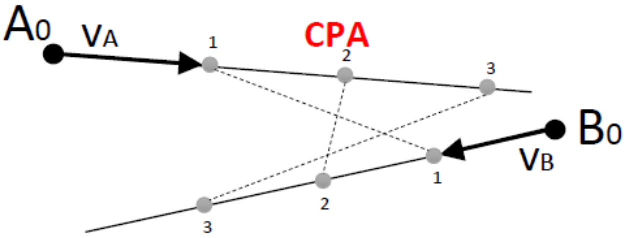


Figure 4.4: CPA representation.

When a conflict with multiple intruders occurs threats must be prioritized. As intruders will have different speeds and bearings using the distances to the collision point is not enough, so the time to collision is used instead. Higher priority will be given to intruders with the smallest t_{CPA} and the conflicts are resolved in a sequential manner.

4.4 Potential Fields

Potential Fields are one of the most used approaches to the local path planning problem[40]. In this approach, the obstacles and the goal position are treated as charged particles. A repulsive force is attributed to the obstacles and an attractive force to the goal point. The sum of those forces is used to generate the direction of motion. The attractive and repulsive potential fields are developed in a similar way to [65][66].

Attractive Potential

The attractive potential is responsible for the directing of the RPAS towards the desired destination. If the objective is to direct the vehicle to a single goal waypoint, the potential function $\mathbf{F}_{at}$ is simply given by the direction from the current position, $\mathbf{P}_{rpas}$, to the desired waypoint, $\mathbf{WP}$. Its representation can be seen in Fig. 4.5.

$$\mathbf{F}_{at} = \frac{\mathbf{WP} - \mathbf{P}_{rpas}}{\|\mathbf{WP} - \mathbf{P}_{rpas}\|} \tag{4.3}$$

When the mission consists on following a pre-planned path, the potential function must take into account two terms: one that brings the RPA close to the given path and other that makes the vehicle follow the path direction. To obtain the first term the closest point on the path, $\mathbf{P}_{close}$, to the current

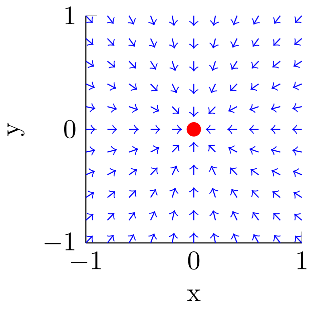


Figure 4.5: Attractive potential field for a waypoint.

position, $\mathbf{P}_{rpas}$, is found and the direction between both is taken. The path following term is obtained from the direction from the closest point on path to the next point on the path, $\mathbf{P}_{next}$.

$$\mathbf{F}_{at} = \alpha_{PF}\frac{\mathbf{P}_{close} - \mathbf{P}}{\|\mathbf{P}_{close} - \mathbf{P}\|} + (1-\alpha_{PF})\frac{\mathbf{P}_{next} - \mathbf{P}_{close}}{\|\mathbf{P}_{next} - \mathbf{P}_{close}\|} \tag{4.4}$$

By selecting the values of α_{PF} more importance can be given to the path following or the path approaching direction. In Fig. 4.6 the potential function for a path can be seen.

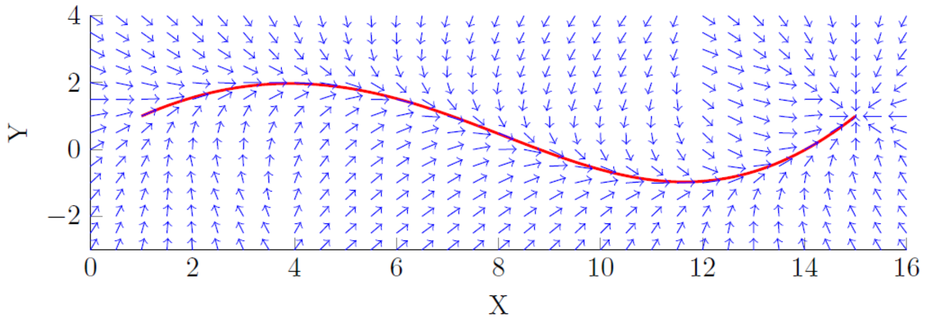


Figure 4.6: Attractive potential field for a path ($\alpha = 0.5$).

Repulsive Potential

To keep the RPAS safe a repulsive potential is placed upon the obstacles. The simplest form of a repulsive potential induced by an obstacle, $\mathbf{P}_{obs}$, is given by:

$$\mathbf{F}_{rep} = \begin{cases} 0, & \text{if } d_o \leq R_a \\ \frac{\mathbf{P}_{rpas} - \mathbf{P}_{obs}}{\|\mathbf{P}_{rpas} - \mathbf{P}_{obs}\|}, & \text{if } R_c \geq d_o \leq R_a \\ \infty, & \text{if } d_o \leq R_c \end{cases} \tag{4.5}$$

If the distance to the obstacle is greater than the action radius, the obstacle has no influence and the potential is zero. Between the action and collision radius, the potential is dependent on the distance to the obstacle. For distances inferior to the collision radius, the potential is infinite and points in the opposite direction of the vector connecting the current point to the obstacle center. The representation of this potential is seen in Fig. 4.7, where an obstacle is depicted with its safety and action radius.

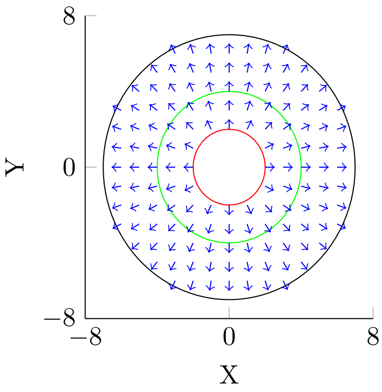


Figure 4.7: Simple repulsive potential flow.

This type of potential keeps the RPA at a distance, but leads to sudden changes in direction. To avoid this type of behavior, where the RPA abruptly changes its heading, the field strength is made proportional to the distance from the obstacle center, $\mathbf{d}_o = \mathbf{P}_{rpas} - \mathbf{P}_{obs}$. The field is given by

$$\mathbf{F}_{rep} = \begin{cases} 0, & \text{if } d_o \leq R_a \\ \frac{\mathbf{d}_o}{\|\mathbf{d}_o\|} \left| \frac{R_t - \|\mathbf{d_o}\|}{R_t} \right|, & \text{if } R_c \geq d_o \leq R_a \\ \infty, & \text{if } d_o \leq R_c \end{cases} \tag{4.6}$$

where $R_t = R_c + R_s + R_a$, and is represented in Fig. 4.8.

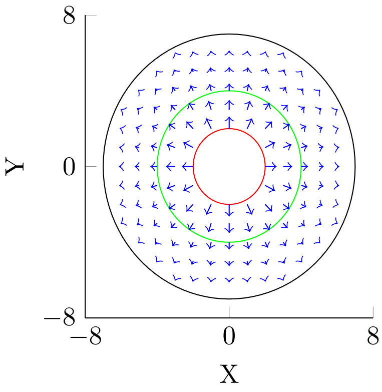


Figure 4.8: Fading repulsive potential flow.

This potential keeps the distance from the obstacle but leads to an irregular and intermittent motion when in the proximities of the obstacle. To provide a smoother movement, a swirling motion must be added to the potential function.

If the swirling direction is static the RPA will always perform the same maneuver so, to comply with the avoidance logic defined in Sec. 4.2.1, the swirling direction must depend on the type of encounter. A repulsive field with an imposed swirl is plotted in Fig. 4.9, where the swirl direction is given by

$$\mathbf{S}_{dir} = \frac{\mathbf{S}_z \times \mathbf{d}_o}{\|\mathbf{d}_o\|}, \tag{4.7}$$

where S_z is the unit vector in the z direction. This will originate a counterclockwise movement. If the order of the cross product in Eq.(4.7) is switched the field will rotate in the clockwise direction.

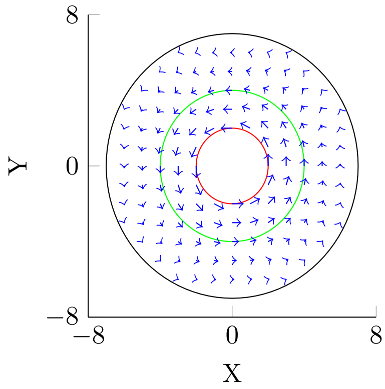


Figure 4.9: Swirling potential flow.

The imposed swirling however can lead to the vehicle being trapped around the obstacle or even enter in a collision course. To avoid this situation, the effect of the swirl is neglected once the obstacle is overcome. To do so, the angle between the desired direction of motion, **D**, and the relative position between RPA and obstacle is evaluated,

$$ang = \frac{acos(\mathbf{D} \cdot \mathbf{d}_o)}{\|\mathbf{D}\|\|\mathbf{d}_o\|}. \tag{4.8}$$

When this angle is bigger than a predefined cut-off angle, $\theta_{cut-off}$, the repulsive potential is ignored and only the attractive term is considered. The total repulsive force is then given by Eq. 4.9, which is the combination of the previous terms,

$$\mathbf{F}_{rep} = \begin{cases} 0, & \text{if } d_o \leq R_a \quad or \quad ang \geq \theta_{cut-off} \\ -\frac{\mathbf{d}_o}{\|\mathbf{d}_o\|}\left|\frac{R_t - \|\mathbf{d}_o\|}{R_t}\right|\mathbf{S}_{dir}, & \text{if } R_c \geq d_o \leq R_a \\ \infty, & \text{if } d_o \leq R_c \end{cases} \tag{4.9}$$

Total Potential Function

The total potential flow force, which determines the movement direction, is given by

$$\mathbf{F}_{tot} = \mathbf{F}_{at} + \sum \mathbf{F}_{rep}. \tag{4.10}$$

From the total field vector the required heading and flight path angles to avoid the obstacle are obtained, from which, knowing the current direction of motion, a series of waypoints are generated until the obstacle has been cleared. However, the combination of both the attractive and repulsive potential can lead to heading changes that are not feasible by the RPA, so the angle between the platform current heading and $\mathbf{F}_{tot}$ is taken. If this angle is greater than the maximum turning angle, the angle is scaled to the maximum allowable value. The same applies to the climb angle.

Outside the detection region, the tracking module is responsible for following the reference path. Inside that area, the reference trajectory is updated to avoid any possible collisions. Once the collision risk is overcome, the closest point on the reference path is found and the navigation system returns to following the original path.

One issue may arise when an intruder obstructs one of the required waypoints defined during the mission planning stage. This can be seen in Fig. 4.10, where the blue line represents the global trajectory to be followed and the red dot is a waypoint of required passage.

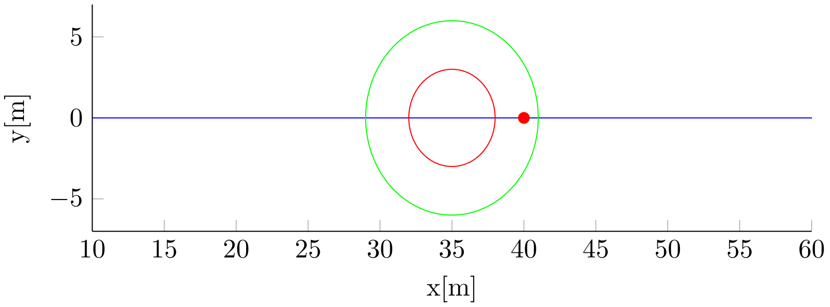


Figure 4.10: Example of obstructed waypoint.

If the obstacle is static, there is no way to go through the required waypoint without violating the safety distance, however if the obstacle is moving, it is possible to return to the required waypoint once the collision has been avoided. To do so, instead of returning to the global path once the threat is overcome, the attractive potential function for a waypoint is activated and a path that directs the RPA towards the missed waypoint is computed. When the waypoint has been passed over, or within a predefined distance, the RPA returns to the global path.

The overall algorithm flowchart for the path replanning stage is presented in Fig. 4.11. The path replan module receives information about the RPA state (position and velocity), the path to follow and the obstacles position and velocity. A check is made to see if there is risk of collision and the CPA is determined. If several conflicts are detected, the threats are prioritized with respect to the time to collision. Depending on the direction of approach, the type of maneuver is determined (turn right/left, climb/descent). With the type of maneuver defined, the potential fields method is applied to generate and avoidance path until the conflict point has been avoided. Once the situation is clear, and if any key waypoint has been missed and it is not blocked by a static obstacle, the potential fields approach is applied, with waypoint guidance, as in Eq. 4.5, until the waypoint has been reached. Once the conflict has been avoided and the waypoint has been visited, the path is updated.

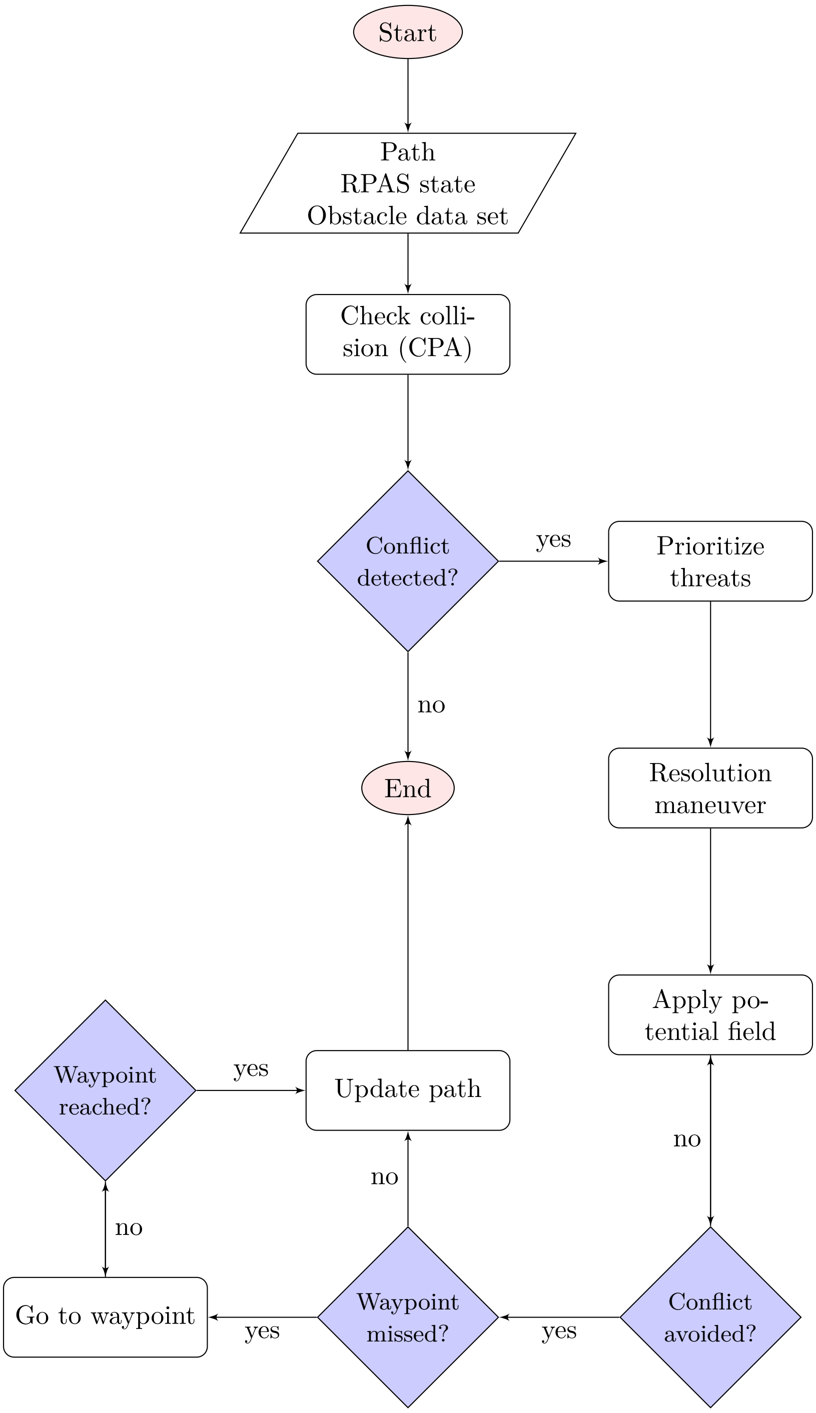

Start
Path
RPAS state
Obstacle data set
Check collision (CPA)
Conflict detected?
yes
Prioritize threats
no
End
Resolution maneuver
Apply potential field
Waypoint reached?
yes
Update path
no
no
no
Go to waypoint
yes
Waypoint missed?
yes
Conflict avoided?

Figure 4.11: Path replan flow chart.

Chapter 5

Results

In this chapter the developed algorithms are tested in several case scenarios to evaluate their strengths and drawbacks. All examples are obtained with MATLAB R2016a running on a Intel Core i5 with a CPU of 2.4 GHz, 4Gb RAM and Windows 7.

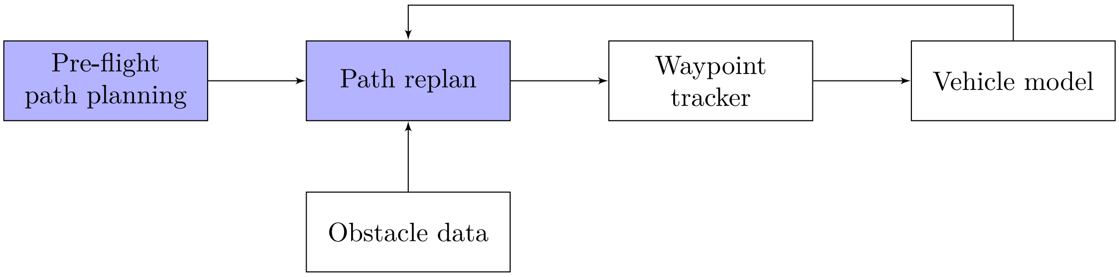


Figure 5.1: Block diagram of tested system.

Fig. 5.1 shows the system used to test the algorithms. The vehicle flies at constant speed, and a simple agent model is used to describe it,

$$\begin{aligned} x_{t+1} &= x_t + v cos(\psi_t) cos(\gamma_t) \Delta t \\ y_{t+1} &= y_t + v sin(\psi_t) cos(\gamma_t) \Delta t \\ z_{t+1} &= z_t + v sin(\gamma_t) \Delta t \end{aligned} \tag{5.1}$$

where the RPA position is denoted as (x, y, z), v is the airspeed, ψ the heading angle and γ the flight path angle, as represented in Fig. 3.8.

The pre-flight path planning module determines an optimal path between the given reference waypoints in a known static environment. The path replan module receives the reference path, the RPA position and velocity and the obstacle data as depicted in Tab. 2.1. If a conflict is detected, the reference path is updated. The waypoint tracker receives the planned path waypoints and determines the required heading, ψ, and flight path angle, γ, that are used as input to eq. 5.1. It is assumed that the values are tracked perfectly.

5.1 Pre-Flight Path Planning

In this first examples planning between a set of waypoints in the presence of several static obstacles is addressed. The presented results consider a fixed-wing and multirotor platforms with the parameters described in Tab. 5.1.

Variable	Value		Description
	Fixed-wing	Multirotor	
v	16	16	Airspeed$[m/s]$
m	2	1	Mass$[kg]$
S	1.5	-	Wing span$[m^2]$
g	9.81	9.81	Gravitational acceleration$[m/s^2]$
C_d	0.02	0.01	Drag coefficient
R_{curv}	10	5	Minimum turning radius$[m]$
γ_{max}	30	90	Maximum flight path angle$[°]$

Table 5.1: RPAS parameters.

The parameters used in the ACO algorithm are given in Tab. 5.2.

α	β	$N°ofants$	q_0	τ_0	ρ	ξ	Iterations	τ_{min}	τ_{max}
1	2	10	0.9	1	0.9	0.9	500	0.1	10

Table 5.2: ACO parameters.

To compare the developed methods, the results are presented for the path given by the A* and ACO algorithms, P_i; the initial rational Bezier curve determined from the given path with unitary weights, B_i; the rational Bezier curve with adjusted weights to only satisfy the constraints (the cost function is set to zero in the optimization problem defined in Sec. 3.3), B_c; and the Bezier curve optimized for the given cost function, B_o.

5.1.1 Minimum Distance Paths between Two Waypoints

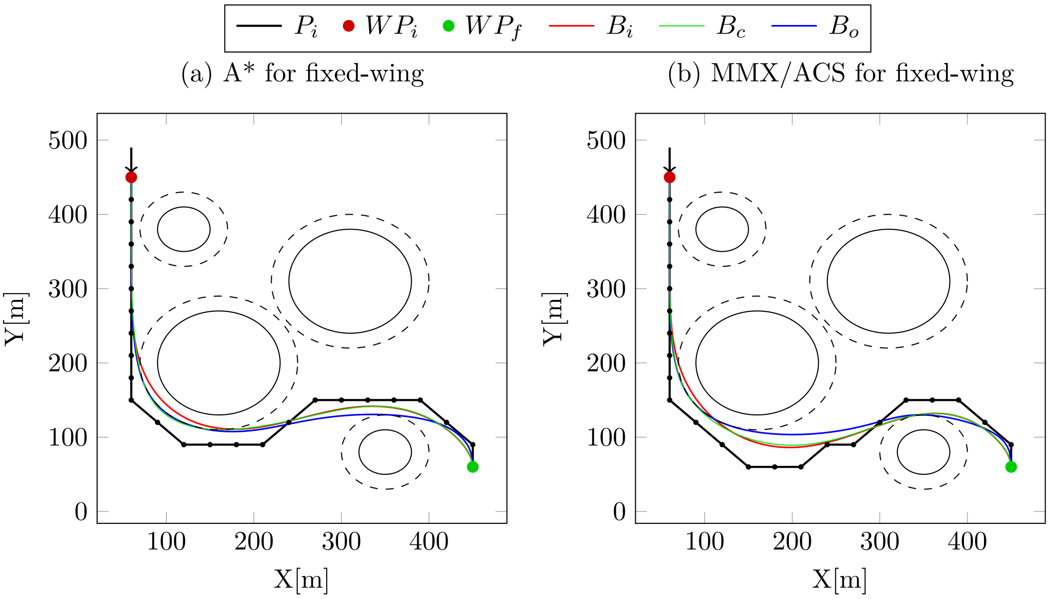
The environment considered in this case has the characteristics presented in Tab. 5.3, and two distinct departure headings are tested for the fixed-wing platform. Results are presented for minimum distance paths. In this first example, the potential fields method is also used to compare it with the global methods, the used parameters are presented in Tab. 5.4. The planned paths can be seen in Figs. 5.2 and 5.3.

Map size	$\Delta x[m]$	$R_s[m]$
400	30	20

Table 5.3: Map characteristics.

R_d	R_a	R_s	α_{PF}	$\theta_{cut-off}[deg]$
100	10	20	0.5	30

Table 5.4: Potential fields parameters.



(a) A* for fixed-wing

(b) MMX/ACS for fixed-wing

(c) A*/MMX/ACS for multirotor/fixed-wing

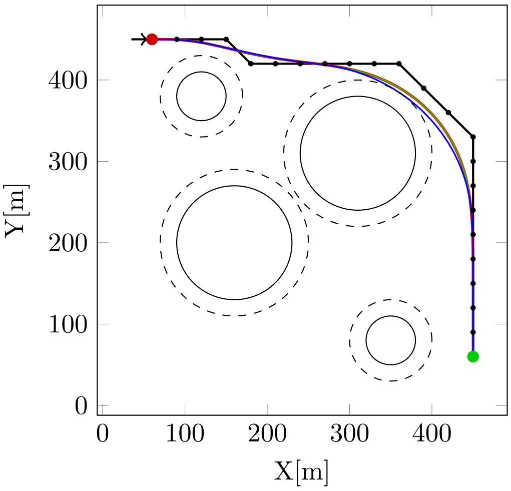
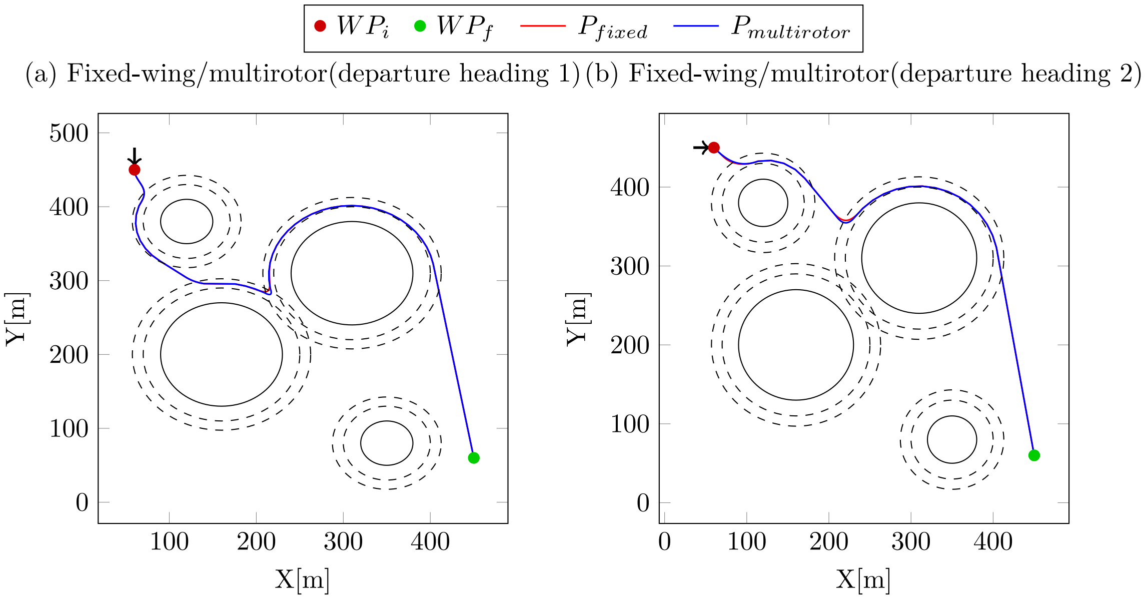


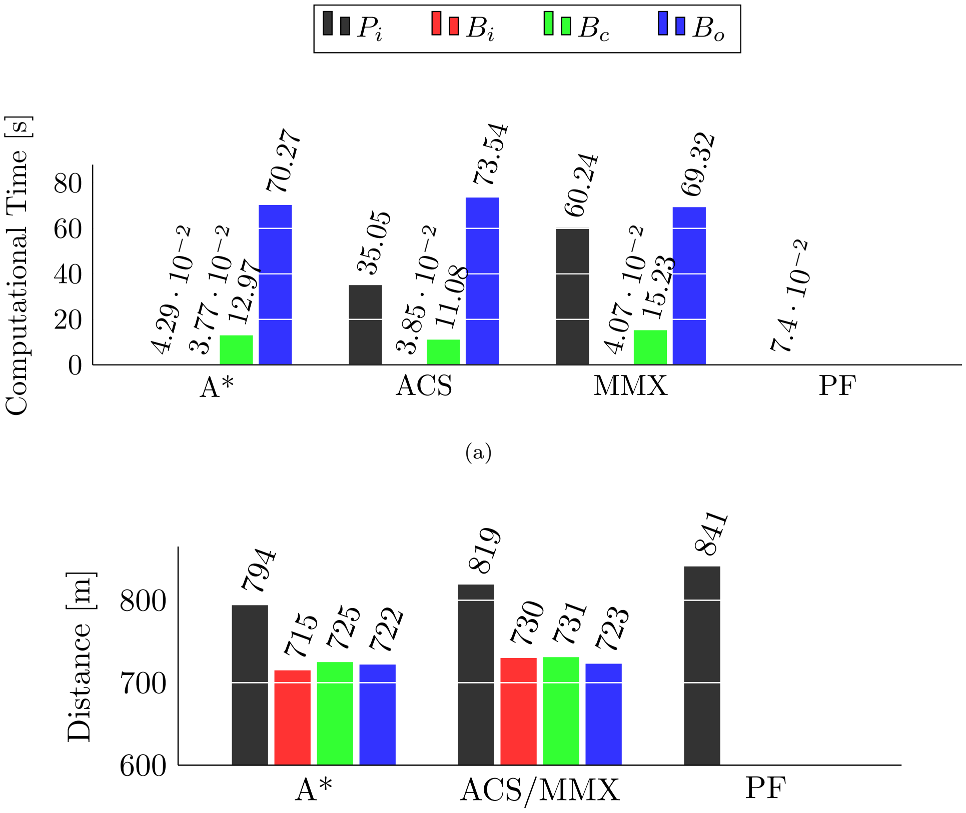
Figure 5.2: Minimum distance paths for the A* and ACO algorithms.



(a) Fixed-wing/multirotor(departure heading 1) (b) Fixed-wing/multirotor(departure heading 2)

Figure 5.3: Planned paths for the Potential Fields algorithm.

The potential fields algorithm is clearly not fitted for global planning as the obtained path is not optimal. For the global planning methods, except in the case of Fig. 5.2(c) where the obtained results are the same, the optimal path is obtained with the optimized Bezier curve from the A* path. Both the ACS and MMX algorithms result in longer paths. It is also noted that the departure heading for a fixed-wing platform will influence the resulting path. In this situation the cost function for minimum distance was used, but the same results are obtained if minimum energy paths are computed. The results comparing computational time and path cost are presented in Figs. 5.4 and 5.5.



(a)

(b)

Figure 5.4: (a) Processing time of the several algorithms and (b) path cost(distance) for the fixed-wing case represented in Figs. 5.2(a) and (b) and Fig. 5.3(a).

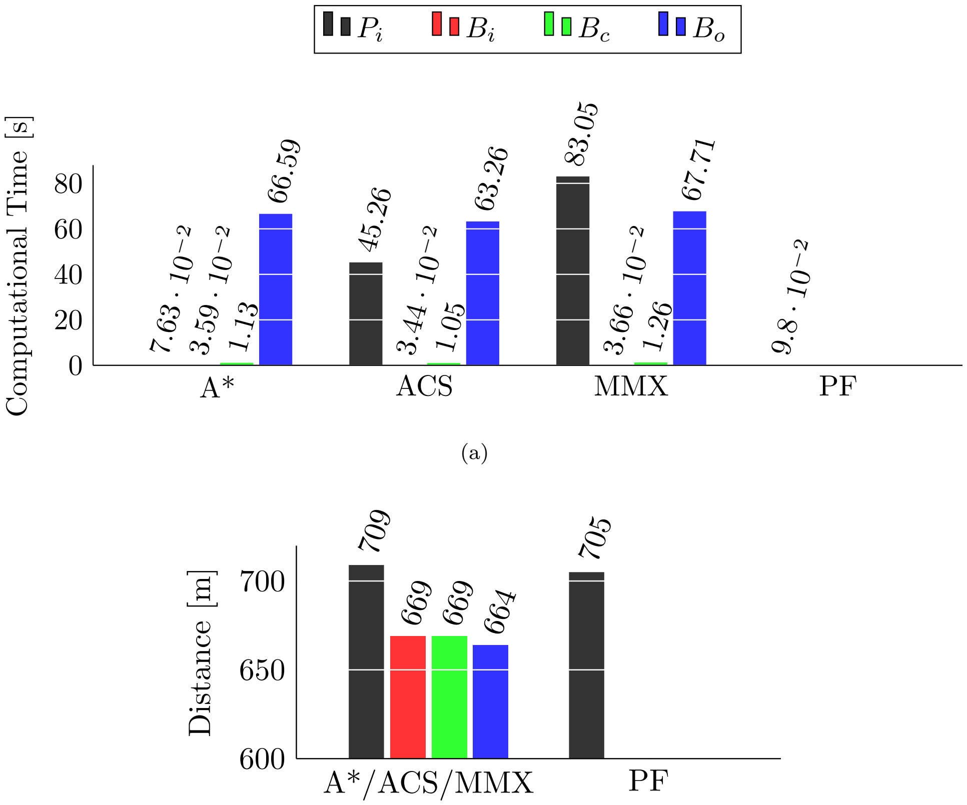
Figure 5.4 shows the computational time and path cost for the paths obtained for the fixed-wing platform, with the first departure heading, using A*, both ACO variants and potential fields. In the potential fields method, there was no significant difference between the fixed-wing and multirotor results.

For the planning of the coarse path P_i, A* was the fastest. Both ACO algorithms have a longer computational time, and the addition of the pheromone limitation step in the MMX algorithm makes it slower than ACS, however the same path geometry is obtained in both cases.

The initial Bezier curve, B_i, is of fast computing and provides a significant improvement over the solution cost of the initial path. The constrained Bezier curve, B_c, which is calculated to only satisfy the constraints, takes longer to compute and leads to a longer path than the initial curve to avoid violating safety constraints. The optimized Bezier curve has the longest computation time but it results in the minimum cost path.

Both ACO algorithms are slower than A*, and due to its random nature, they result in a path longer than the A* algorithm, however the optimization of the Bezier curve is able to find a path which is only 1 meter longer than the optimized curve obtained from the A* path.

The potential field method was the fastest to compute the final path but, being a local method, it leads to the worst results in terms of path cost.



(a)

(b)

Figure 5.5: (a) Processing time of the several algorithms for the multirotor platform and (b) path cost (distance) for the fixed-wing and multirotor cases represented in Figs. 5.2(c) and Fig. 5.3(b).

Figure 5.5(a) sows the computational time for the multirotor platform. Figure 5.5(b) shows the path cost for the paths obtained for the fixed-wing platform with the second departure heading and the multirotor.

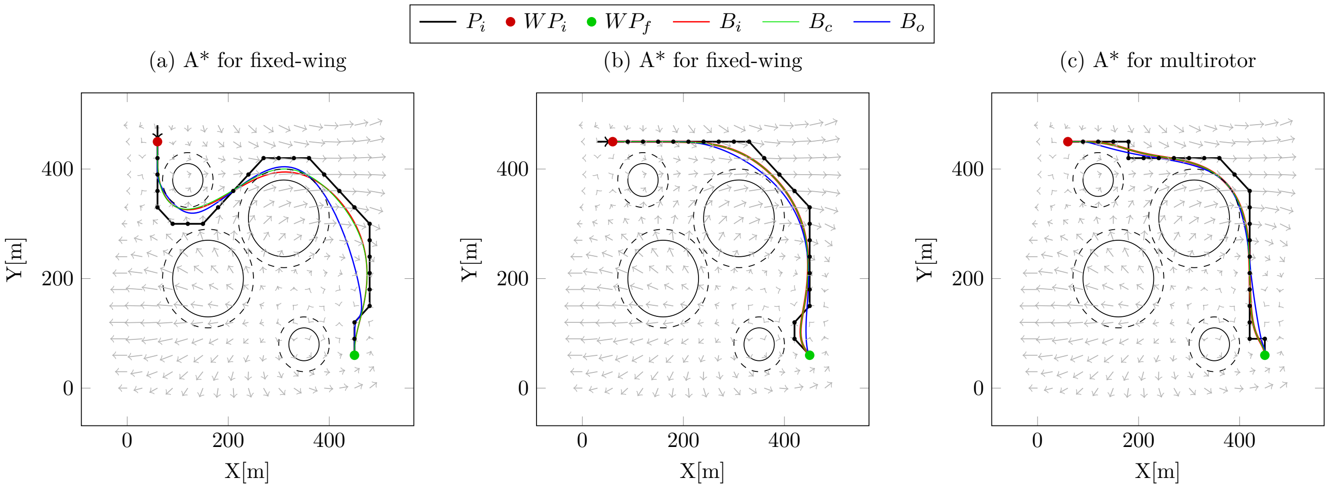
In this situation, the same results are obtained using A*, ACS and MMX in terms of path cost. A* was the fastest to compute the initial path, and again MMX was slower that the ACS. The computational time was significantly higher than in the previous case due to the increase in number of node expansion for the multirotor platform when compared to the fixed-wing.

Both the initial and constrained Bezier curve are the same, as no constraints were initially violated. The optimized curve took longer to compute but resulted in the minimum cost path.

Again, the potential fields method was the fastest, but it resulted in the worst path in terms of cost.

5.1.2 Minimum Energy Paths between Two Waypoints

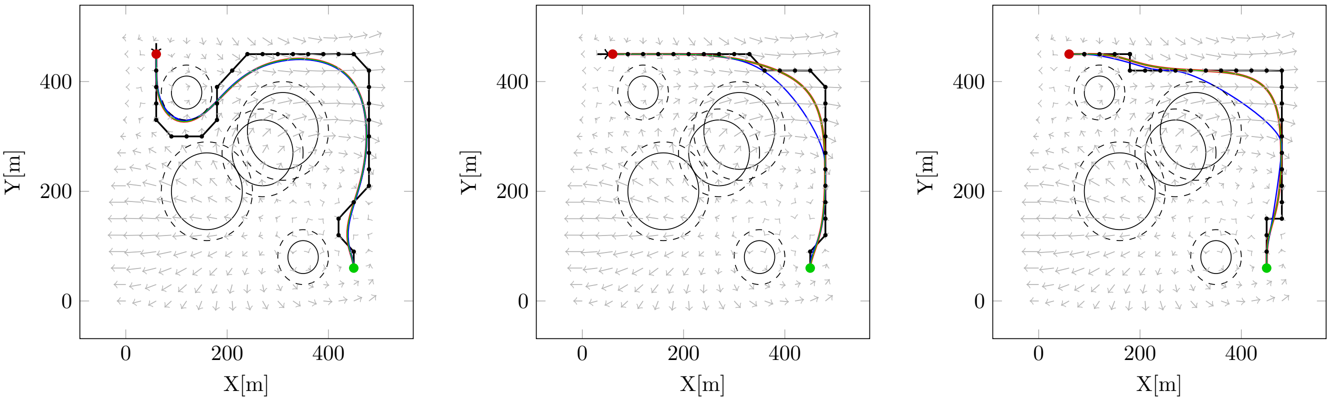
In this example, the previous environment is used but the algorithms are now evaluated for the minimum energy paths in the presence of a wind field. Its intensity varies up to $5m/s$ and the field direction was generated to favor a specific route. The planned paths are depicted in Fig. 5.6. Different departure headings are tested for the fixed-wing platform.



(a) A* for fixed-wing

(b) A* for fixed-wing

(c) A* for multirotor



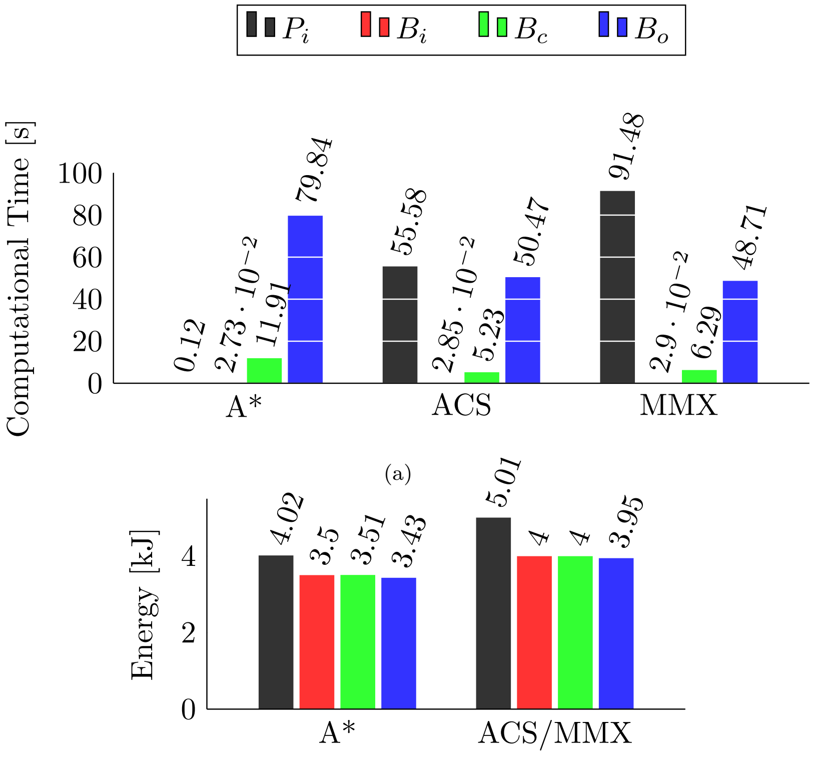
(d) ACS/MMX for fixed-wing

(e) ACS/MM for fixed-wing

(f) ACS/MMX for multirotor

Figure 5.6: Minimum energy paths for the A* and ACO algorithms, for both fixed-wing and multirotor platforms.

The obtained paths differ from the minimum energy paths without wind (which are equivalent to minimum distance paths of Sec. 5.1.1). In this situation the best results are given by the optimized Bezier curve obtained from the waypoints generated by the A* algorithm. Like in the previous example the same path is given by the ACS and MMX algorithms. In Fig. 5.6(a) the initial Bezier curve violates the safety distance, but both the optimized and the constrained curves are able to respect the safety distance.



(a)

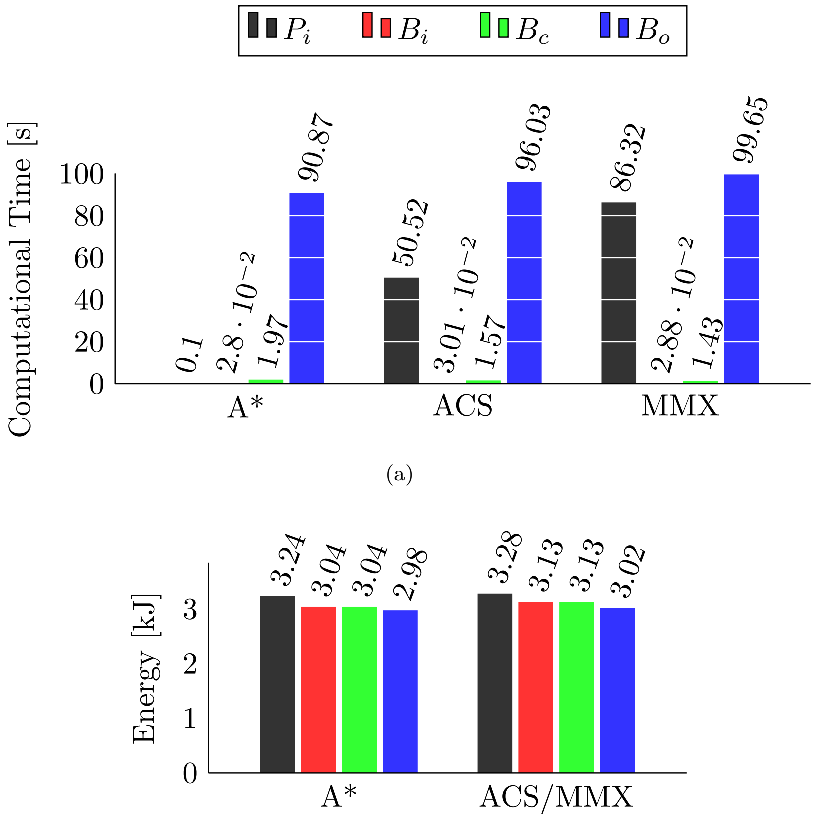
(b)

Figure 5.7: (a) Processing time of the several algorithms and (b) path cost (energy) for the case represented in Figs. 5.6(a) and(d).

Figure 5.7, shows the processing time and path cost obtained using A* and ACO, for the fixed-wing platform with the first departure heading.

The A* algorithm is the fastest to find the initial path while the MMX takes the longest time. The constrained Bezier curve for the A* algorithm results in a path of higher cost when in comparison to the initial curve B_i, this is to maintain the safety distance.

The path of minimum energy is found by the optimized Bezier curve using the original A* path. For the paths obtained with ACS and MMX both the initial and constrained Bezier curves have the same cost because the constraints where met.



(a)

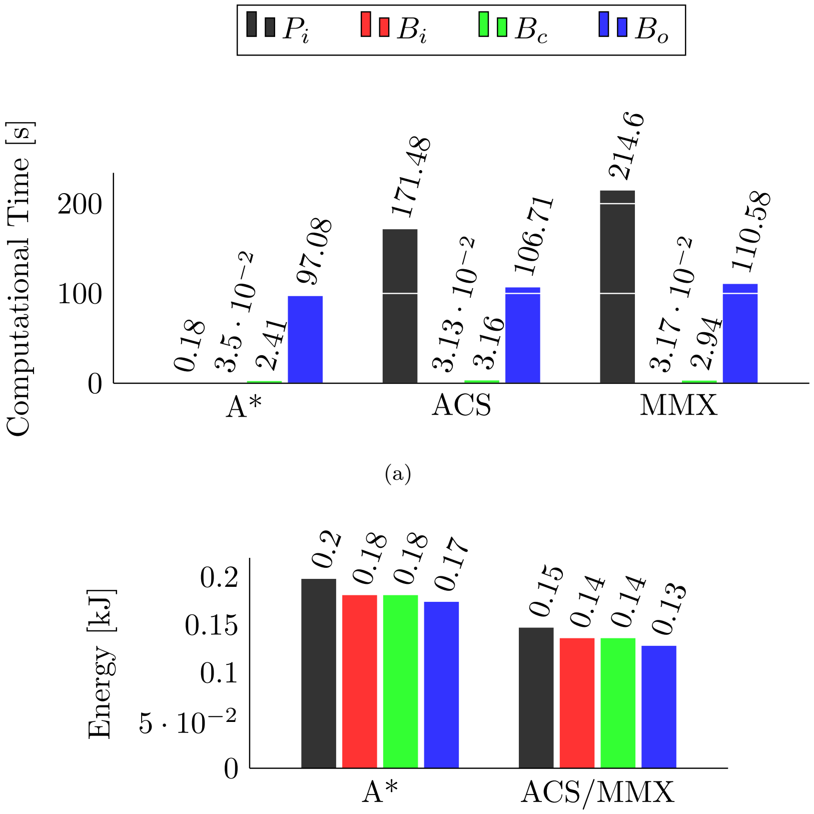
(b)

Figure 5.8: (a) Processing time of the several algorithms and (b) path cost, evaluated according to distance for the case represented in Figs. 5.6(b) and(e).

Figure 5.8, shows the processing time and path cost obtained using A* and ACO, for the fixed-wing platform with the second departure heading.

A* was the fastest algorithm followed by ACS and MMX. In terms of path cost the best results, for initial path and the resulting Bezier curves, are given once more by the A* algorithm.

For both A* and ACO the constrained and initial curves have the same cost and only a small improvement is achieved through the optimization of the Bezier curve.



(a)

(b)

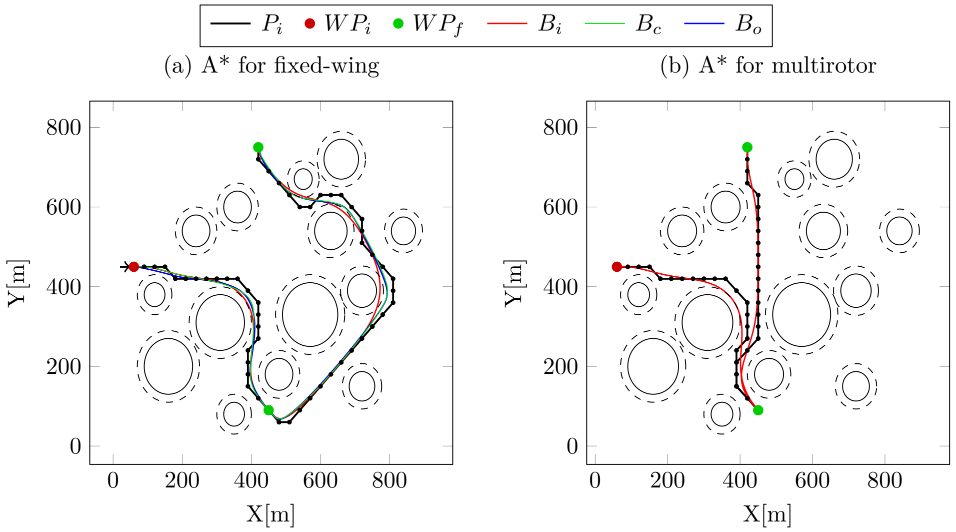
Figure 5.9: (a) Processing time of the several algorithms and (b) path cost, evaluated according to distance for the case represented in Figs. 5.6(c) and(f).

Figure 5.9, shows the processing time and path cost obtained using A* and ACO, for the multirotor platform. Similar conclusions to the ones in the previous case of Fig. 5.8 can be drawn. The only noticeable difference is the increase in computational time, mainly in the ACO algorithms, in comparison with the fixed-wing case. This shows the influence of the search space dimension in the processing time of the algorithms.

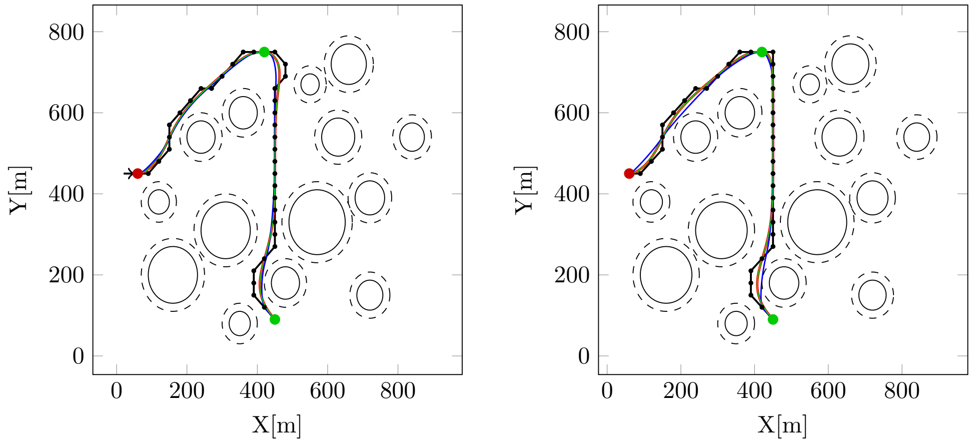
Again, the best results are obtained through the optimized Bezier curve of the A* path. However, the increase in computational time versus the decrease in the cost function value when compared with the constrained curve B_c, shows that the optimization may have no practical interest.

5.1.3 Minimum Distance Paths with Three Waypoints

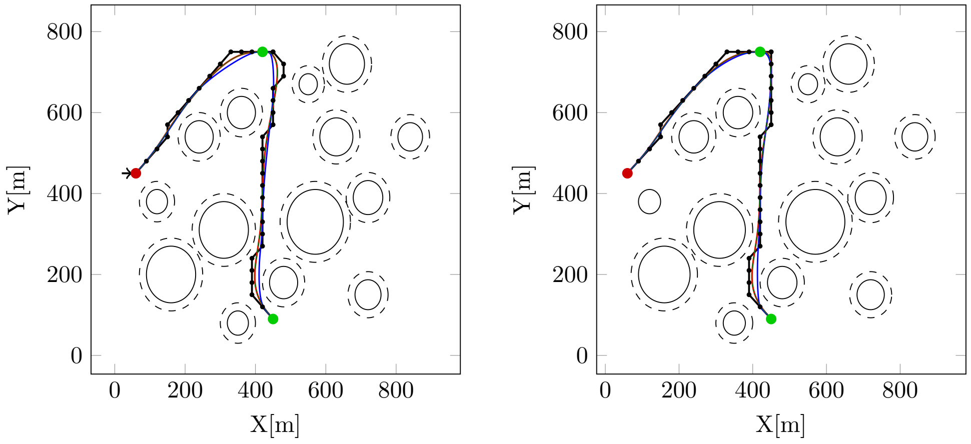
In this example, the previous case is expanded and three waypoints are considered.


(a) A* for fixed-wing

(b) A* for multirotor


(c) ACS/MMX for fixed-wing

(d) ACS/MMX for multirotor

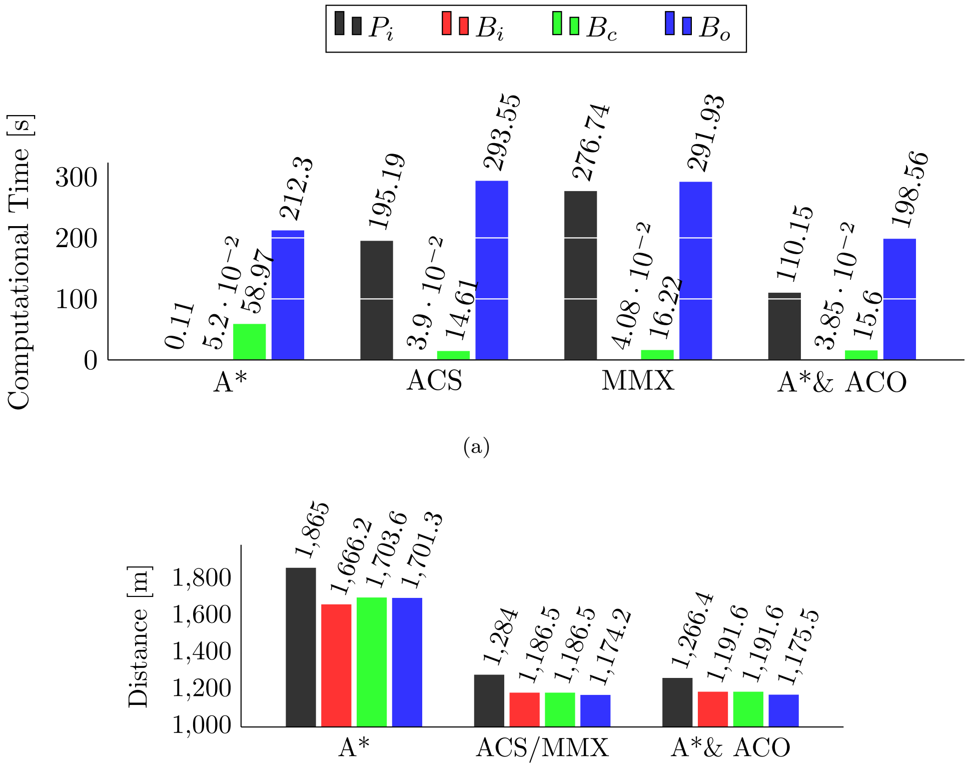

(e) A* and ACO for fixed-wing

(f) A* and ACO for multirotor

Figure 5.10: Minimum distance paths. (a),(b): A* with random waypoint order;(c),(d): ACS and MMX with random waypoint order; (e),(f): A* with optimal waypoint order given by ACO.
.

The reference waypoints to visit are given in a random order to both the A* and the ACO algorithms separately. The optimal order obtained from ACO is then feed to the A* algorithm. The planned paths are depicted in Fig. 5.10.

It is concluded that the best results are obtained by using ACO to optimize the reference waypoint order and A* to plan the path between those waypoints. In Fig. 5.10(b) it is noted that the expansion rules defined for the multirotor lead to a planned path that cannot be followed at constant speed, requiring the platform to stop and reverse its direction to follow the path. Again, in Fig. 5.10(a) the optimized and constrained paths are longer than the initial one because a deviation has to be made to avoid the safety radius of the obstacles. Also slightly shorter paths are obtained for the multirotor platform due to its smaller turning radius.



(a)

(b)

Figure 5.11: (a) Processing time of the several algorithms and (b) path cost(distance) for Figs. 5.10 (a), (c) and (e).

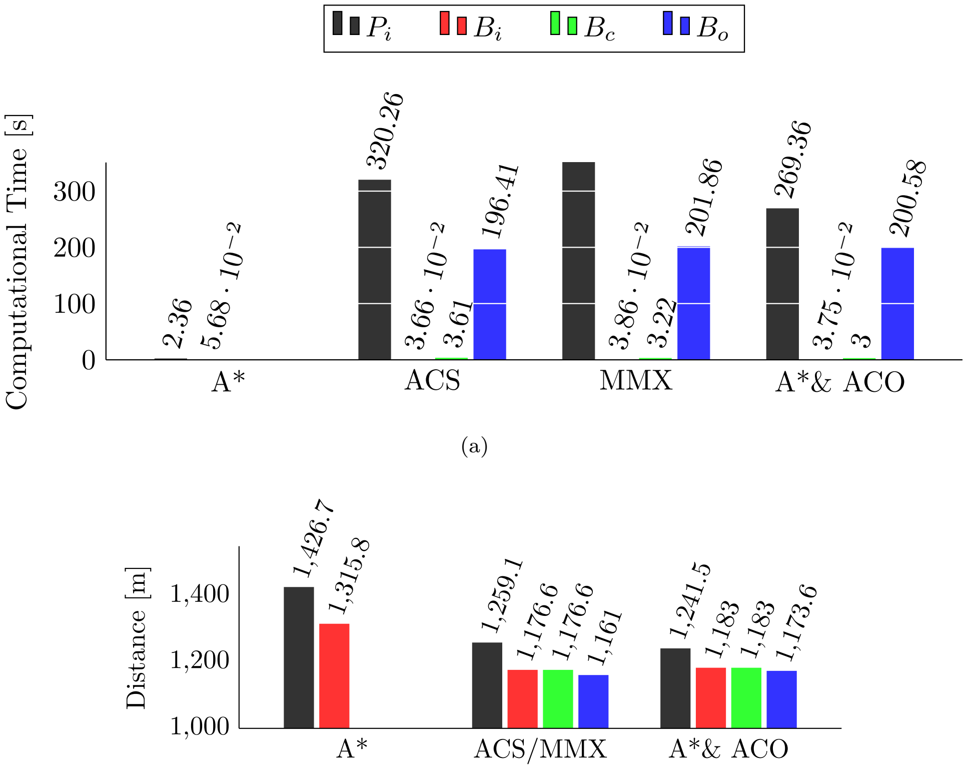
Figure 5.11 shows the results for computational time and path cost for the fixed-wing platform in three situations: A* and ACO with random waypoint order and A* with the order given by ACO.

The A* algorithm is only able to find the optimal path between waypoints and cannot optimize its order, as it can be seen in the difference in path cost between A* and the ACO variants.

As the environment complexity increases the computation time also grows.

When combining A* and ACO, the ACS version is used as it is faster and provides the same results as MMX. By combining A* and ACO a compromise is achieved in terms of computational time, as the ACO is only used to find the optimal sequence of waypoints and A* to connect them.

It is also noted that even though for the coarse path P_i, the best cost is given by the combination of A* and ACO, the Bezier curves are shorter for the control points given by any of the ACO algorithms.



(a)

(b)

Figure 5.12: (a) Processing time of the several algorithms and (b) path cost(distance) for Figs. 5.10 (b), (d) and (f).

Figure 5.12 shows the results for the multirotor example in the same instances as previously.

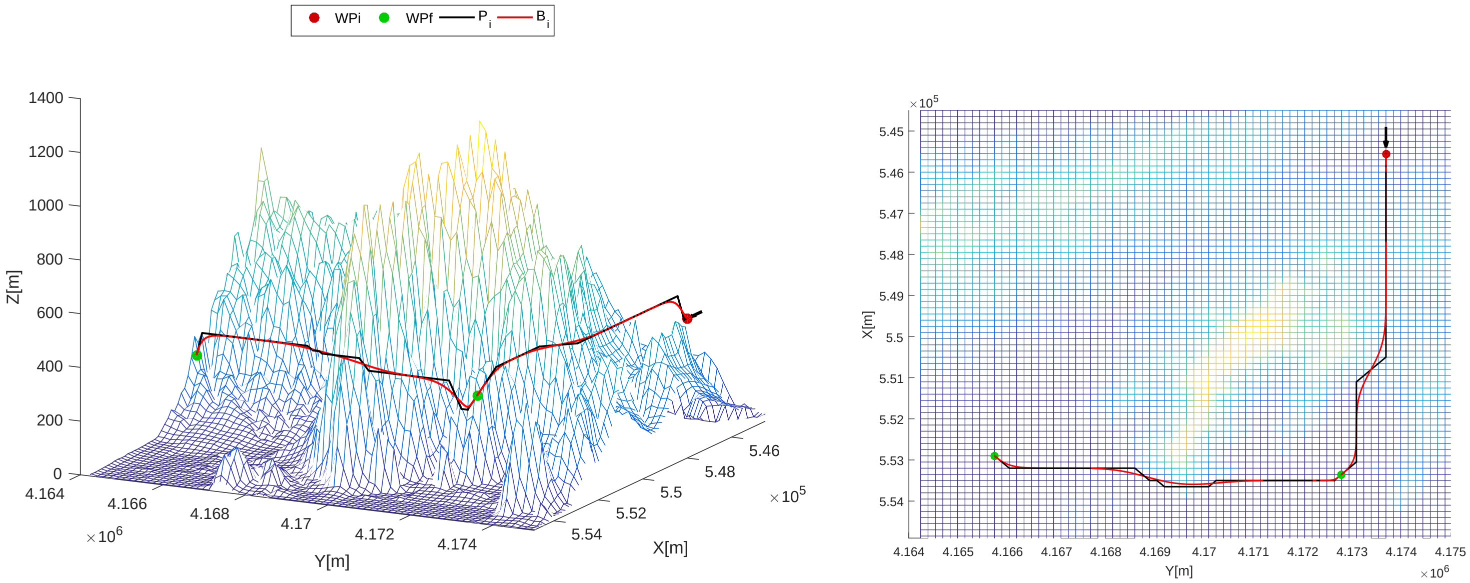
When giving to the A* algorithm the same waypoint order as to the fixed-wing example a different path is obtained as seen in Fig. 5.10(b). This is due to the expansion rules defined for this type of vehicles. The immediate inversion of direction at the second waypoint makes it impossible to meet the turning radius constraint, however the multirotor ability to hover enables turning but requires a speed change.

Even though A* finds a shorter coarse path, like in the previous case, the position of the path points given by ACO allows the generation of a shorter Bezier curve.

5.1.4 Path Planning in a Digital Elevation Model

In this example, a tridimensional environment is considered with three waypoints to visit. The ACO algorithm is used to optimize the waypoint order and the A* to plan the path between waypoints. If no constraints are violated by the initial Bezier curve, no optimization is made. A DEM is used to simulate the environment. Both minimum distance and minimum energy paths for a fixed-wing platform are tested.

The minimum distance paths are depicted in Fig. 5.14 and the minimum energy paths are shown in Fig. 5.13. The results for both cases are presented in Tab. 5.5.



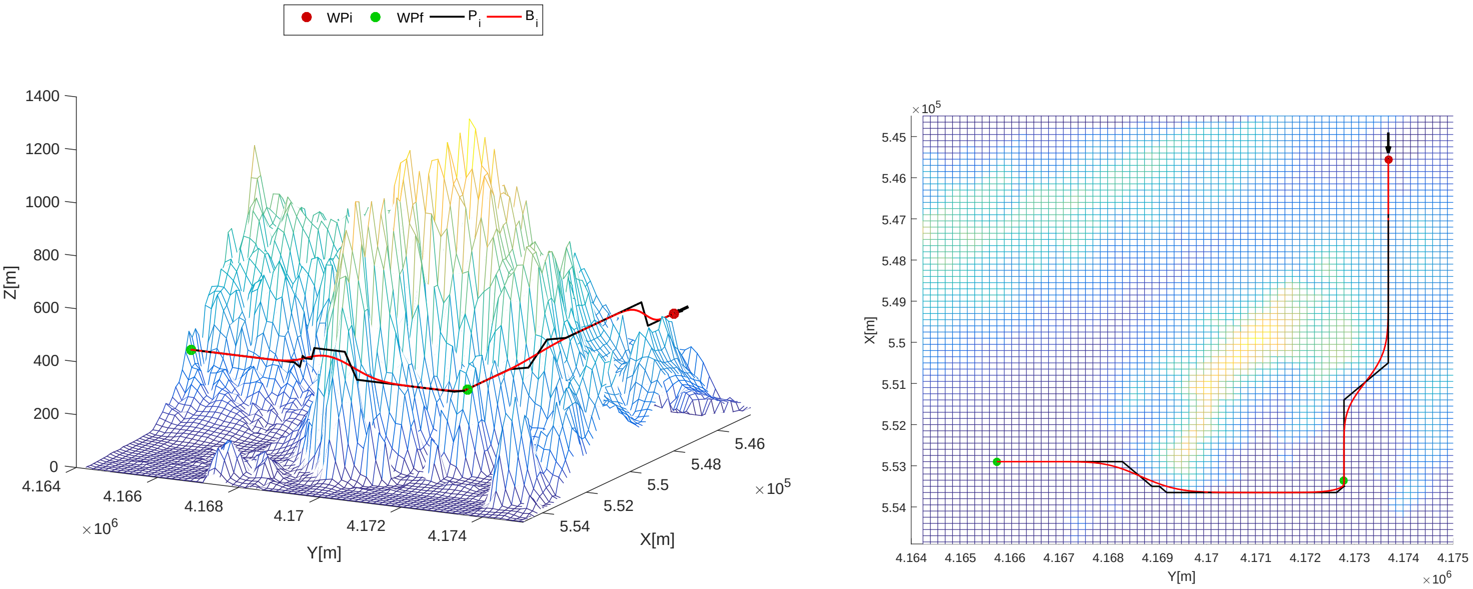
(a) (b)

Figure 5.13: Minimum energy path planned for a fixed-wing platform: (a) 3D view, (b) 2D view.

Cost function	$Time[s]$	$Distance[m]$	$Energy[J]$
D_{min}	5.560	1.3513e+03	8.5813e+04
E_{min}	472.470	1.4863e+03	8.6145e+04

Table 5.5: Computational time and cost function values for the resulting Bezier curve.

In this case, it can be seen that when using as heuristic the minimum energy cost function, the optimal path is not found. Even though that without wind the minimum energy path should be equivalent to the minimum distance path, as it happens in the example presented in Sec. 5.1.1, such does not happen here. This could be due to the fact that this a more complex configuration space and the energy cost function does not satisfy the requirements of admissibility and consistency required for the optimal path to be found by A*. The results are presented for a fixed-wing platform but the same paths are obtained for a multirotor, the computational time however is larger as the number of states to be explored increases.



(a) (b)

Figure 5.14: Minimum distance path planned for a fixed-wing platform: (a) 3D view, (b) 2D view.

5.2 Path Replanning

In this section, the Potential Fields algorithm is tested in several cases where the original path must be updated to avoid collisions with moving and unforeseen static obstacles. The parameters used for a fixed-wing platform in the following tests are presented in Tab. 5.6.

$R_{detection}$	α_{PF}	$\theta_{cutoff}[°]$	$\Delta\psi_{max}[°]$	$\Delta\theta_{max}[°]$
50	0.5	30	40	30

Table 5.6: Potential fields and vehicle parameters.

The path replanning module receives the path planned in the pre-flight module, the global path, and information about the obstacles present in the environment. If a new obstacle, that was not previously known, is detected the planning of a local segment to avoid it is activated and a new path is planned until the obstacle has been cleared. Instead of fully replanning the original path, only a local segment is calculated.

5.2.1 Head-on Scenario

In this example, several head-on conflict scenarios, where the RPA and the moving obstacle move in the same line but oposite directions, are tested. The parameters used for the three cases represented in Fig. 5.15 are presented in Tab. 5.7.

For case *a*, a fixed-wing RPA with a $1.5m$ wing span is considered, hence a collision radius around the CPA is defined with the same value. The purpose is to avoid a collision while maintaining a small deviation from the original path, so the safety radius is set to be at least equivalent to the collision radius. In case *b* the same situation is tested, but with a smaller safety radius. In case *c* the radii of example *a* are maintained but the speeds are increased.

Case	$R_c[m]$	$R_s[m]$	$v_{RPAS}[m/s]$	$v_{obs}[m/s]$
a	1.5	1.5	1	2
b	1.5	0.5	1	2
c	1.5	1.5	12	10

Table 5.7: Replaning scenario parameters.

Case	Minimum distance to obstacle[m]	Safety distance[m]	Computation time[s]
Figure 5.15(a)	3.1010	3	0.0147
Figure 5.15(b)	1.9713	2	0.0123
Figure 5.15(c)	3.4334	3	0.0098

Table 5.8: Head-on scenario results.

In Tab. 5.8 it can be seen that the RPA successfully avoids the intruder by deviating to the right without the occurrence of any collision. In the case of Fig. 5.15(b), where a smaller safety radius is defined, it is noted that the safety distance is compromised by a slight margin.

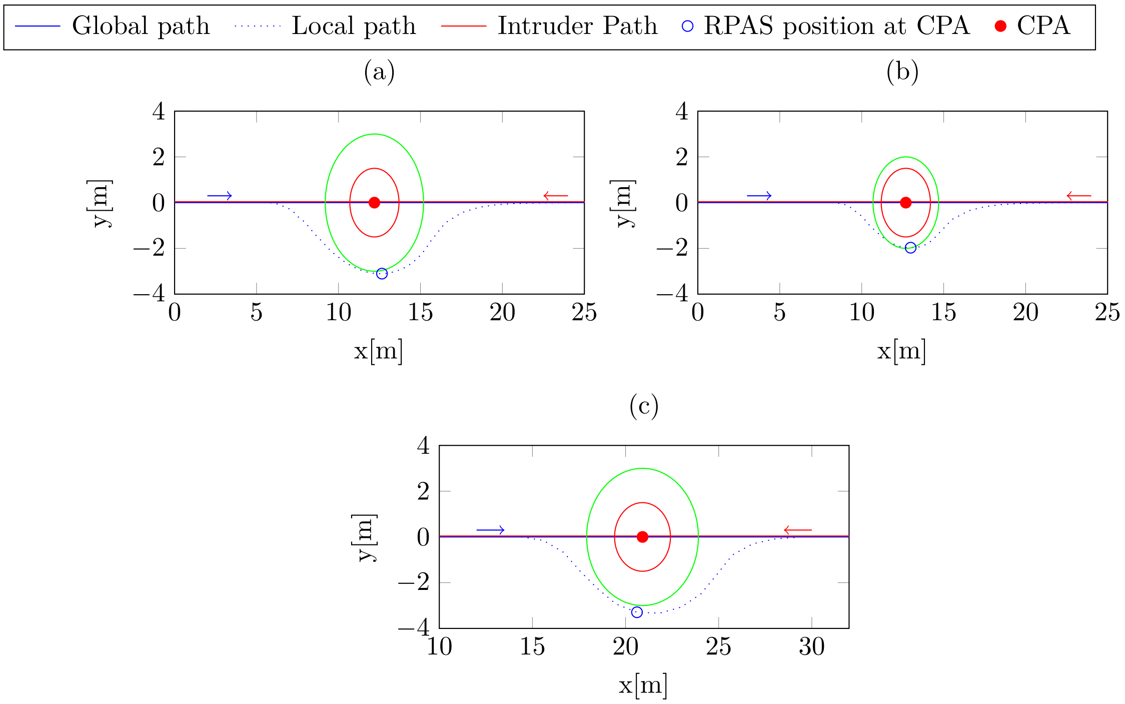


Figure 5.15: Head-on encounters. The obstacle is depicted with the safety (green) and collision (red) radius.

5.2.2 Converging Scenarios

In this section, different scenarios of converging encounters are tested.

Right Approach

Figure 5.16 shows several converging conflict scenarios, with an intruder approaching from the right. The same values as referred in Tab. 5.7 are used for the collision and safety radius and velocities.

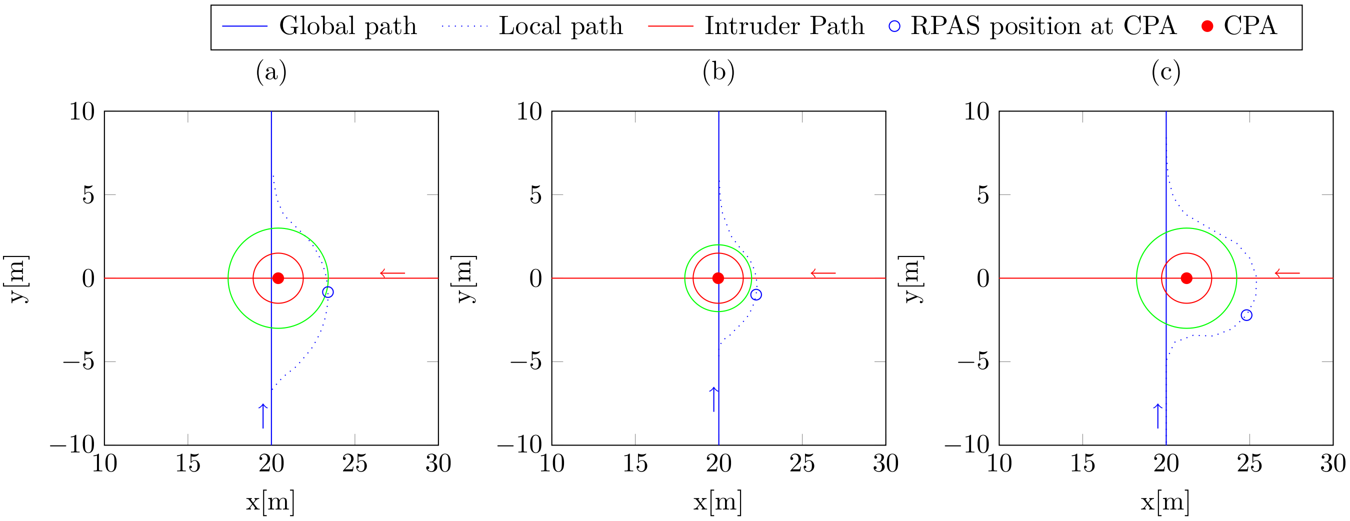


Figure 5.16: Right approach converging conflicts.

Case	Minimum distance to obstacle[m]	Safety distance[m]	Computation time[s]
Figure 5.16(a)	2.9996	3	0.0158
Figure 5.16(b)	1.8492	2	0.0175
Figure 5.16(c)	4.1691	3	0.0104

Table 5.9: Right approach scenario results.

As the intruder approaches from the left, the collision is avoided by planning a new path that makes the RPA turning right to avoid the intruder. In Fig. 5.16(a) a collision is successfully avoided and the RPA comes to the limit of the safety distance. In Fig. 5.16(b), even though at the CPA point the vehicle is beyond the safety range, the safety distance is not respected throughout the entirety of the replanned segment. Like in the previous case, the situation where the vehicles move at a faster speed is the one where the largest distance to the obstacle is maintained.

Left Approach

Figure 5.17 shows a set of converging conflict scenarios with a left approaching intruder and safety and velocity values as presented in Tab. 5.7.

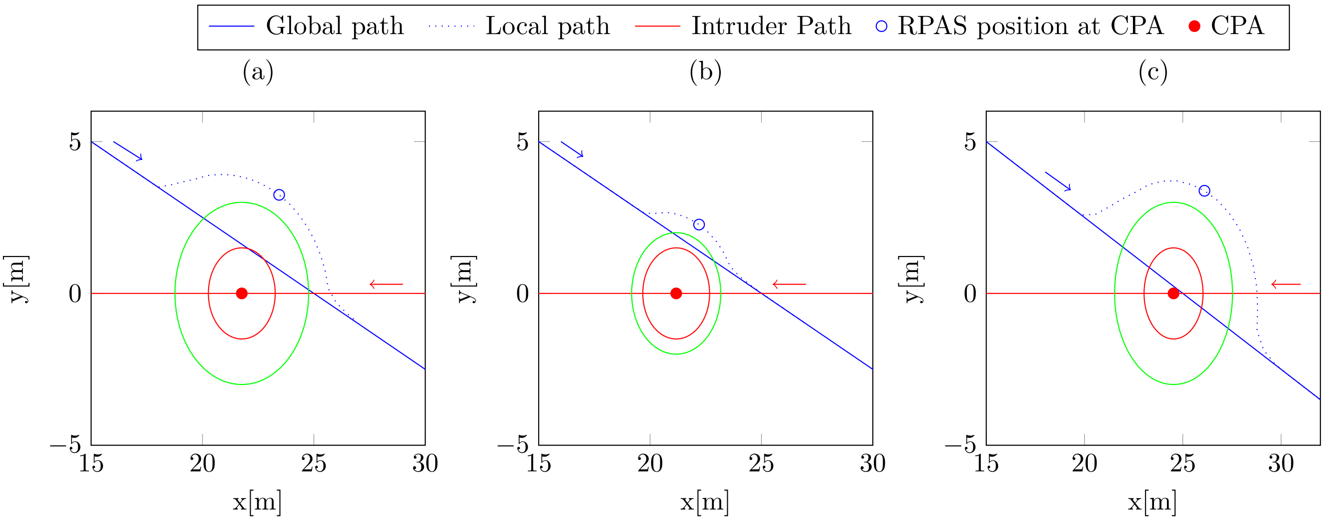


Figure 5.17: Left approach encounters. The obstacle is depicted with the safety (green) and collision (red) radius.

Case	Minimum distance to obstacle[m]	Safety distance[m]	Computation time[s]
Figure 5.17(a)	3.5216	3	0.0084
Figure 5.17(b)	2.4788	2	0.0082
Figure 5.17(c)	3.3681	3	0.0106

Table 5.10: Left approach scenario results.

In this example, the intruder is approaching from the left. To avoid passing on front of the obstacle, the new path is replanned according to the avoidance logic to deviate to the left and avoid the collision by going behind the intruder. From Tab. 5.10 it is seen that no collision occurs, and the safety distance is maintained in all situations.

Climb

Figure 5.18 shows conflict scenarios with the RPA in a climbing route and an intruder approaching in leveled flight. The parameters presented in Tab. 5.7 are used.

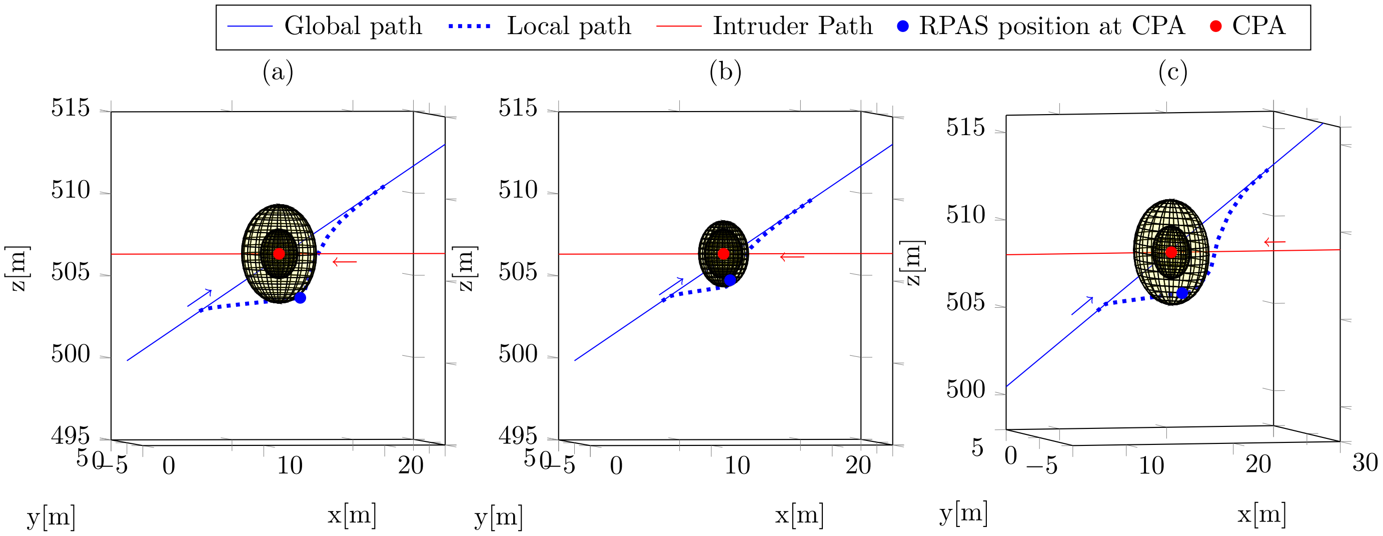


Figure 5.18: Climb encounters. The obstacle is depicted with the safety and collision spheres.

Case	Minimum distance to obstacle[m]	Safety distance[m]	Computation time[s]
Figure 5.18(a)	2.7230	3	0.0198
Figure 5.18(b)	1.7985	2	0.0206
Figure 5.18(c)	2.3759	3	0.0220

Table 5.11: Climb scenario results.

In this example as the RPA is climbing, the avoidance path is determined by leveling off the aircraft to avoid the collision point, and then climbing again to return to the original path. In Tab. 5.11 it can be seen that in all cases no collision occurs, but the safety distance is not maintained. In this situation a larger action radius must be set, so that the RPA begins its action sooner and keeps a bigger distance from the obstacle.

5.2.3 Missed Waypoint Scenario

In this example the RPA is following the reference path when a moving obstacle appears in a colliding course blocking one of the waypoints of mandatory passage. In this example a larger vehicle is considered and the following radius are defined: $R_c = 3m$ and $R_s = 3m$. The RPA moves at 2m/s and the intruder at 3m/s. The global and local paths are shown in Fig. 5.19.

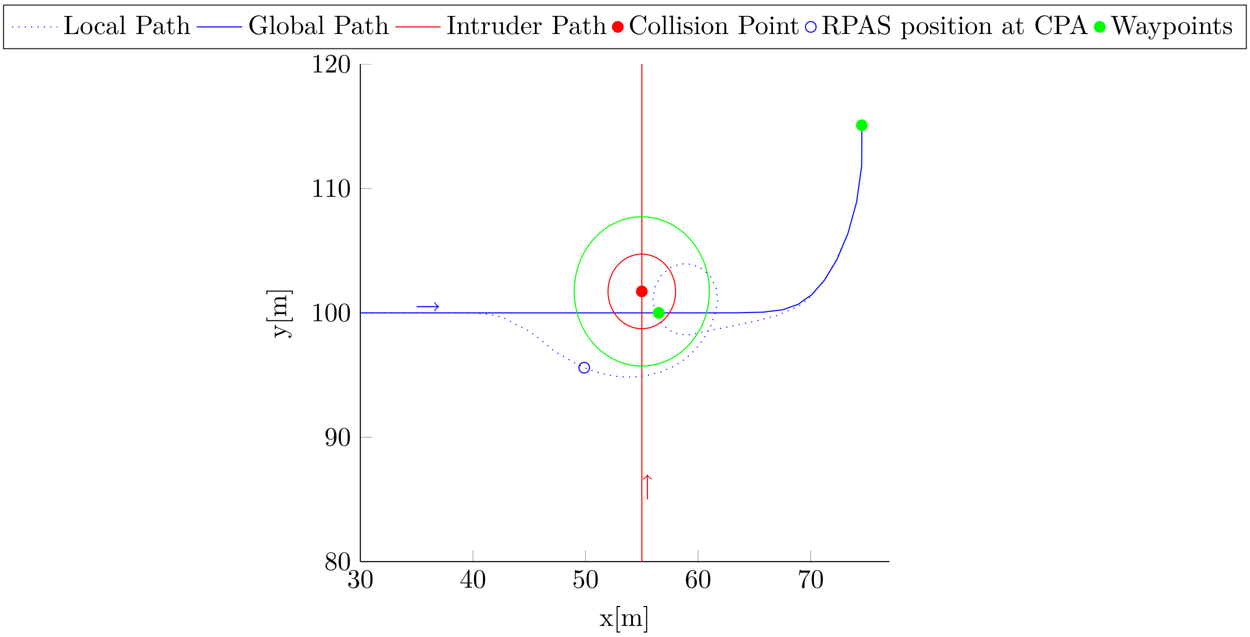


Figure 5.19: Converging encounter with missed waypoint. The obstacle is depicted with the safety (green) and collision (red) radius.

Minimum distance to obstacle[m]	Safety distance[m]	Computation time[s]
7.9852	6	0.015270

Table 5.12: Missed waypoint scenario results.

The conflict case is resolved by following the right of way rules, and replanning the path to deviate to the right. Looking at Tab. 5.12 it is seen that the safety distance is not compromised Once the obstacle is overcome, the RPA is directed to the missed waypoint. Once the waypoint has been reached the vehicle successfully returns to the reference path.

5.2.4 Replanning with Static and Moving Obstacles

In this example a reference path that goes through the required waypoints has been planned by the pre-flight path planner. During the execution of this path the RPA encounters two moving intruders, one of them blocking a pre-defined waypoint, and a static obstacle. The global and replanned path is depicted in Figs. 5.20 and 5.21. In this case th RPA moves at 1m/s and both moving obstacles travel at 1.5m/s.

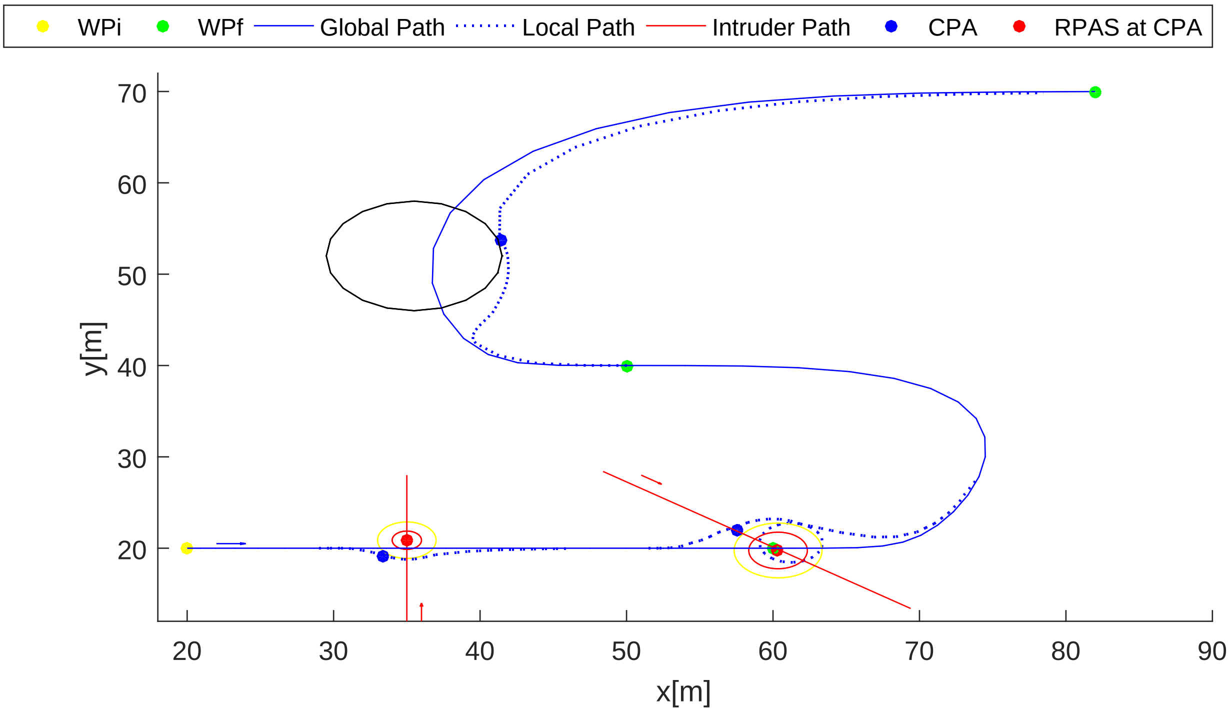


Figure 5.20: 2D view of path.

Obstacle	Minimum distance to obstacle[m]	Safety distance[m]
1	2.32	2
2	2.89	3
3	6.19	6

Table 5.13: Replan with static and dynamic obstacles results.

The first encounter is resolved by turning the vehicle to the right. In the second encounter with a moving obstacle, the path is replanned to deviate to the left and a waypoint is missed. The path is then conducted to the missed waypoint. Due to the defined turning radius the path only passes near the waypoint and not exactly on it. In Tab. 5.13 it is noted that in the encounter with the second intruder, even though a collision does not occur, the RPA enters the obstacle safety radius. The third obstacle is a static obstacle and the avoidance is made in the direction that leads to the minimum deviation from the original path.

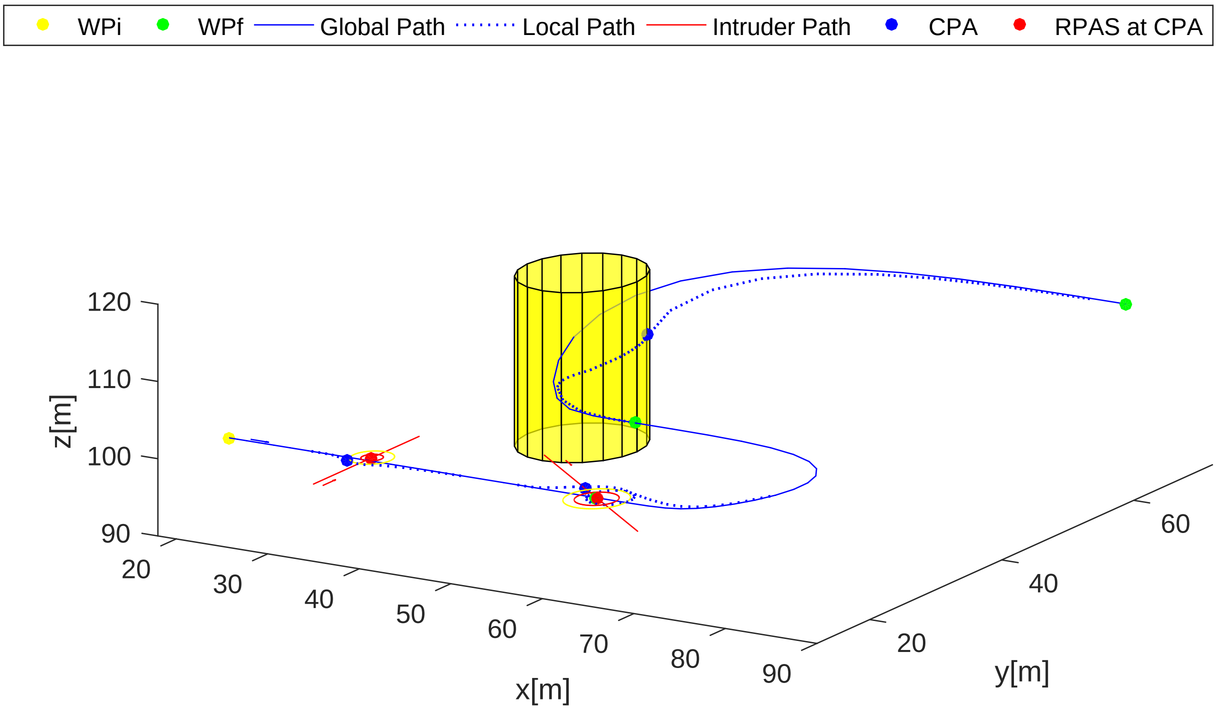


Figure 5.21: 3D view of path.

5.3 Final remarks

From the presented results some conclusions can be drawn. The offline planner provides the best results when A* is used to plan between waypoints and ACO used to optimize waypoint order. There is no guarantee of optimality when using the energy cost function. The optimization of the Bezier curve results in the minimum cost paths, but it provides litle improvement over the curve that is only computed to satisfy constraints at a cost of increased computational time.

In the online stage the algorithm can successfully avoid uncooperative moving and static obstacles. In some situations the safety distance is not maintained because th relative velocity between RPAS and intruder is only considered during the computation of the CPA, and during the maneuver execution the safety distance may be breached. This issue can be solved by either altering the RPAS velocity, increasing it to its maximum or reducing it to the minimum, during the replanned path segment, or increasing the safety radius. When returning to a missed waypoint, the turning radius can prevent the RPAS from exactly passing on the waypoint. The computation time of the online stage is fast enough to be suited for real-time implementation. The algorithm does not require expensive computations and the only data that needs to be stored is the path computed during the pre-flight stage, the obstacles information and the replanned segment. There are many choices of on-board computers that can be used in small platforms. Considering the results obtained for the processing time with a 2.5 GHz machine, using for instance a 800 MHz processor would approximately triple the computation time, this would still result in a processing time in the order of the centiseconds.

However, to determine the processing speed and load of the algorithm for real-time use, the algorithm must be implemented and tested on the hardware platform.

Chapter 6

Conclusions

This work was developed with the aim of investigating and implementing methods that can provide autonomous flight capabilities to RPAS. The focus of this thesis was the development of path planning algorithms with collision avoidance capabilities, that could be applied to both fixed-wing and multirotor platforms to generate safe and optimal paths. The developed system requires the vehicle to have the ability of following a reference path. The task was divided into a global and a local layer, so that when executing the planned mission new obstacles can be avoided without integrally re-designing the initial path.

A pre-flight path planning algorithm was developed using the A* and ACO algorithms. For the planning of the initial coarse path, a set of reference waypoints which the RPA must visit are defined. The planner also has available an obstacle map of the environment, so that the online path only has to be replanned if new obstacles appear, reducing its computational burden, as well as some kinematic constraints of the platform. With this information, a path is planned through a graph like structure with respect to a given cost function, that attempts to minimize either distance or energy expenditure. The resulting path is then used to produce a Bezier curve which is optimized to comply with the vehicle safety and curvature constraints. From the two considered graph search methods, A* proved to be superior to ACO when planning a path between two waypoints, however if a series of reference waypoints are given with no specific order ACO is able to find the optimal sequence. For the global path planning stage, when given a set of unordered waypoints to visit, the best results were obtained by using ACO to optimize the waypoint order and A* to find a path between them when obstacles are present in the way. The global planner can resolve a series of different scenarios, and new optimization criteria can be easily added to expand the range of problems to solve.

For the online stage, Potential Fields are used to generate a local trajectory when unknown obstacles are detected. The replanning of the path is made considering an uncooperative situation between the vehicles, and a sequential resolution of encounters, prioritized according to time to collision. This method balances the following of the reference path and the avoidance of the obstacles. The avoidance maneuver is determined considering the Rules of the Air when dealing with moving obstacles. This is done by evaluating the type of encounter with the intruder to determine if it is a head-on, converging or climb/descent scenario. Depending on the situation, the type of maneuver is set by imposing a distinct swirling direction for each case. If the new local path leads to the avoidance of any of the reference waypoints given during the mission planning stage, the potential flow is adjusted to go to the reference waypoint before returning to the global path. The algorithm was developed to provide a path to a waypoint manager that guides the RPA through the given list, but it can easily be adapted to be integrated with the lower level control modules serving as a navigation system for a reference motion. The online solution performed well on relatively simple scenarios but, to ensure that a secure distance is always maintained, the safety radius must be appropriately defined. The potential fields method is fast and computationally inexpensive, being a feasible solution for real-time implementation.

Overall the developed system provided a satisfactory solution to the path planning problem of small RPAS.

6.1 Achievements

A new approach is presented to solve a constrained optimization problem based on adjusting the weights attributed to the fixed control points of a rational Bezier curve. Two techniques were tested: adjusting the weights to minimize a given cost function subject to bounds and constraints, and adjusting the weights to only meet the imposed constraints on safety distance and curvature radius. As expected, the minimum cost paths are obtained when the cost function is minimized, but the improvements over the latter case were not relevant when considering the significant increase in computational time and the small reduction in the cost function value.

6.2 Future Work

To tackle increasingly complex scenarios some issues need to be addressed. A better integration of the vehicle dynamics is important to improve the system reliability and performance. A complete solution should consider a cooperative scenario where the vehicles exchange flight plans among each other. The type of sensors used must also be taken into account, as they are a crucial part of the real-world implementation, that can significantly affect the performance of the system. Different obstacle configurations must be incorporated to encompass the diversity found in the real word.

The developed solution was created with the intention of increasing the vehicle autonomy, but the interaction with the human operator should be considered, and the pilot's experience should be accounted for the development of better decision systems.

Bibliography

[1] J. Herzog. German Luftwaffe 99-01 Northrop Grumman/Cassidian RQ-4B EuroHawk mock-up at ILA berlin air show, 2012. Retrieved from ”`https://upload.wikimedia.org/wikipedia/commons/thumb/d/d0/Luftwaffe_99-01_RQ-4B_EuroHawk_ILA_2012_1.jpg/800px-Luftwaffe_99-01_RQ-4B_EuroHawk_ILA_2012_1.jpg`”.

[2] Bayhaluk. Bayraktar mini UAV, 2009. Retrieved from ”`https://upload.wikimedia.org/wikipedia/commons/thumb/a/af/ISTANBUL_DHA_GOZCU1.jpg/800px-ISTANBUL_DHA_GOZCU1.jpg`”.

[3] ICAO. Unmanned Aircraft Systems (UAS). Circular 328-AN/190, 2011. ISBN:789292317515.

[4] ANAC. Condições de operação aplicáveis á utilização do espaço aéreo pelos sistemas de aeronaves civis pilotadas remotamente (drones). Diário da República, série 2 — N° 238, Parte E, Dec. 2016.

[5] Amazon. Revising the airspace model for the safe integration of small Unmanned Aircraft Systems., July 2015. Retrieved from ”`https://www.amazon.com/b?node=8037720011`”.

[6] NASA. Unmanned Aircraft System (UAS) Traffic Management (UTM), September 2016. Retrieved from ”`https://utm.arc.nasa.gov/`”.

[7] EASA. Concept of operations for drones - A risk based approach to regulation of Unmanned Aircraft. EASA brochure, May 2015. Retrieved from ”`https://utm.arc.nasa.gov/`”.

[8] K. G. Susan Frost and J. Celaya. A briefing on metrics and risks for autonomous decision-making in aerospace applications. *American Institute of Aeronautics and Astronautics, AIAA 2012-2402*, pages 1–13, 2012.

[9] A. Mujumdar and R. Padhi. Evolving philosophies on autonomous obstacle/collision avoidance of unmanned aerial vehicles. *Journal of Aerospace Computing, Information, and Communication*, 8(2): 17–41, 2011.

[10] H. Alturbeh. *Collision Avoidance Systems for UAS operating in civil airspace.* PhD thesis, Cranfield University, November 2014.

[11] ICAO. Surveillance and Collision Avoidance Systems, Volume IV. Annex 10 to the Convention on International Civil Aviation, 2014.

[12] S. Joint Undertaking. European ATM Master Plan, The roadmap for delivering high performing aviation for europe : edition 2015, 2015. ISBN:9789292160340.

[13] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1): 269–2717, 1959. doi:10.1007/BF01386390.

[14] S. J. Russell and P. Norvig. *Artificial Inteligence: A Modern Approach.* Pearson Education, 3^{rd} edition, 2003. ISBN: 9780136042594.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, 4(2):100–107, 1968.

[16] T. Liao. RPAS collision avoidance using A* algorithm. Master's thesis, Auburn University, May 2012.

[17] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. In *Journal of Artificial Intelligence Research*, pages 533–579, 2010.

[18] L. Filippis, G. Guglieri, and F. Quagliotti. Path planning strategies for UAVs in 3D environments. *Journal of Intelligent and Robotic Systems*, 65(1):247–264, 2012. doi:10.1007/s10846-011-9568-2.

[19] M. Hwangbo, J. Kuffner, and T. Kanade. Efficient two-phase 3D motion planning for small fixed-wing UAVs. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1035–1041, April 2007. doi:10.1007/s10846-011-9568-2.

[20] A. Abdel-Hafez. *Recent Advances in Aircraft Technology.* InTech, dr. ramesh agarwal (ed.) edition, 2012. doi: 10.5772/2406.

[21] S. M. Lavalle. Rapidly-Exploring Random Trees: A new tool for path planning. Technical report, 1998.

[22] Y. Lin and S. Saripalli. Sense and avoid for Unmanned Aerial Vehicles using ADS-B. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6402–6407, May 2015.

[23] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT press, 3^{rd} edition, 2004. ISBN: 9780262042192.

[24] B. Englot and F. Hover. Multi-goal feasible path planning using Ant Colony Optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 2255–2260, May 2011.

[25] M. Brand, M. Masuda, N. Wehner, and X. Yu. Ant Colony Optimization algorithm for robot path planning. In *2010 International Conference On Computer Design and Applications*, pages V3–436–V3–440, June 2010.

[26] Z. Q. Wen and Z. X. Cai. Global path planning approach based on ant colony optimization algorithm. *Journal of Central South University of Technology*, 13(6):707–712, Dec 2006.

[27] Y. Gigras and K. Gupta. Ant colony based path planning algorithm for autonomous robotic vehicles. *International Journal of Artificial Intelligence & Applications*, 3(6):000, Nov 2012.

[28] Y. Z. Cong and S. Ponnambalam. Mobile robot path planning using Ant Colony Optimization. In *2016 2^{nd} IEEE International Symposium on Robotics and Manufacturing Automation*, pages 1–6, Sept. 2016.

[29] G. Ma, H. Duan, S. Liu, G. Ma, H. Duan, and S. Liu. Improved ant colony algorithm for global optimal trajectory planning of UAV under complex environment. *International Journal of Computer Science and Applications*, 4(3):57–68, 2007.

[30] C. Zhang, Z. Zhen, D. Wang, and M. Li. UAV path planning method based on ant colony optimization. In *Control and Decision Conference (CCDC), 2010 Chinese*, pages 3790–3792, May 2010.

[31] E. F. Camacho and C. Bordons. *Model Predictive Control.* Springer, 2^{nd} edition, 2007.

[32] V. Mahalingam and A. Agrawal. Learning agents based intelligent transport and routing systems for autonomous vehicles and their respective vehicle control systems based on model predictive control (MPC). In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 284–290, 2016.

[33] E. Boivin, A. Desbiens, and E. Gagnon. UAV collision avoidance using cooperative predictive control. In *2008 16th Mediterranean Conference on Control and Automation*, pages 682–688, 2008.

[34] Z. Chao, L. Ming, Z. Shaolei, and Z. Wenguang. Collision-free UAV formation flight control based on nonlinear MPC. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 1951–1956, 2011.

[35] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 3, pages 1936–1941, 2002.

[36] K. F. Culligan. Online trajectory planning for UAVs using Mixed Integer Linear Programming. Master's thesis, MIT, 2006.

[37] A. Stalmakou. UAV/UAS path planning for ice management information gathering. Master's thesis, Norwegian University of Science and Technology, 2011.

[38] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1936–1941, May 2002.

[39] S. M. LaValle. *Planning algorithms.* Cambridge University Press, 2^{nd} edition, 2006. ISBN: 9780521862059.

[40] J.Ruchti, R.Senkbeil, J.Carroll, J.Dickinson, J.Holt, and S.Biaz. Unmanned Aerial System collision avoidance using artificial potential fields. *Journal of Aerospace Information Systems*, 11(3):140–144, 2014.

[41] H. Safadi. Local path planning using virtual potential field., April 2007. Retrieved February 20 from `http://www.cs.mcgill.ca/~hsafad/robotics/#_msoanchor_4`".

[42] T. Paul, T. R. Krogstad, and J. T. Gravdahl. Modelling of UAV formation flight using 3D potential field. *Simulation Modelling Practice and TheorY*, 16(9):1453–1462, 2008.

[43] W. B. S. Thrun and D. Fox. *Probabilistic Robotics.* MIT Press, 2005.

[44] J. R. et al. Collision avoidance for unmanned aircraft using markov decision processes. *AIAA Guidance, Navigation, and Control Conference. Toronto, Ontario, Canada.*, 2008.

[45] H. B. et al. Unmanned aircraft collision avoidance using continuous-state POMDPs. In *Robotics: Science and Systems*, pages 1–8, Jun 2012.

[46] C. Lai. ACAS X- the future of airborne collision avoidance. In *Newsletter, EUROCONTROL*, May 2013.

[47] Z. S. Paolo Fiorini. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 1(1):760–772, 1998. doi:10.1007/BF01386390.

[48] Y. I. Jenie, E.-J. van Kampen, C. C. de Visser, and Q. P. Chu. Velocity obstacle method for non-cooperative autonomous collision avoidance system for UAVs. *AIAA Guidance, Navigation, and Control Conference*, 2014.

[49] C. C. d. V. J. E. Yazdi I. Jenie, Erik-Jan Van Kampen and J. M. Hoekstr. Three-dimensional velocity obstacle method for UAV uncoordinated maneuvers. *AIAA Guidance, Navigation, and Control Conference*, pages 1–16, 2016. doi:10.2514/6.2016-1629.

[50] Y. I. Jenie, E.-J. van Kampen, and C. C. de Visser. Selective velocity obstacle method for deconflicting maneuvers applied to Unmanned Aerial Vehicles. *Journal of Guidance, Control, and Dynamics*, 38(6):1140–1146, 2015.

[51] J. van den Berg, J. Snape, S. J. Guy, and D. Manocha. Reciprocal collision avoidance with acceleration-velocity obstacles. *2011 IEEE International Conference on Robotics and Automation*, 5(2):3475–3482, 2011.

[52] F. P. Beer and E. R. J. Jr. *Vector Mechanics for Engineers (Dynamics).* McGraw- Hill, 4^{th} edition, 2006. ISBN: 9780070044388.

[53] T. Baiao. Energy monitoring system for low-cost UAVs. Master's thesis, Instituto Superior Técnico, May 2017.

[54] C. Wilt and W. Ruml. When does weighted A* fail? Proceedings of the Fifth Annual Symposium on Combinatorial Search, 2012.

[55] M. Owen, R. W. Beard, and T. W. McLain. Implementing dubins airplane paths on fixed-wing UAVs. In K. P. Valavanis and G. J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 1677–1701. Springer Netherlands, 2015. ISBN 9789048197071.

[56] E.-M. Kan, M.-H. Lim, S.-P. Yeo, J.-S. Ho, and Z. Shao. Contour based path planning with b-spline trajectory generation for Unmanned Aerial Vehicles (UAVs) over hostile terrain. In *Journal of Intelligent Learning Systems and Applications*, pages 122–133, August 2011.

[57] K. Yang and S. Sukkarieh. 3D smooth path planning for a uav in cluttered natural environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 794–800, Sept. 2008.

[58] F. Hu and S. Wang. An algorithm for path planning of multiple Unmanned Aerial Vehicles based on Bezier curve. In *Proceedings of the* 29^{th} *Chinese Control Conference*, pages 3660–3665, July 2010.

[59] J. A. S. Jayasinghe and A. M. B. G. D. A. Athauda. Smooth trajectory generation algorithm for an Unmanned Aerial Vehicle (UAV) under dynamic constraints: Using a quadratic Bezier curve for collision avoidance. In *2016 Manufacturing Industrial Engineering Symposium (MIES)*, pages 1–6, Oct. 2016.

[60] D. Lee and D. H. Shim. Spline-RRT based optimal path planning of terrain following flights for fixed-wing UAVs. In *2014* 11^{th} *International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 257–261, Nov. 2014.

[61] D. Marsh. *Applied Geometry for Computer Graphics and CAD*. Springer-Verlag London, 2^{nd} edition. ISBN 9781852338015.

[62] J.-W. Choi, R. E. Curry, and G. H. Elkaim. Continuous curvature path generation based on Bezier curves for autonomous vehicles. *IAENG International Journal of Applied Mathematics*, pages 91–101, 2010.

[63] M.S.Floater. Derivatives of rational Bezier curves. In *Computer Aided Geometric Design*, volume 9, pages 161–174, 1992.

[64] G. J. Kennedy and J. E. Hicken. Improved constraint-aggregation methods. *Computer Methods in Applied Mechanics and Engineering*, 289(0):332–354, June 2015.

[65] D. Ruivo. Formation of unmanned vehicles with collision avoidance capabilities. Master's thesis, Instituto Superior Técnico, Nov. 2015.

[66] P. Panyakeow and M. Mesbahi. Decentralized deconfliction algorithms for unicycle UAVs. In *Proceedings of the 2010 American Control Conference*, pages 794–799, 2010. doi: 10.1109/ACC.2010.5530943.

[67] S. Srikanthakumar, C. Liu, and W. H. Chen. Optimization-based safety analysis of obstacle avoidance systems for unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 65(1):219–231, 2012. doi: 10.1007/s10846-011-9586-0.

[68] P. Panyakeow and M. Mesbahi. Deconfliction algorithms for a pair of constant speed unmanned aerial vehicles. *IEEE Transactions on Aerospace and Electronic Systems*, 50(1):456–4761, 2014. doi: 10.1109/TAES.2013.110766.

[69] ICAO. Rules of the Air. Annex 2 to the Convention on International Civil Aviation, 2005.

[70] S. Arumugam and C. Jermaine. Closest-point-of-approach join for moving object histories. In 22^{nd} *International Conference on Data Engineering (ICDE'06)*, pages 86–95, April 2006.