UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Genetic Algorithm Applied in UAVs Path-Planning

Gustavo de Moura Souza

São Carlos – SP

Genetic Algorithm Applied in UAVs Path-Planning

Gustavo de Moura Souza

Advisor: **Prof. Dr. Claudio Fabiano Motta Toledo**

Final Monograph of course conclusion submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Bachelor in Information Systems.
Concentration Area: Computational Systems, Computational Intelligence, Bioinspired Computing

USP – São Carlos
November 2019

Souza, Gustavo de Moura
Genetic Algorithm Applied in UAVs Path-Planning / Gustavo de Moura Souza. - São Carlos - SP, 2019.
70 p.; 29,7 cm.

Advisor: Claudio Fabiano Motta Toledo.
Monograph (Graduação) - Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos - SP, 2019.

1. Genetic Algorithm. 2. Route Planner. 3. Ray Casting. 4. Risk Allocation. 5. Robotic Operating System (ROS). 6. Embedded Critical Systems. I. Toledo, Claudio Fabiano Motta. II. Instituto de Ciências Matemáticas e de Computação (ICMC/USP). III. Título.

à Joana e Nair

ACKNOWLEDGEMENTS

The acknowledgements is a section of the work dedicated to giving thanks to those who helped me build this project and relevantly helped me in some way in my scientific path. The section is presented in Portuguese so those whom I want to read this text are able to understand what I meant. In particular, I thank my father Wilson, my mother Marcia, my brother Lucas, I also thank Veronica and Allan for contributions to this works, Claudio, for being my advisor and Simões, for the inspiration.

O agradecimento é um ato de reconhecimento de uma pessoa que causou algum tipo de impacto naquele que agradece. Impacto maior do que estar sempre comigo e me apoiar nas minhas decisões foi o ato de me amar e deixar-me amar de volta. Agradeço profundamente ao meu pai Wilson e à minha mãe Marcia por toda a base que eles me proporcionaram, pelos incentivos, apoios e união. Agradeço também pelas dicas amigáveis (as vezes nem tanto), mas que me estimularam a seguir o caminho do conhecimento. Agradeço ao meu irmão, pelas dúvidas, suporte e ser um ótimo ouvinte.

Aqueles que contribuíram para que este trabalho acontecesse e que têm meu reconhecimento foram a Veronica, a qual estou colaborando com seu doutorado e o Allan, pelo suporte e revisões do meu trabalho, também agradeço a todos os professores e funcionários do ICMC.

Em especial agradeço ao meu orientador Claudio pelas longas reuniões de planejamento e de solução dos códigos e métodos utilizados nesse trabalho, pela paciência, pela grande quantidade de conhecimentos que me passou e principalmente por aceitar essa empreitada maluca de ser meu orientador.

Uma pessoa digna de um agradecimento especial é o Simões, por sua contínua fonte de inspiração e por me apresentar esse maravilhoso mundo da computação bioinspirada.

Uso esse espaço para reconhecer o grande impacto que essas pessoas tiveram na minha vida científica.

“Você tem que prometer pra mim que nunca vai matar o melhor de todos.”

(Eduardo do Valle Simões)

“それが俺の忍道だ *sorega oreno nindou da!* ”
うずまきナルト *Uzumaki Naruto*

ABSTRACT

DE MOURA S., G. **Genetic Algorithm Applied in UAVs Path-Planning**. 2019. 70 f. Monograph (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

Path-planning is a problem present in the execution of Unmanned Aerial Vehicles (UAVs) missions where it is previously intended to establish a route capable of navigating the UAV between its origin and its destination. Executing a safe flight between two points, the path-planning problem is well studied and present in the literature, containing several proposed solutions that consider the mathematical programming. It is proposed to use a custom Genetic Algorithm (GA) for optimizing a safe, goal-fulfilling route that can perform well in terms of fuel consumption and route smoothing. The genetic algorithm is widely used in mathematical function optimization, proving to be an exciting tool for route planning, where the problem can be modeled as a function, as already present in the literature. A significant factor for safe path-planning that must be included in mathematical programming is obstacle avoidance. A new use for the Ray Casting (RC) algorithm is proposed by applying it to obstacle collision detection. The RC algorithm is widely used in computer graphics, very present in the rendering of movies and games. Comparisons of different implementations of the genetic algorithm are performed considering or not the statistical risk allocation. The system is applied and tested in an embedded environment using the ROS operating system. Ray Casting's applicability in route planning has proven effective, resulting in feasible and often optimal solutions. The combination of the Genetic Algorithm with the Ray Casting technique applied in path-planning is feasible for integration in an embedded environment under critical systems.

Key-words: Genetic Algorithm, Route Planner, Ray Casting, Risk Allocation, Robotic Operating System (ROS), Embedded Critical Systems.

RESUMO

DE MOURA S., G. **Genetic Algorithm Applied in UAVs Path-Planning**. 2019. 70 f. Monograph (Graduação) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O planejamento de rotas é um problema presente na execução de missões de Veículos Aéreos Não Tripulados (VANTs) onde pretende-se previamente estabelecer uma rota capaz de navegar o VANT entre sua origem e seu destino. Constituindo-se em realizar um voo seguro entre dois pontos, o planejamento de rotas é bem estudado e presente na literatura, contendo diversas soluções propostas que consideram a programação matemática. Propõe-se a utilização de um Algoritmo Genético (AG) customizado para a otimização de uma rota segura, que cumpra o objetivo e que consiga ter bom desempenho em termos de consumo de combustível e suavização da rota. O algoritmo genético é amplamente utilizado na otimização de funções matemáticas, mostrando-se uma ferramenta interessante para o planejamento de rotas, onde o problema pode ser modelado como uma função, como já presente na literatura. Um fator importante para o planejamento de rotas seguras e que deve ser incluído na programação matemática é o desvio de obstáculos. É proposto uma nova utilização para o algoritmo Ray Casting aplicando-o na detecção de colisão com obstáculos. Este algoritmo é profundamente utilizado na área de computação gráfica, muito presente na renderização de filmes e jogos. São realizadas comparações de diferentes implementações do algoritmo genético considerando ou não a alocação de risco de maneira estatística. O sistema é aplicado e testado em ambiente embarcado através da utilização do sistema operacional ROS. A aplicabilidade do Ray Casting no planejamento de rotas demonstrou-se efetiva, resultando em soluções factíveis e muitas vezes ótimas. A combinação do algoritmo genético com a técnica de Ray Casting aplicada no planejamento de rota mostra-se viável para integração em ambiente embarcado sob sistemas críticos.

Palavras-chave: Algoritmo Genético, Planejador de Rotas, Ray Casting, Alocação de Risco, Robotic Operating System (ROS), Sistemas Críticos Embarcados.

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS AND ACRONYMS

AG	Algoritmo Genético
alt	altitude
CASA	Civil Aviation Safety Authority
CCPP	Chance-Constraint Path-Planning
DE	Differential Evolution
EASA	European Aviation Safety Agency
FAA	Federal Aviation Administration
FoG	Fellowship of the Game
GA	Genetic Algorithm
Geo-point	geographical point
GPU	graphical processing units
lat	latitude
LCR	Reconfigurable Computation Laboratory
long	longitude
PSO	Particle Swarm Optimization
RC	Ray Casting
RL	reinforcement learning
ROS	Robotic Operating System
RTL	Return to Launch
UAV	Unmanned Aerial Vehicle
UAVs	Unmanned Aerial Vehicles
UTV	Unmanned Terrestrial Vehicle
VANTs	Veículos Aéreos Não Tripulados

LIST OF SYMBOLS

$Geo\omega---$Geographical waypoint

$\omega---$Cartesian waypoint

ω_o---Mission's origin waypoint

ω_d---Mission's destination waypoint

$\Phi---$Area

Φ_b---Bonification Area

Φ_p---Penalization Area

Φ_n---Non Navigable Area

$a---$UAV's acceleration (meters/second^2)

$e---$UAV's angle (degrees)

$\gamma DNA---$Subject's decoded DNA

$\gamma gene---$Individual's decoded gene

x_t---UAV's position on the X axis (meters)

y_t---UAV's position on the Y axis (meters)

v_t---Horizontal UAV's velocity (meters/second)

α_t---Horizontal UAV's angle (direction) (degrees)

$\Gamma()---$Function that decodes a gene

$\varepsilon---$Precision parameter (used on 4.8)

ω_i---Waypoint described by $\gamma gene_i$

$\zeta_{i,j}---$Segment that connects two waypoints: ω_i and ω_j

CONTENTS

Chapter 1

INTRODUCTION

"Who are you?"
—Chapter 5, Advice from a Caterpillar[2]

1.1 Motivation and Contextualization

Unmanned Aerial Vehicle (UAV) is defined as a powered aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry lethal or nonlethal payloads (DEFENSE, 2005). Today, there is broad interest worldwide in the development of better, faster and more efficient UAV systems. Those systems can be used in a series of applications such as field recognisance, agricultural surveying, disaster assistance, law enforcement and others, and they can perform tasks such as pesticides pulverisation, taking pictures and monitoring crowds of people, citing a few (LU *et al.*, 2019).

Significant advancements in the last few years resulted on the improvement of UAV systems, but the use of such systems is still complicated and requires at least some practical and theoretical experience to be able to fly safely. Law definitions regarding the safe use of UAV's are under development in many countries. Some boundaries defined by the Brazilian laws follow the definitions of international organisations such as the Federal Aviation Administration (FAA), United States; the Civil Aviation Safety Authority (CASA), Australia and the European Aviation Safety Agency (EASA), Europe Union (CIVIL, 2017). These rules define a safe flight and how to achieve it, regarding the people's safety and environment's.

Historically, automation has been a great tool to mitigate problems, and it becomes easy the use of complex systems such as UAVs. An autonomous UAV system can fly safer and execute tasks quicker and preciser than human pilots without proper training could. Many fields of applications require the use of UAV where the use of human pilots can be either expensive or impracticable, presenting the need for autonomous UAV systems.

Autonomous control systems use techniques from the field of Artificial Intelligence (AI) to achieve autonomy (CHEN; WANG; LI, 2009). This research project is part of a bigger project from the ICMC-USP's Reconfigurable Computation Laboratory (LCR) that aims the

[2] All those crazy, inspirational phrases presented on the beginning of each chapter are from the same book, it shall be mentioned where in the book it is from, since they all are from *Lewis Carroll's Alice in Wonderland*.

development of a higher degree of UAV automation using AI tools. The automation can be divided into three main categories: (i) *not autonomous* - the UAV system needs to be operated by a human and does not perform any movement actions by itself; (ii) *semi-autonomous* - the systems have some elements that facilitate the flight, such as stabilisers, autopilots and previously coded actions (e.g. return to land); (iii) *autonomous* - the system can complete a given mission without human interference. The solution the LCR team is developing tries to make the UAV system get to the third level of automation. Nevertheless, still it has not been accomplished entirely, the systems developed so far (such as the work of Arantes (2016) and Arantes (2017), both from LCR) are improving the state-of-the-art and setting new benchmarks, with new systems being currently under development.

1.2 Goals

In a very uncertain environment the variables to describe it are complex, to automate the mission completion is a task that has been approached for a long time. The proposed system intends to simplify the process of flying a UAV. This research particularly investigates a couple of ways to address the path planning problem with risk allocation and obstacle avoidance, while taking care of optimising the fuel consumption and other resources.

Thus, the main goal of this project is to develop a path planning and re-planning algorithm for non-convex environments. This is obtained by applying an evolutionary computation technique called Genetic Algorithm (GA). The biology process of evolution inspires the GA being applied to many optimisation problems. In the present work, GA is combined with Ray Casting (RC) algorithm (BALACHANDRAN *et al.*, 2017) for obstacle avoidance. RC allows solving problems in 3D computer graphics and computational geometry using ray–surface intersection tests.

It is developed a Robotic Operating System (ROS) Service Node that runs the GA planner in a real-world pre-defined map to test and evaluate the performance of the proposed method. This ROS node is integrated on the MOSA system, developed by the ICMC-USP's LCR research group. This module shall be capable of self-generating a route that obeys the environment imposed to it through the previously defined map.

1.3 Work Organization

The present work is divided as follows: Chapter 2 has the main concepts related to GA and the path-planning approached in this research, allowing a better understanding of the work. The following chapter, 3, explains the problem tackled, and Chapter 4 presents the research's methodology for the proposed algorithms. Chapter 5 shows the results obtained during the experiments. The last chapter, 6, reports the final thoughts about the project and conclusions from the results achieved and presents some ideas for future works.

Chapter 2

LITERATURE REVIEW

"'And what is the use of a book,' thought Alice, 'without pictures or conversation?'"
—Chapter 1, Down the Rabbit-Hole

The path planning was described by Li (2010), Blackmore, Ono e Williams (2011) and Ono, Williams e Blackmore (2013) as an integer linear programming problem. In Li (2010), it is proposed a system capable of planning a sequence of discrete actions and continuum controls applied in unmanned aerial and terrestrial vehicles. The planner provides autonomy by giving the system a decision-making module. That work comprehends two innovations: it uses a compact representation for all the feasible planes, being the first to approach continuum and discrete representations at once; and it presents a formulation to define the set of actions a UAV can take.

The work of Blackmore, Ono e Williams (2011) uses chance constraint during the UAV path planning, where the risk of colliding with obstacles must be between a safety margin. Stochastic models describe the path planning problem. The problem becomes non-convex since it includes obstacle avoidance. Hence, the authors propose relaxations and approximations to solve it. The mission planning, otherwise, is not considered by the authors. They approach only the planning of a path between an origin point and a destination point, avoiding obstacles with chance constraints.

Ono, Williams e Blackmore (2013) study mission planning where a new planner is proposed and improvements in the chance constraints are introduced. The system incorporates scheduling for the task execution, closely related to the work of Li (2010), but planning now the time when the vehicle might hit a series of mission goals. In that way, the mission can be described by a set of episodes that occurs between two events. Those episodes specify the goals to be hit by the UAV, and each episode has an associated risk. Two scenarios validate their work: personnel transportation system and aerospace load transportation system. The algorithm applied is based on the Branch and Bound technique.

Evolutionary algorithms have been used for UAV path planning. The work of Patterson *et al.* (2012) used the GA with a Voronoi diagram for the planing of autonomous flight's routes. In this work, a new strategy for the mutation process is introduced, separating global and local diversity. The Voronoi diagram is applied for population generation. The proposed technique improved the quality of the initial individuals within the population, which accelerated the method convergence as a whole.

The work of Tuncer e Yildirim (2012) uses the GA for Unmanned Terrestrial Vehicle (UTV). A new mutation operator is presented and applied for dynamic environment path planning with obstacles. This operator prevents early convergence and can find the ideal path many times. The environment proposed on the work is described as a grid.

Metrics to compare path planners have been proposed by Besada-Portas *et al.* (2013) taking into account the complexity and peculiarity of the problem approached. Those metrics analyse the performance of the method graphically. The authors evaluated three techniques: Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Differential Evolution (DE). The results indicated that the GA was the best method, ahead of PSO and DE.

The Zhang e Duan (2015) applies ED in its method for three-dimensional path planning. The route is projected to be short, and it has a low altitude. The airship must avoid non-navigable regions, radar areas and areas with missiles and anti-aerial weapons. The results were obtained from two different scenarios based on the war environment. The proposed ED showed itself as superior to other methods in terms of robustness and convergence velocity.

An approach to UAV security was the work of Arantes (2016), where the path-planning receives the critical situation approach. This work compares the Genetic Algorithm, the Multi-Population Genetic Algorithm, the Greedy Search and a Mixed Integer Linear Programming to deal with critical situations. The solutions obtained were tested in offline environments. The main goal of this work was to activate the path re-planning, once a UAV reaches a critical state, by conducting it to a safer landing area. The GA and MPGA had similar performance, saving the UAV in 89% of the cases.

The Ray Casting technique used in this research was proposed by Roth (1982) applying the algorithm to CAD solid modelling systems. The complex solids are modelled by combining simpler solids, such as cylinders and pyramids, that receive a virtual light ray cast at its surface. Due to its simplicity in terms of performance and execution time, ray casting is reliable and extensible.

The ray casting technique is widely used on the computer graphics area as mentioned by Kruger e Westermann (2003) where the research goal was to explore an acceleration technique in graphical processing units (GPU) and interpreting the solids as described volume ray casting instead of object-order units. This work influenced the use of the technique to render 3D meshes in movies, games and scientific visualisation software.

In Tarbutton, Kurfess e Tucker (2010), the ray casting technique is applied for machine-tools using the path generation to optimal re-orientation. The work is ground to demonstrate the effectiveness of using graphics hardware on manufacturing problems. With this statement, the base for studying a graphical approach in path planning problems is given.

A path planner for UAV systems using the ray casting technique was presented by a NASA research in Balachandran *et al.* (2017) where it was used to detect and avoid obstacles, geofences and other traffic, taking in account the safety of the mission. The planning algorithm used a rapidly exploring random tree to find a suitable path in the quickest time.

There has been no work found in this research's literature review that combines the GA with ray casting techniques for path planning. More specifically, this work goes further than the combination mentioned above and applies it for UAV systems along with chance constraint embedded in critical systems with Robotic Operating System (ROS).

Given this literature review, the problem addressed by this research uses many of the concept introduced on the literature and tackle a complex problem where many techniques are required to propose a factual solution. The following chapter will introduce and explain the problem details and formulation, while the following sections will develop the work.

Chapter 3

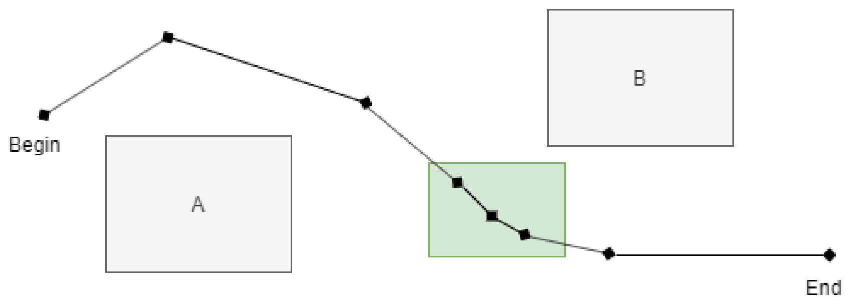
PROBLEM

"Curiouser and curiouser!" cried Alice (she was so much surprised, that for the moment she quite forgot how to speak good English).
—Chapter 2, The Pool of Tears

3.1 Path Planning

When executing a mission, an UAV can wander through many routes. The path planning problem is a way to give this UAV a particular and objective route to follow that respects some constraints imposed by the map definition.

The authors from Blackmore, Ono e Williams (2011) introduce the Chance-Constraint Path-Planning (CCPP) as an stochastic non-convex optimization problem. Stochastic stands for probabilistic problem solving, where many possible alternatives are presented, and each receives a probability for success. In the CCPP, the chance-constraints will model the risk to hit an obstacle or to go through a no-fly zone, when planning a route. The problem is non-convex since the trajectory must avoid obstacles, which become non-convex regions through the path planning map. Figure 1 give us an example.



Source: Author's illustration

Figure 1 – Path Planning problem illustration

Starting on the point *Begin*, the UAV must reach the goal point *End*. During the whole flight, the UAV must stay within the secure regions avoiding collision with obstacles A and B. The probability of the UAV leaving the secure region and collide with obstacles must be at maximum $\Delta = 0.001\%$. This means that the path planning for CCPP assumes that any trajectory has an inherent risk, but the decision-maker can define for it a maximum level of risk.

3.1.1 Problem Description

A mathematical formulation for the chance-constraint path planning is presented on the work of Blackmore, Ono e Williams (2011) as follows.

$$Minimize\ \Theta() \tag{3.1}$$

$$\Theta() = \sum_t \|\overline{u}_t\| \tag{3.2}$$

Where:

$$\overline{x}_T = x_{goal} \tag{3.3}$$

$$\overline{x}_{t+1} = Ax_t + B_{\overline{u}_t} + \mu_t,\ \ \forall(t) \tag{3.4}$$

$$x_0 \sim \mathbb{N}(\overline{x}_0, \sum x_0),\ \ \mu_t \sim \mathbb{N}(0, \sum \mu_t),\ \ \forall(t) \tag{3.5}$$

$$Pr\left[\left(\bigwedge_{j\in\mathbb{I}}\bigwedge_{t\in\Theta\mathbb{I}_j} x_t \in \mathbb{I}_j\right) \wedge \left(\bigwedge_{j\in\mathbb{O}}\bigwedge_{t\in\Theta\mathbb{O}_j} x_t \in \mathbb{O}_j\right)\right] \geq 1-\Delta \tag{3.6}$$

The objective function 3.1 Θ will minimized some metric related to the mission planning such as distance, fuel consumption, collision risk, among others. The authors in Blackmore, Ono e Williams (2011) define the objective function as a cost proportional to the applied controllers' magnitude 3.2.

The restriction 3.3 defines that the last UAV's path's waypoint must coincide with the state x_{goal} that is, the mission's goal point. The state transitions is defined by 3.4, in which the first UAV's state x_0 follows a Gaussian distribution being on the expected initial state $\overline{x}_0$ and a covariance matrix $\sum x_0$. There is also an additive Gaussian noise μ with location zero and covariance matrix $\sum \mu_0$ applied for each state transition.

The constraint 3.6 settling the states x_t must respect the waypoints $(\mathbb{I}_j)$ and avoid the obstacles $(\mathbb{O}_j)$ in every mission state: $t \in \Theta(\mathbb{I}_j)$ e $t \in \Theta(\mathbb{O}_j)$. The expression $Pr[\cdot] \geq 1-\Delta$ stands for the probability of being outside the obstacles, being greater or equal $1-\Delta$, therefore the probability of flaw is less than Δ.

The traveling waypoint and the obstacle's points are defined by convex regions. The traveling waypoints are established as junction of linear restrictions in which the UAV must stay within during the mission time interval $t \in \Theta(\mathbb{I}_j)$. The obstacles are defined as disjunction of linear restrictions in which the UAV must stay outside of them during $t \in \Theta(\mathbb{O}_j)$. The constraints 3.7 and 3.8 describe these stay-in and stay-out situations in which $i \in H_j$ are the hyperplanes on the state space that defines the convex regions for obstacles or traveling waypoints.

$$x_t \in \mathbb{I}_j \Leftrightarrow \bigwedge_{i \in H_j^{\mathbb{I}}} h_i^T x_t \leq g_i \tag{3.7}$$

$$x_t \in \mathbb{O}_j \Leftrightarrow \bigvee_{i \in H_j^{\mathbb{O}}} h_i^T x_t \geq g_i \tag{3.8}$$

3.1.2 Considerations for this Work

At this point, there are some considerations regarding the original path planning problem by Blackmore, Ono e Williams (2011) that must be made to fit this research's scope.

- It is only considered the convex obstacle areas, defined as *non navigable areas* (3.8). It is assumed that the aircraft is flying over a stay-in area with some obstacles on it. Thus, the stay-in situations defined by 3.7 were not considered.
- For the optimization 3.1 the following additional parameters were used: the fuel consumption and the route's curvature. All of them are presented on the GA's fitness function (section 4.2.4).
- Different from Blackmore, Ono e Williams (2011), which defines a risk allocation for each obstacle, we define the same risk for all obstacles as it will be explained in section 4.2.4.

3.2 Obstacle Avoidance

Obstacle avoidance is a hard task to approach due to its irregularity and complex modelling of the problem. There can be defined two ways of avoiding obstacles: reactive and planned. The first considers the environment presented on execution time, by analyzing and interpreting the obstacles that appear after the UAV has started its mission, such as birds, statues, moving objects, and obstacles that may not be previously known. As for the last, the environment considered to compute the route is already defined and immutable, containing obstacles such as constructions, buildings, rivers and fixed objects.

Reactive obstacle avoidance needs a more complex system due to the number of sensors and information needed to perform the task with success. This is not the case of the present project. The planned obstacle avoidance is dealt with here, where the constraints for the problem are already known and mapped. The constraint for this type of object is presented by equation 3.8.

3.3 ROS

The Robotic Operating System (ROS) is an operating system developed particularly for robotic purposes being widely applied to electric cars, drones, planes, robotic arms, submarines and any robotic object that needs a custom operating system. The main task of this research to incorporate the ROS is to make available the algorithms produced for real-world applications, integrating the ROS module developed into a running ROS system that will fly along a rotating wing UAV.

The construction of this ROS service node shall be made available in the context of LCR's operating system and be used for testing, which results are presented in Chapter 5.

3.4 Genetic Algorithm

Evolutionary computation is a wide field of research containing the Genetic Algorithm (GA) as a rich tool to solve many optimization problems. This research aims to apply the genetic algorithm as a way to solve the path-planning problem 3.1.

GA is an intelligent heuristic to optimize a function by cleverly searching the domain space, reducing the number of iterations it is needed to find at least a local optimum. The application on the problem described can be verified in the next chapter where detailed explanations on the development were made.

3.5 Conclusion

Given the problem defined on previous sections, this chapter introduced the main challenge this research is focusing and formalized the concepts and restrictions to be tackled on the solution.

The next chapter uses the concepts defined here to explore and present the work performed to get to a solution.

Chapter 4

METHODOLOGY

"I don't see how he can ever finish, if he doesn't begin."
—Chapter 9, The Mock Turtle's Story

This section presents the methodology followed by this research, introducing the genetic algorithm developed and the techniques used to calculate the UAV collision into non-navigable areas. It is explained how the fitness function is computed and the two versions that incorporate uncertainty.

4.1 Representation of the Problem

4.1.1 Waypoints

The real-world positions of the UAV are given in geographical coordinates that identify a unique location in the world. Since computationally this is not the best approach to perform calculations and execute algorithms, the chosen way to represent the environment is to transform those geographical coordinates to Cartesian coordinates, setting the home point as a chosen geographical point (Geo-point) of reference. The representation of the locations in Cartesian coordinates enables the problem to be approached by computer graphics algorithms, as the Ray Casting algorithm and ease the representation of the UAV controls as vectors.

A *Waypoint* is defined as a point the UAV is supposed to pass over when executing a mission. The waypoints can be represented as Cartesian (ω) by having X, Y and Z coordinates with respect to a chosen system origin point (point (0,0,0)), or Geographical $(Geo\omega)$ with latitude (lat), longitude (long) and altitude (alt) values related to the world's measurement system.

4.1.2 Areas

The areas (Φ) are regions of the map that can represent either penalization, bonification or non-navigable areas. All of those are pre-defined regions that represent how good the space is for the UAV to fly over or land.

The bonification areas are regions that are good for the UAVs to fly and to land on. These are specially designed locations that, if some failure occurs on the vehicle, bonus the system when finding an emergency path to land are selected as the last waypoint. A good bonification area, for example, is a football field, with a clear vision and big open spaces.

The penalization areas are regions that are not good for landing. These regions can be flight over, yet the landing is not allowed. The routes that land in one of those regions are penalized. One example of penalization areas is a parking lot, the flight is allowed but it is hard or insecure for landing.

The non-navigable areas are strictly obstacle definitions. They represent regions that cannot be flight over and the UAV cannot land. These regions also penalize the algorithm that lands inside and that pass over. These can be considered as buildings, big trees and monuments, for example.

Since the incorporation of bonification and penalization areas implies significant modifications on the fitness function, those were not considered for the GA planner. However, they are constraints that can make the system more robust.

For this research implementation, the non-navigable areas are the only type used, and they are treated as obstacles on the field with maximum altitude, i.e. a UAV cannot fly over the obstacle, even if it is far above the limits of the obstacle.

The system is built to manipulate the data on the Cartesian system. Although, the real-world data are provided on the Geographical system. In order to use the appropriate type, Geo-points ($Geo\omega$) are converted to Cartesian-points (ω).

In order to convert to Cartesian coordinates, a system's origin point (0,0,0) must be defined. The Geo-point defined as the *home* point, will consider the Cartesian's origin point for calculations. $Geo\omega_{home}$ contains the latitude, longitude and altitude of the chosen home point, hence the ω_{home} contains the X, Y and Z equal to 0, defining the system's origin. It makes the need to disclose the Cartesian system's origin and the mission's origin. The mission's origin is the place where the UAV starts the mission and can have any values for ω.

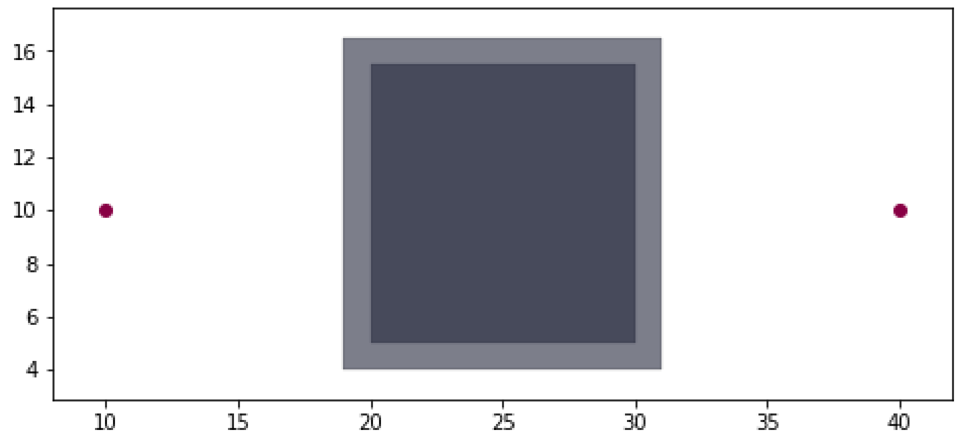
$$\begin{aligned} \omega_t^x &= (Geo\omega_t^{long} - Geo\omega_{home}^{long}) \cdot (6.4 \cdot 10^6 \cdot \cos((Geo\omega_{home}^{lat} \cdot \frac{\pi}{180}) \cdot \frac{2\pi}{360})) \\ [!htb]\omega_t^y &= (Geo\omega_t^{lat} - Geo\omega_{home}^{lat}) \cdot \frac{10^7}{90} \\ \omega_t^z &= Geo\omega_t^{alt} \end{aligned} \tag{4.1}$$

$$\begin{aligned} Geo\omega_t^{lat} &= \frac{\omega_t^y \cdot 90}{10^7} + Geo\omega_{home}^{lat} \\ [!htb]Geo\omega_t^{long} &= \frac{\omega_t^x \cdot 90}{10008000 \cdot \cos(Geo\omega_{home}^{lat} \cdot \frac{\pi}{180})} + Geo\omega_{home}^{long} \\ Geo\omega_t^{alt} &= \omega_t^z \end{aligned} \tag{4.2}$$

The equations 4.1 convert a Geo-point to Cartesian point. Therefore, the inverse way is also necessary in order to feed the autopilot $Geo\omega$ of the aircraft. The equations 4.2 perform the conversion from Cartesian to Geographical points.

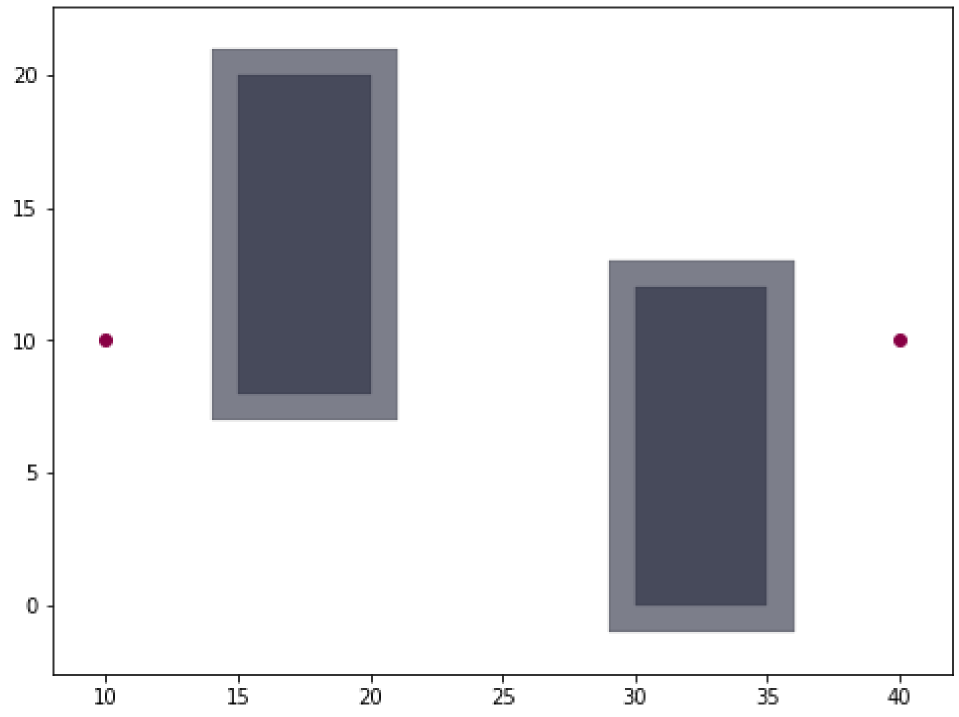
4.1.3 Maps and Risk

The performance of the proposed path planner will be evaluated over the three maps illustrated by Figures 2,3 and 4.



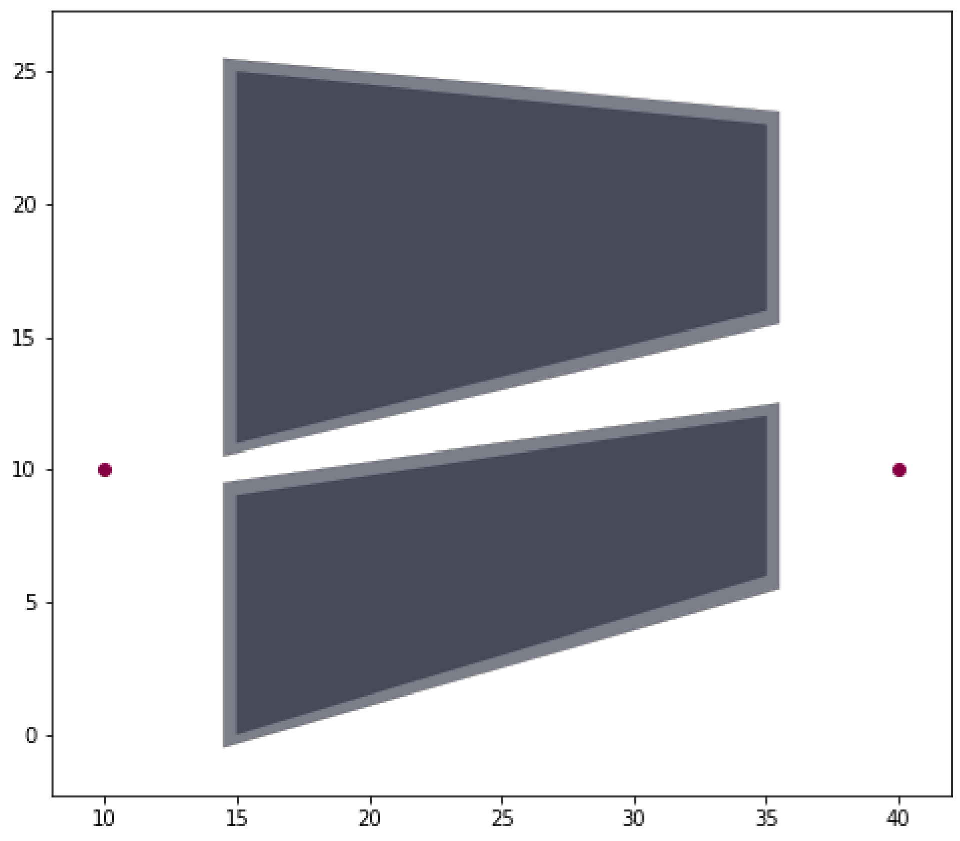
Source: Provided by Author - Code execution

Figure 2 – Visualization of Map 1. The red dot on the left is the ω_o and the right is the ω_d. The darker square is the original Φ_n and the light gray region is the inflated zone.



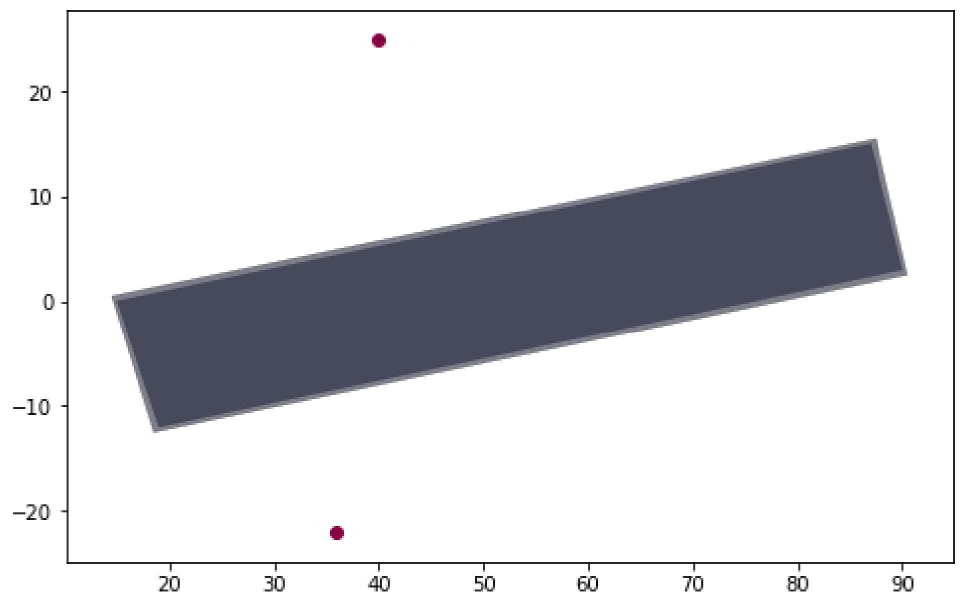
Source: Provided by Author - Code execution

Figure 3 – Visualization of Map 2, with two Φ_ns.



Source: Provided by Author - Code execution

Figure 4 – Visualization of Map 3, with harder Φ_n.



Source: Provided by Author - Code execution

Figure 5 – Visualization of Map 4, reproducing a real map presented on figure 19

When defining the waypoints for the UAV to go through, the aircraft is subjected to some uncertainty from the GPS (μ). This uncertainty has been modelled in previous works as an integer linear programming (ARANTES, 2016) (ARANTES, 2017). To the boundaries of this research, the uncertainty is not directly expressed on the mathematical formulation, instead, it is dealt with risk allocation and relaxation of the destination precision.

The risk allocation is applied to the pre-processing of the non-navigable areas by inflating the edges to increase the area. Two ways of inflating the areas were developed. The first adds a fixed amount and move the edge outwards by this amount. This approach is sensitive to the size of the areas. For example, if the map contains small and big areas at the same time, the uncertainty expressed will be very different. Although, to comply this issue a second approach was developed, where to take into account the size of the original area, the inflation is applied by expanding each edge by a percentage of the original size. In this way, every area inflates in a more regular way.

The uncertainty is also implied on the precision of the route's last waypoint target hit ω_T by adding relaxation to the fitness calculation. It is set a constant ε indicating the precision given in meters that the UAV must be within. Further details are explained on fitness function 4.8.

4.1.4 Data Definitions

The first contribution of this work is to provide the LCR's system data definitions. This process comprehended the formal definition and design of data structures for the objects used in a mission.

An object class for the mission itself was created to standardize the system's function callings. A *mission* contains a reference for a *map* and a set of *instructions*. The instructions can vary depending on the subsystem used, yet, some default instructions are *navigate*, *Return to Launch (RTL)* and *fill*. The RTL instruction calls the autopilot's RTL method and the fill instruction starts another algorithm to perform a sweep over a defined area.

The instruction that interests this research is the *navigate*. Its goal is to fly from an origin point and arrive and land at a destination point. This instruction contains the definition of the waypoints data structure for origin, expressed as ω_o, and a waypoint for the destination, expressed as ω_d. A mission is considered to have only one instruction, which is a *navigate*, therefore, the instruction's waypoints are referred to as the mission's waypoints.

Another *mission*'s element is a *map*, which contains a set of areas, noted as Φ. A non navigable area is noted as (Φ_n). A *map* also contains the chosen $Geo\omega_{home}$.

The described architecture was implemented on Python [1] classes and used JSON files for storage. The use of Python was selected due to its capability of running under the ROS system and facilitate the GA implementation. The choice of using JSON files for storage is due to its readability and easiness to be used by different programming languages since the ROS system

[1] Code available on author's GitHub page: https://github.com/gustavo-moura/path-planning (last accessed, before publishing, on November 30, 2019)

contains nodes written in C++ and Python, and the LCR's system uses even more languages for backend and frontend that needs to communicate between those subsystems.

To standardize the classes and methods it was created a definition for the data using the file type JSON. The use of JSON files optimizes the process of inserting data for the mission, UAV hardware and areas definition.

4.2 Genetic Algorithm

The Genetic Algorithm is inspired by the biological way of evolving as discussed by Darwin in his book *"On the origin of species"* (DARWIN, 2004). The algorithm is an abstraction of the evolution process of species and it works by iterating over many generations, selecting the best individuals to breed and propagates its genes.

Following the *natural selection* theories, the fittest individuals stay alive longer and generate new offsprings that are sometimes better than even the best of them all. The best of them all is the individual with the best fitness value, i.e. the individual who is most adapted to live in the conditions provided by the environment.

In this project, the best individual is abstracted as the best route that solves 3.1. The GA proposed here receives as input an object that defines the mission the UAV must complete. The object contains one or more non navigable areas (Φ_{ni}), an origin (ω_o) and a destination (ω_d) waypoint, all defined in the Cartesian system. The output is the Cartesian waypoints of the final route selected.

The genetic algorithm performs an optimization on the fitness function (4.7) which is the problem's objective function (3.1) that needs to be minimized. The pseudo-code 1 presents the algorithm implemented in this research.

4.2.1 Versions

The process of evaluating the method consisted of testing and comparing two different versions of the GA. The variations consisted of applying the ray casting algorithm for object collision detection and optimizing or not the DNA's size, i.e. optimize T. The naming convention and detailed explanation on each version are described:

- **Version** α := Risk allocation using Ray Casting algorithm, T parameter static, set previously;

- **Version** β := Risk allocation using Ray Casting algorithm, T parameter included on the fitness function to be optimized.

Algorithm 1: Genetic

```
   Input: Criteria: stop_criteria; Int: crossover_rate, population_size; Map: map
   Output: Route: best_route
 1 population ← Genesis()
 2 Initialize_Routes(population)
 3 Evaluate_Fitness(population, map)
 4 while not stop_criteria do
 5     while not converge do
 6         for i=1 to crossover_rate x population_size do
 7             parents ← Tournament(population)
 8             offspring ← Crossover(parents)
 9             Mutation(offspring)
10             Evaluate_Fitness(offspring)
11             Insert(offspring)
12     new_population ← Restart()
13 return best_route
```

It is important to note that the GA's core is the same for both versions with minimal adaptations to include the T parameter optimization, to enable comparisons between the two models.

4.2.2 Individual

The encoding of the GA has the controllers that a UAV needs to perform a route. Thus, the individual's DNA (encoding) is described as a set of values that changes a UAV's position. The decoding of such DNA returns a set of waypoints along with the velocity and angle derived from the DNA itself.

The definitions and the functions used to code and decode the individual's DNA is defined as follows. Figure 6 illustrates the individual's DNA.

4.2.2.1 Encoding

Let T be the planning horizon size (quantity of waypoints that is desired to compute), The individual's DNA is defined for $t = 1, \dots T$:

$$\begin{aligned} DNA &= [gene_1, gene_2, \dots gene_T] \\ gene_t &= (a, e) \end{aligned} \tag{4.3}$$

Where:

a := UAV's acceleration (meters/second^2)
e := UAV's angle (degrees)

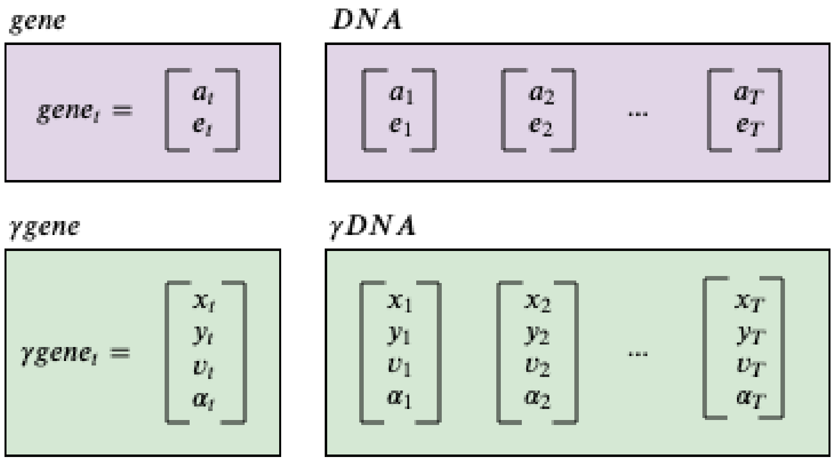

gene
$gene_t = \begin{bmatrix} a_t \\ e_t \end{bmatrix}$
DNA
$\begin{bmatrix} a_1 \\ e_1 \end{bmatrix} \begin{bmatrix} a_2 \\ e_2 \end{bmatrix} \cdots \begin{bmatrix} a_T \\ e_T \end{bmatrix}$
γgene
$\gamma gene_t = \begin{bmatrix} x_t \\ y_t \\ v_t \\ \alpha_t \end{bmatrix}$
γDNA
$\begin{bmatrix} x_1 \\ y_1 \\ v_1 \\ \alpha_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ v_2 \\ \alpha_2 \end{bmatrix} \cdots \begin{bmatrix} x_T \\ y_T \\ v_T \\ \alpha_T \end{bmatrix}$

Figure 6 – Individual's *DNA* and decoded DNA (γDNA)

This representation of individual allows the route to move forward with increasing and decreasing velocity and changing angle freedom, constrained by the UAV design.

4.2.2.2 Decoding

The individual represents the movement in a vectored form. To imply the Cartesian position where this vector leads, it is needed to decode each of the individual's DNA gene. Define γDNA the set of UAV controllers, i.e. the decoded DNA, for $t = 1, \ldots T$:

$$\begin{aligned} \gamma DNA &= [\gamma gene_1, \ldots \gamma gene_T] \\ \gamma gene_t &= (x_t, y_t, v_t, \alpha_t) \end{aligned} \tag{4.4}$$

Where:

x_t := UAV's position on the X axis (meters)
y_t := UAV's position on the Y axis (meters)
v_t := Horizontal UAV's velocity (meters/second)
α_t := Horizontal UAV's angle (direction) (degrees)

To decode each gene and generate the γDNA with $\gamma gene$s it is applied the function 4.5. It transforms the vectors describing the acceleration and angle to a new interpretation, which represents the x and y coordinates, the velocity and the horizontal angle. DNA represents transitions and the decoded DNA represents states. The $\gamma gene_0$ is simply the *origin* waypoint as for the others *genes*, they are decoded by Γ. Let Γ be the function that decodes a gene:

$$\Gamma(gene) = \begin{cases} x_{t+1} & = x_t + v_t \cdot \cos(\alpha_t) \cdot \Delta T + a_t \cdot \cos(\alpha_t) \cdot \frac{\Delta T^2}{2} \\ y_{t+1} & = y_t + v_t \cdot \sin(\alpha_t) \cdot \Delta T + a_t \cdot \sin(\alpha_t) \cdot \frac{\Delta T^2}{2} \\ v_{t+1} & = v_t + a \cdot \Delta T - \frac{F \cdot \Delta T}{m} \\ \alpha_{t+1} & = \alpha + e \cdot \Delta T \end{cases} \tag{4.5}$$

where ΔT is time variation between two consecutive waypoints.

The F term is the drag equation[1], defined by equation 4.6, that is used to better estimate the position and the movement of the UAV on the air by adding friction into consideration. The drag equation allows to determine the strength that an object is subjected to when passing through a fluid[2]. The constants Cd[3], ρ[4] and A[5] are defined according to the UAV's and environment's conditions.

$$F = 0.5 \cdot Cd \cdot \rho \cdot A \cdot v_t^2 \tag{4.6}$$

4.2.3 Selection

The selection process uses the Tournament as defined by Miller, Goldberg *et al.* (1995). It is randomly selected two subjects from the population. The one with the best fitness is selected as the first parent for reproduction by applying the crossover operator. The second parent is selected following the same steps.

4.2.4 Fitness

Each subject from the population has a grade called *fitness*. The fitness represents how much this particular individual is adapted to live in the environment proposed and how much of survival chance it has. In this research it was used a reverse logic: a subject with smaller fitness have a better chance of surviving, therefore, that is why it is a minimization problem.

As the goal is to minimize the function, the penalization is positive. The equation 4.7 describes the function to compute the fitness of the individual.

$$fitness = C_{dist} \cdot f_{dist} + C_{obs} \cdot f_{obs} + C_{curv} \cdot f_{curv} + C_{cons} \cdot f_{cons} + C_T \cdot f_T \tag{4.7}$$

[1] This particular equation was obtained from Arantes (2016, equation 3.5)

[2] In this case, the fluid is the air.

[3] The drag coefficient considered was the same for an *angled cube*, which is 0.8.

[4] The specific fluid mass, a.k.a. density, used was 1.225 given in k/m^3.

[5] Area of reference.

The first term of the function penalizes the distance from the routes last waypoint to the destination waypoint, by calculating the Euclidean distance between those two points (4.8). As part of the uncertainty mitigation, if the distance is less than a precision parameter ε, then the distance is considered zero to not be penalized.

$$f_{dist} = \sqrt{(\omega_d^x - \gamma gene_T^x)^2 + (\omega_d^y - \gamma gene_T^y)^2} \tag{4.8}$$

The second term penalized the collision on non navigable areas. This research applies the ray casting technique to avoid obstacles. Let ω_i be the waypoint described by $(\gamma gene_i^x, \gamma gene_i^y)$. Let $\overline{\omega_i, \omega_j}$ be the segment that connects two waypoints: ω_i and ω_j. The equation 4.9 describes the obstacle fitness function[1]:

$$f_{obs} = \begin{cases} \sum_{t=0}^{T}\sum_{n=0}^{N} RC_P(\omega_t, \Phi_n) + RC_S(\omega_t, \omega_{t+1}, \Phi_n) & \text{, if version} = \alpha \\ \sum_{t=0}^{T}\sum_{n=0}^{N} Pr(\omega_t \in \Phi_n) & \text{, if version} = \beta \end{cases} \tag{4.9}$$

$$RC_P(\omega, \Phi) = \begin{cases} 1 & \text{, if } \omega \in \Phi \\ 0 & \text{, otherwise} \end{cases} \tag{4.10}$$

$$RC_S(\omega_i, \omega_j, \Phi) = \sum_{n=0}^{N} |\overline{\omega_i, \omega_j} \bigcap \Phi_n| \tag{4.11}$$

The equations 4.10 and 4.11 describes the results from the Ray Casting algorithm that are further explained in the next section

Equation 4.12 indicates the third term of the fitness function, this penalizes routes with lots of curves by summing the angle between each segment of the route. Straighter routes can be simplified by transforming three or more collinear waypoints into two, that way, the information passed to the Autopilot is reduced, decreasing the chance of future communication problems with the UAV. Due to that, smoother routes are bonified.

$$f_{curv} = \frac{1}{e_{max}} \cdot \sum_{t=0}^{T} gene_t^e \tag{4.12}$$

The fourth fitness function's term (4.13) is taking in consideration the fuel consumption. To simplify the introduction of fuel consumption on the fitness function, it is interpreted as simply the total route's sum of squared accelerations. Longer the route, bigger the fitness. This forces the algorithm to find the shortest route, saving fuel and completing the mission faster.

$$f_{cons} = \sum_{t=0}^{T} gene_t^{a2} \tag{4.13}$$

[1] When t=T, set t+1=0

The GA's β version developed in this research optimizes the number of waypoints the route has (T) by including on the fitness function the T parameter. This equation (4.14) simply counts the number of genes the individual has. This makes the algorithm to choose routes with fewer waypoints.

$$f_T = T \tag{4.14}$$

4.2.5 Crossover

The crossover is a procedure performed over selected parents to generate a new individual. The process combines the information of these selected parents from the population to generate a new individual, called offspring. Two crossover operators were applied as follows (MILLER; GOLDBERG *et al.*, 1995):

- **OX**: The offspring gene are randomly selected between $parent_1$'s or $parent_2$'s gene.
- **BLX-α**: It is performed a calculation in order to use both the $parent_1$'s and $parent_2$'s gene to generate the offspring's gene. It is taken the medium from each element of the gene and modified by a small value.

4.2.6 Mutation

The mutation step is performed over each of the new genes with a probability $Pr(\text{mutation})$ of mutating of not the gene. There were defined some operators that perform mutation. Each operator is randomly selected with equal chances.

- **Creep**: It is added or subtracted a small random value from each gene's element;
- **Change**: Each gene has a probability of 50% of being replaced by a random generated new gene;
- **Insert**: It is inserted a single random generated gene anywhere on the DNA, if after muting $T \leq T_{max}$;
- **Remove**: A random gene is removed from the DNA, if after muting $T \geq T_{min}$.

The version α only has the *creep* and the *change* operators, in the β version it is included the *insert* and *remove* operators in order to enable the DNA's size changing by mutation.

4.2.7 Extinction

The population that has already converged is extinguished, except for the best subject, where it will perpetuate for the next generation. The new generation with $T-1$ new subjects plus the best subject from the previous generation move on to the next evolution process. For backup reasons, it is saved the "best of all" lineage, always keeping an ancestry history.

4.3 Ray Casting Algorithm

This algorithm is widely used on the computer graphics area, especially for rendering 3D scenes for games and animation movies. It simulates the path a light ray makes when inserted on an environment, by calculating the trajectory of the ray from the light source until it reaches an object. Further developments on the Ray Casting simulated the reflex of objects and the projection of new rays.

Given a 2D polygon and a point, a ray is cast from the point to the infinite. If this ray crosses an even number of the polygon's edge, the point is outside the polygon, if it crossed an odd number, then it is inside the polygon.

For the path-planning problem, the Ray Casting algorithm (2) served to avoid collision with a non-navigable area. By translating the real-world environment to the Cartesian plane, it was possible to use cast horizontal rays from each waypoint of the route. Then, we calculated how many times the ray cast intercepted the polygon formed by the area. If the ray intercepted the area an odd number, the waypoint is inside the area, if it intercepted an even number, then the waypoint is outside of the area.

To avoid the "ray on vertex" problem, the point is moved upward of a small quantity epsilon.

The waypoints are part of the route, but they are not all of it, they are a representation of the instructions given to the UAV. As we want to compute the risk for the whole route, the segment connecting two waypoints of the route must be considered.

The ray casting algorithm can be adapted to perform some calculations and become able to identify if the segment connecting two waypoints is colliding with an obstacle. To perform this calculation, we changed Ray Casting to evaluate the segment Intersection as described by Algorithms 4 and 5.

4.4 ROS Implementation

The ROS System is very useful for constructing UAV's operating systems involving lots of elements and messages passing. The LCR's UAV system is built over the ROS system

Algorithm 2: Ray Casting

```
   Input: Point: p; Polygon: poly
   Output: Boolean
 1 count ← 0
 2 for each side of poly do
 3     vertex1, vertex2 ← get_vertex(side)
                                         // Point A must be below B
 4     if vertex1.y < vertex2.y then
 5         A ← vertex1
 6         B ← vertex2
 7     else
 8         A ← vertex2
 9         B ← vertex1
10     if Ray_Intersects_Segment(p, A, B) then
11         count ← count + 1
12 if count mod 2 == 0 then
                                                               // Odd
13     return True                  // The point is inside the polygon
14 else
15     return False
```

and comprehends more tasks involved such as the use of sensors and actuators. This research's algorithms were also adapted for running in real-life systems.

It was created a ROS node service capable of running the GA algorithm. To construct this node it was necessary to install and evaluate each of the ROS architectural parts. The choice of building a service was made due to the service's characteristics of enabling a request to be solicited and answered when the computation is done. This is particularly useful for the system to not be overloaded by planning solicitations. While waiting for the service's answer the system can perform a series of other tasks.

When involving the implementation for the ROS system, the coordinates received are expressed in geographical format. Because of this other form of receiving the coordinates the bigger adaptation needed for the algorithm to run on the ROS service was the conversion between their geographical point and the Cartesian point. After doing these conversions, the process of optimizing the routes is done exactly as outside ROS.

When the algorithm outputs a solution it is expressed in Cartesian points. For the UAV to be able to fly this inferred route, it is needed to convert back to geographical points and save it in a *.mv* format. The Mavros file format is read by the Mavros autopilot system, which automates the navigation by using waypoints as it's controllers.

Algorithm 3: Ray Intersects Segment

```
Input: Point: P, A, B
Output: Boolean
    // It is added a small constant ε to avoid the point on vertex problem
if $P.y == A.y$ or $P.y == B.y$ then
    P.y ← P.y + $\varepsilon$
// P is upwards or downwards the polygon
if $P.y < A.y$ or $P.y > A.y$ then
    return False
// P is on the right of the polygon
if $P.x > max(A.x, B.x)$ then
    return False
else
    // Ray intersects side
    if $P.x < min(A.x, B.x)$ then
        return True
    else
        // Check if it is a straight line
        if $A.x \neq B.x$ then
            ray1 = (B.y - A.y)/(B.x - A.x)
        else
            ray1 = $\infty$
        if $A.x \neq P.x$ then
            ray2 = (P.y - A.y)/(P.x - A.x)
        else
            ray2 = $\infty$
        if $ray1 >= ray2$ then
            return True
        else
            return False
```

4.5 Conclusion

This chapter presented the methodology approached to solve the path-planning problem, defining and explaining the steps taken to build the GA and the procedures to code the Ray Casting algorithm while obeying the restrictions.

The next chapter will present the results obtained after performing a series of experiments, simulations and real-flights, comparing the two versions developed.

Algorithm 4: Segment Intersects Segment

```
Input: Point: $A_1$,$A_2$,$B_1$,$B_2$
Output: Boolean
/* Function checks if segment $\overline{A_1A_2}$ intersects segment $\overline{B_1B_2}$ */
o1 ← Orientation(A1, A2, B1)
o2 ← Orientation(A1, A2, B2)
o3 ← Orientation(B1, B2, A1)
o4 ← Orientation(B1, B2, A2)
                                                  // General Case
if o1 ≠ o2 and o3 ≠ o4 then
    return True
                                                  // Special Cases
if o1 == Colinear and On_Segment(A1, B1, A2) then
    return True
if o2 == Colinear and On_Segment(A1, B2, A2) then
    return True
if o3 == Colinear and On_Segment(B1, A1, B2) then
    return True
if o4 == Colinear and On_Segment(B1, A2, B2) then
    return True
                                                  // Neither Cases
return False
```

Algorithm 5: On Segment

```
Input: Point: A, B, C
Output: Boolean
/* Function checks if point B is on segment $\overline{AC}$ */
if B.x ≤ max(A.x, C.x) and B.x ≥ min(A.x, C.x) and
B.y ≤ max(A.y, C.y) and B.y ≥ min(A.y, C.y) then
    return True
else
    return False
```

Algorithm 6: Orientation

```
Input: Point: A, B, C
Output: Orientation
$val = (B.y - A.y) \cdot (C.x - B.x) - (B.x - A.x) \cdot (C.y - B.y)$
if val == 0 then
    return Colinear
if val > 0 then
    return Clockwise
else
    return Counter-clockwise
```

Chapter 5

RESULTS

“The best way to explain it is to do it.”
—Chapter 3, A Caucus-Race and a Long Tale

This chapter presents the tests performed to evaluate the proposed method. It is first tested the Ray Casting algorithm performance, to evaluate its use on this problem. Then, it is tested the genetic algorithm’s implementation over a set of defined maps. Following is detailed the GA’s ROS node simulation and real flight test. During the tests, the local minima problem are found and then it is described and explained in the following section.

5.1 Ray Casting performance

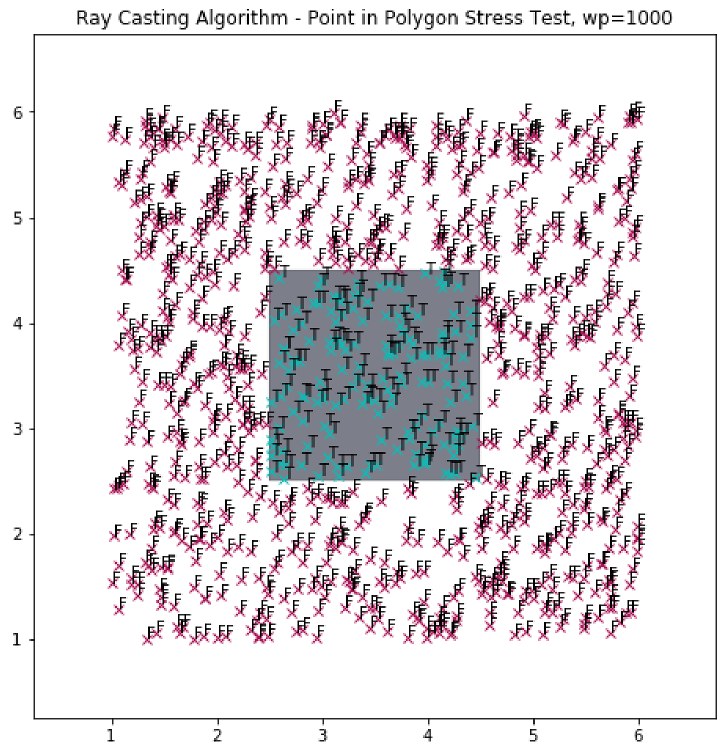
The Ray Casting algorithm used for path planning inside the GA’s fitness function is a novel work being presented. It is considered as the main contribution of this research. To evaluate the performance of the method, it was made a stress test over the algorithms 2 and 4.

To evaluate if a given waypoint is inside a polygon, i.e. inside an area, the algorithm *point in polygon* 2 is used. The process selected a thousand points ω and assigned to them uniformly distributed values for its x and y coordinates, where the equation 5.1 describe the process of selecting the coordinates.

$$\omega_i = \{x, y\},\ \ x \in U[1,6] \ \text{ and } \ y \in U[1,6] \quad \forall i \in [1,1000] \tag{5.1}$$

The generated points were inferred to see if they were inside a defined non-navigable area. Figure 7 shows the results of the algorithm, where each cross represents a waypoint and the square on the centre is the non-navigable area. The label "T" stands for *True* along with the colour green, i.e. the point is **inside** the polygon, and the label "F" stands for *False* along with the colour purple, indicating that the point is **outside** the area. Visually figure 7, this result shows that for every point tested the classification is correct.

The second algorithm needed for the obstacle collision detection considers the intersections between the area’s edges and the segment connecting two waypoints, it is called the



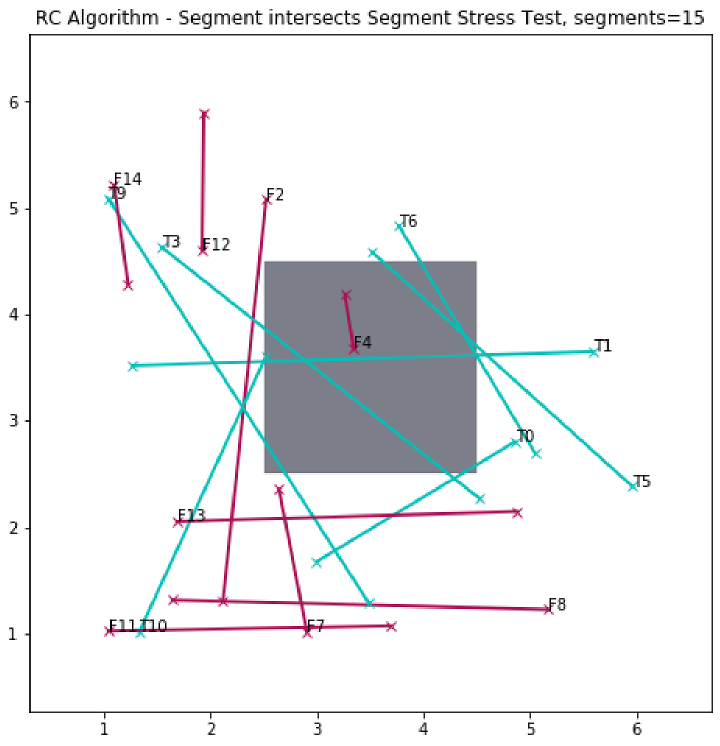
Source: Provided by the author - Code execution

Figure 7 – Stress test for the *point in polygon* Ray Casting Algorithm 2

segment intersects segment Ray Casting algorithm (4). The stress test for this part followed the same steps as the *point in polygon*, although this time the equation is expressed on equation 5.2.

$$\begin{aligned} \overline{\omega_a \omega_{b\,i}} &:= \text{line segment connecting } \ \omega_a \ \text{ with } \ \omega_b \quad \forall i \in [1,15] \\ \omega &= \{x,y\}, \ \ x \in U[1,6] \ \text{ and } \ y \in U[1,6] \end{aligned} \tag{5.2}$$

The results of this execution (Figure 8) shows that the algorithm performs as expected. The green lines represents the classification as *True*, i.e. the segment $\overline{\omega_a \omega_{b\,i}}$ intersects at least one edge of the polygon. The purple lines represent the *False* label. An interesting note about this execution is that the segment F4 (F stands for *False* and the number is an identifier) is **inside** the polygon and does not crosses any edges. It occurs as expected since the algorithm is built to detect intersections only. There is no problem this segment not being penalized by this algorithm because when running the *point in polygon* RC the algorithm will accuse the presence of two waypoints inside the area.



Source: Provided by the author - Code execution

Figure 8 – Stress test for the *segment intersects segment* Ray Casting Algorithm 4

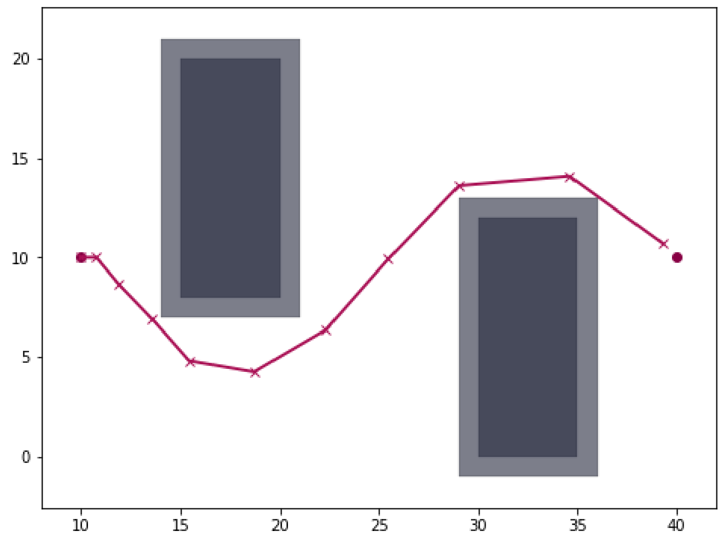
5.2 Evaluation

The parameters for the tests execution are presented on Table 1.

Parameter	Notation	Value
Crossover rate		5
Population Size		10
Mutation Probability		0.7
Destination Cost	C_d	1000
Obstacle Cost	C_{obs}	10000
Consumption Cost	C_{con}	500
Curvature Cost	C_{cur}	100
DNA lenght Cost	C_t	100
DNA genes minimum qty.	T_{min}	1
DNA genes maximum qty.	T_{max}	25

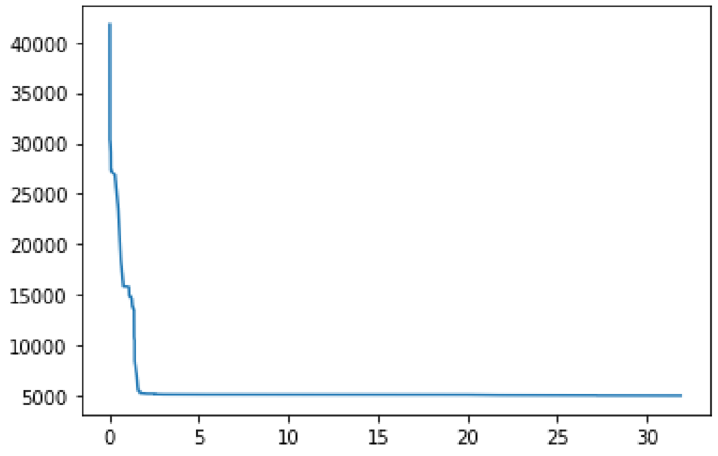
Table 1 – Parameters used on testing

The used methodology for evaluation comprehended the execution of 5 minutes long tests for each map. Inspired by the k-fold validation, it was used 10 different executions of the same algorithm (β version) for the same of the four maps presented. The total testing time took around 200 minutes to complete. The chart in Figure 11 shows the best fitness value through the execution time. On the Y-axis it is represented the fitness value while the X-axis contains the execution time.



Source: Provided by the author - Code execution

Figure 9 – Map 2 solution (β version)

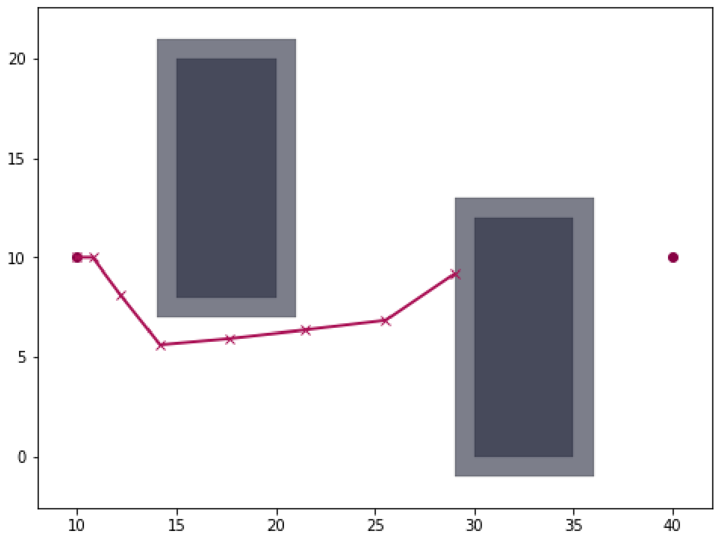


Source: Provided by the author - Code execution

Figure 10 – Best Fitness (β version). Y axis represents the fitness value and X axis the time the individual was found.

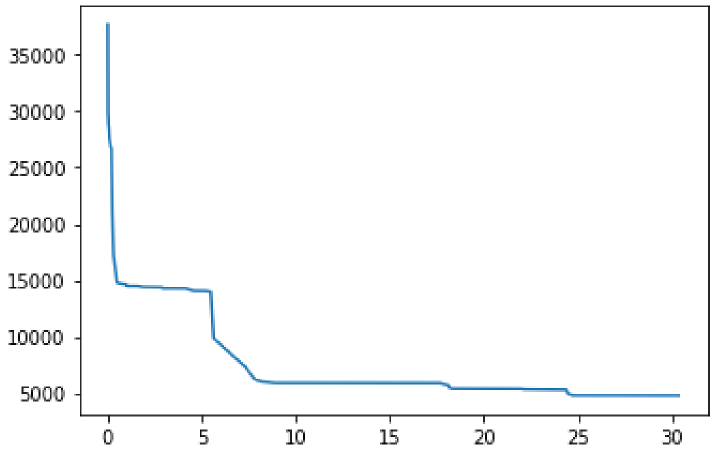
An interesting point to notice is that when the execution took a lot of time, i.e., the algorithm iterated over many generations, the last waypoint of the route ω_T did not hit exactly the destination point ω_d. The ω_T stays the furthest it can, respecting the delimited precision restriction, to minimize the fuel consumption. That happens due to the penalization on longer routes, as getting closer to ω_d would increase the route's length, this solution would have a worse fitness than the solution in which the route is shorter. Hence, the algorithm bonuses shorter routes that consume less fuel and yet arrives at the correct destination, due to the imposed precision. This fact leads us to notice the algorithm's optimization ability and the importance of defining correctly the restrictions.

Another interesting factor to note is that after a short period of execution, the speed of finding the best individual decreases due to the difficulty of finding better solutions.



Source: Provided by the author - Code execution

Figure 11 – Map 2 solution (α version) with $T = 7$



Source: Provided by the author - Code execution

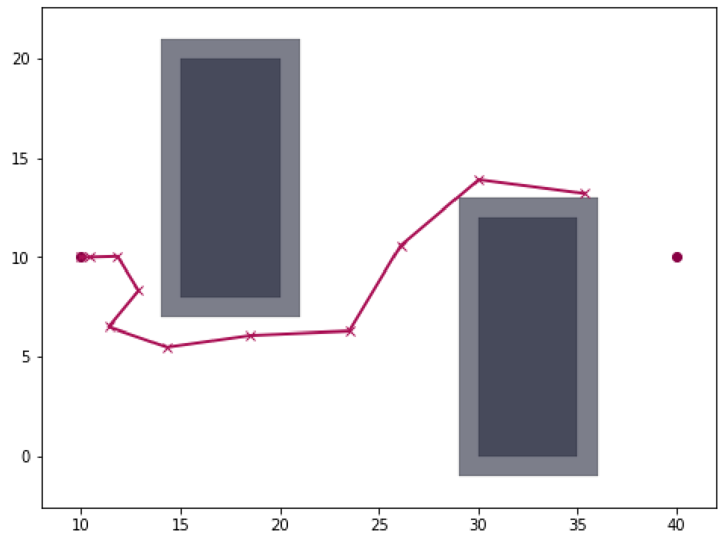
Figure 12 – Best Fitness (α version) with $T = 7$. Y axis represents the fitness value and X axis the time the individual was found.

By running the same parameters (with particular differences specific for the versions) and max_execution_time of 30 seconds, the routes from Figure 11 α $T = 7$ and 9 β differs a lot. The alpha did not get to the destination while the beta version with T being optimized did get.

Since the best route on the beta version, that got to the destination, had 10 waypoints (plus the origin), a second experiment was made running the alpha version now with T=10. The result is presented on Figure 13. It can be seen that the route is a little better, although it still needed more time to find the ideal solution.

This shows us that the parameter T in the alpha version must be fine-tuned by hand, concluding the optimization over T is an important contribution for the method.

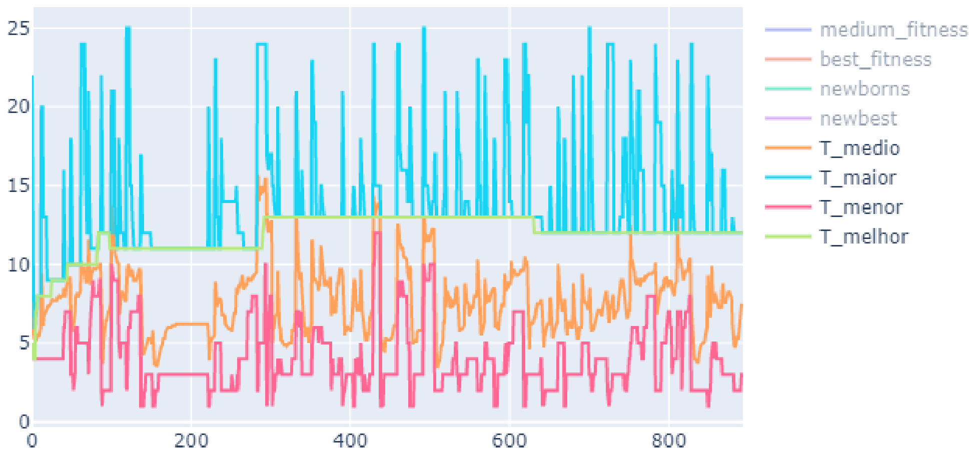
The comparison of the fitness values between different versions (α and β) are not



Source: Provided by the author - Code execution

Figure 13 – Map 2 solution (α version) with $T = 10$

recommended due to the differences of the fitness function, therefore the dimensionality of the results shall vary. Although, it can be compared the dropout rate, in which the beta version (Figure 10) shows a quick convergence while the alpha version (Figure 12) needs more time to find a good solution.

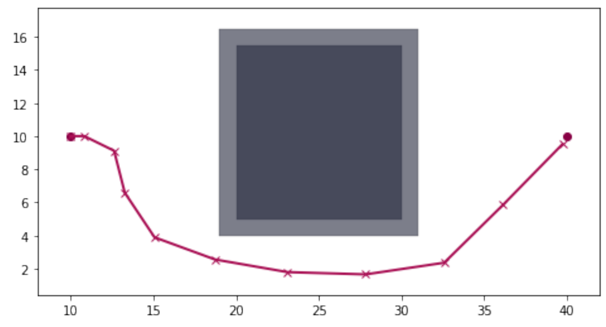


Source: Provided by the author - Code execution

Figure 14 – Map 2 β version, optimization in T. The Y axis shows the amount of T, while the X axis are the different generations

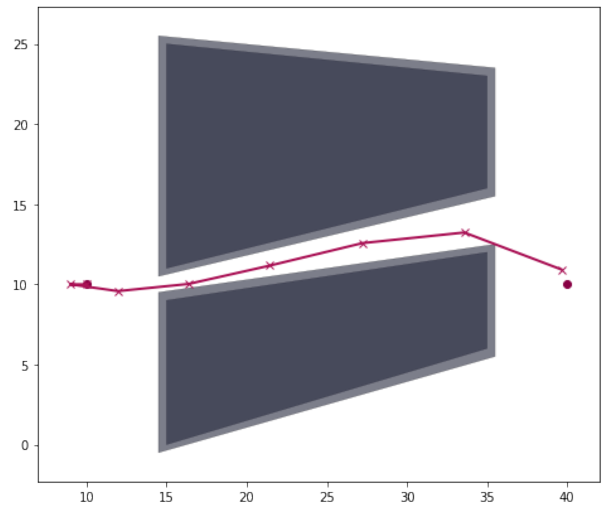
Figure 14 shows the beta version quantity of T over the generations. It is shown the maximum T, the minimum and the medium, considering the whole population, and also is shown the quantity of T for the best. It is interesting that the best quantity of T stays little above the medium and that the convergence pikes occurs when the population is restarted.

The GA was also used to test the maps provided on previous sections. It is important to notice all tests were executed under the same parameters. The figures 15 16 and 17 presents the best route found while executing the beta version for 30 seconds.



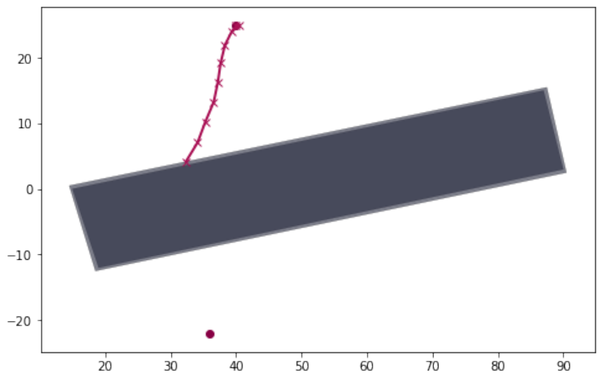
Source: Provided by the author - Code execution

Figure 15 – Best route from Map 1 β version.



Source: Provided by the author - Code execution

Figure 16 – Best route from Map 3 β version.



Source: Provided by the author - Code execution

Figure 17 – Best route from Map 4 β version.

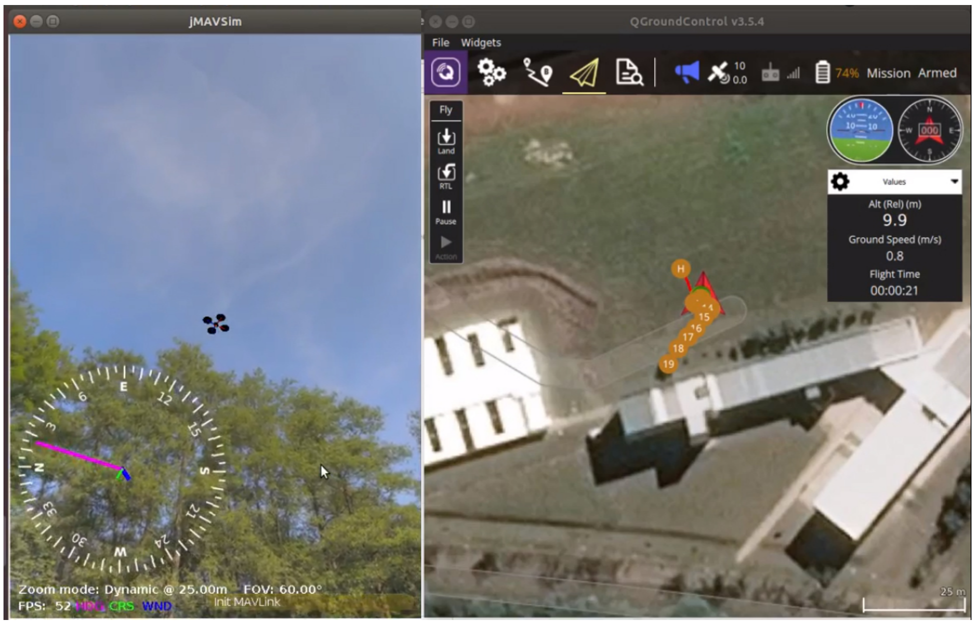
5.3 ROS Simulation

The implemented system was tested on the ROS environment. For the computer simulation execution, it was used the QGroundControl, that allows a satellite view of the mission, and the software jMAVSim that allows the 3D view of the UAV flying and following the route. The GA parameters used were the same as mentioned in Table 1 and only the β version was used.

The GA's ROS node created was inserted on the complete system to execute real-world tests. The system was simulated on a computer and tested on the field. The implemented system was tested on ROS environment, locally. The program QGroundControl was used for the execution of the computer simulation, which enables a satellite vision of the terrain. The software jMAVSim allows the 3D view of the UAV flying and following the route. The GA parameters used were the same as mentioned on Table 1 and only the β version was used, however, the maximum time for execution of the algorithm defined was 30 seconds. This definition was made because, in an embedded environment, the time to execute the algorithm is a critic factor since this occurs while the processing of the route is made by the UAV, therefore finding a path in a short time of execution is fundamental.

For the simulation in a real environment, the UAV ROS system developed by the LCR laboratory was used and executed in the environment described by the real map of campus 2 from USP São Carlos. We can assume that the system had positive results because the mission was completed although the processing exceeded the given time.

The map illustrated on figure 5 is showing the definitions of the real environment where the tests occured.



Source: LCR

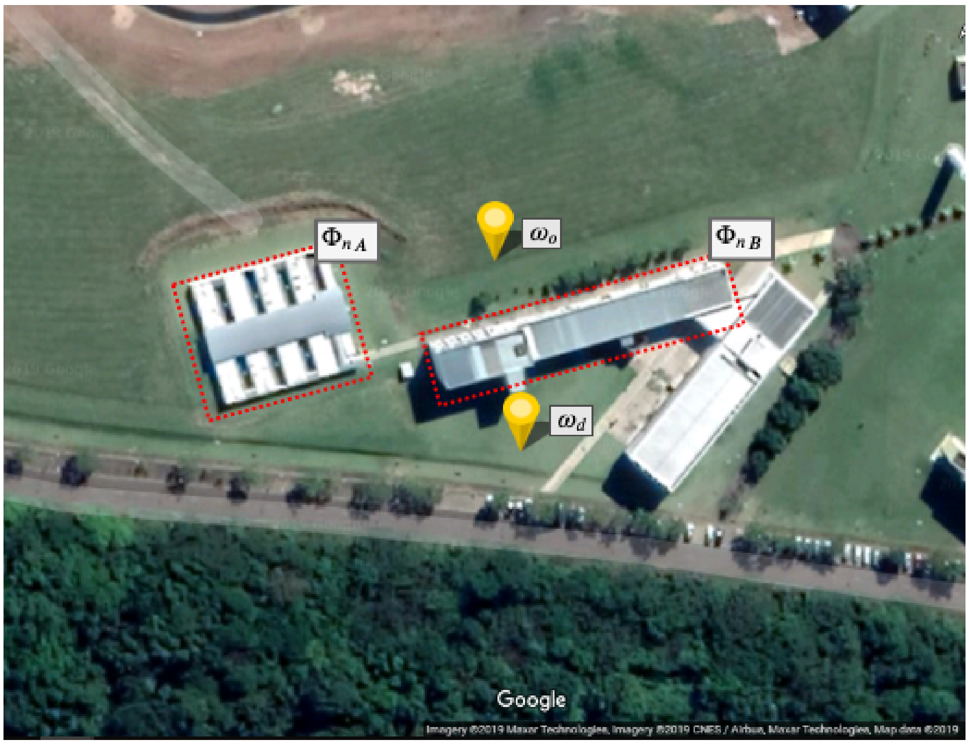
Figure 18 – Screenshot of the simulators. jMAVSim on the left showing the vision of the pilot and QGroundControl on the right showing the generated route execution.

5.4 Real-World Evaluation

Testing on the real world requires open physical space to fly the UAV. The chosen location was the USP São Carlos' Campus 2 ICMC area. The definitions were made previously as defined by 19. The algorithms developed were tested on a real flight. To provide a factual route, the system run over 5 minutes to generate a good solution that could complete the mission successfully.

Many execution and tests were required to empirically choose each parameter. A good set of parameters were defined to compute and execute the system's tests. Table 1 shows the chosen fixed hyperparameters used on the tests.

The final route selected for this map was close to figure's 5 solution since the map 4 were inspired on this map 19. Even though after a long time of execution the algorithm ends up finding a solution that is factual, due to the necessity of running the algorithm in a few seconds, the best route found still is not optimal.


$\Phi_{n\,A}$
ω_o
$\Phi_{n\,B}$
ω_d
Google
Imagery ©2019 Maxar Technologies, Imagery ©2019 CNES / Airbus, Maxar Technologies, Map data ©2019

Source: Author's illustrations over Google Maps' satellite view

Figure 19 – ICMC USP São Carlos Campus 2 satellite view mission environment. In red dashed lines the non navigable areas. The origin and destination waypoints are in yellow marks.

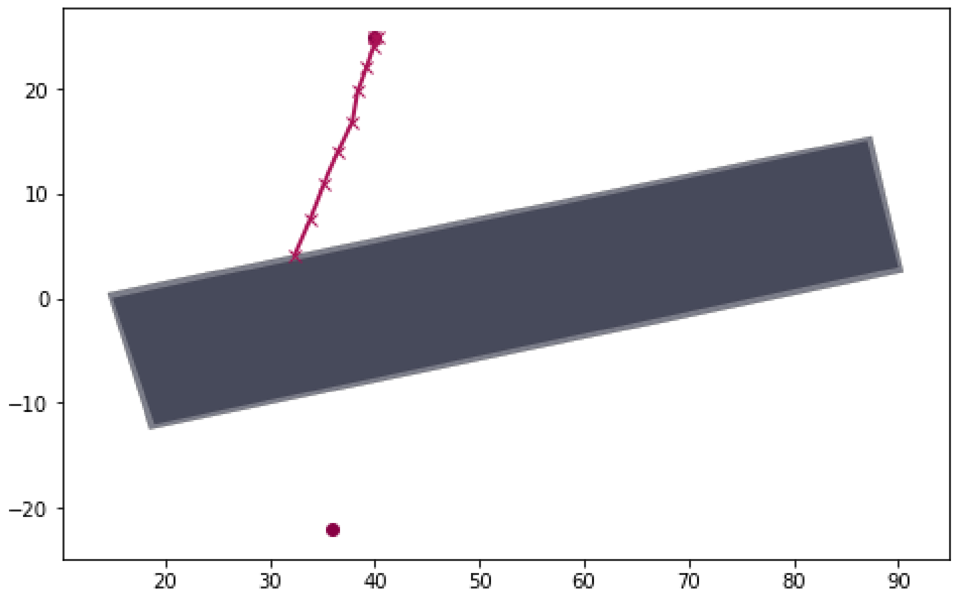
Source: Provided by the author - Picture taken from video

Figure 20 – Real UAV flying a mission

5.5 Local Minima

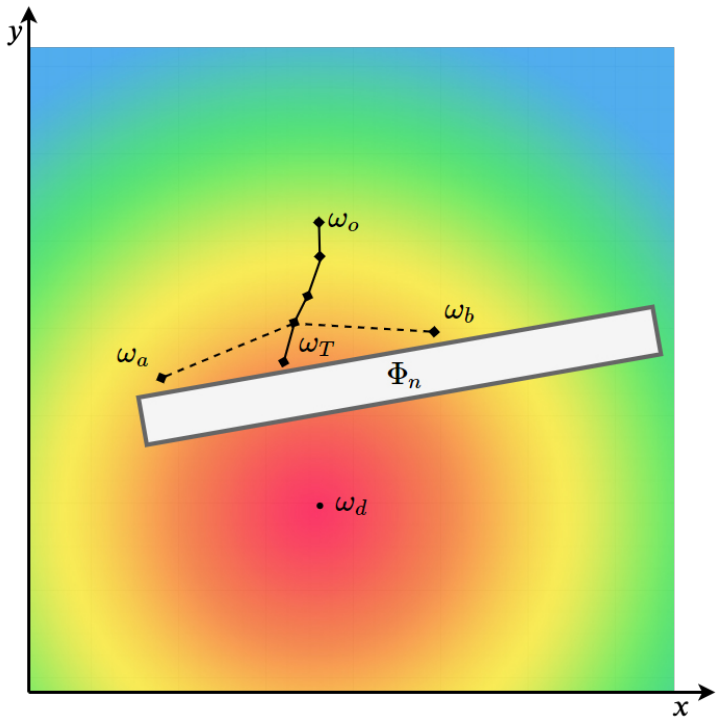
When executing the code for the real world's map that represents USP São Carlos Campus 2 environment, we noticed that the algorithm had a lot of difficulties to optimize and find a factual solution as shown in Figure 21. Even after a lot of time of running (about 10 minutes), the algorithm could not yet get to the destination waypoint, being trapped in the middle of the way by the obstacle. This trap is called local minima.

Local minima are regions of the problem's domain that are minimal only for the close neighbourhood.



Source: Provided by the author - Code execution

Figure 21 – Non factual solution for Map 4 trapped on local minima. $max\_execution\_time = 30s$



Source: Author's illustration

Figure 22 – Color representation of the potential field for f_{dist} 4.8. From red to blue the fitness value increases.

Note that the yellow region has worse distance fitness 4.8 than the red one. That happens because the distance restriction has the same values for every point in the same distance from ω_d. The function can be represented as a potential field, which ends up having a radial form. Even though there is an obstacle between the origin and destination waypoints, the total fitness is trapped in local minima. If it tries to set $\omega_T \leftarrow \omega_a \vee \omega_b$, the fitness will get worse, as shown by 5.3.

$$\begin{aligned} f_{dist}(\omega_T) < f_{dist}(\omega_a) \\ f_{dist}(\omega_T) < f_{dist}(\omega_b) \end{aligned} \tag{5.3}$$

Due to the randomness applied to the generation of new individuals, it is still possible for the algorithm to escape the local minima trap, yet is a hard task to complete successfully.

5.6 Conclusion

This chapter showed the results obtained while executing many instances of the algorithms developed, along with the solution routes for each of the presented maps. It also showed the real-life flight constrains obtained and the analysis of the performance.

Next chapter presents the research conclusions and the possibilities for future work.

Chapter 6

CONCLUSION

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to," said the Cat.
"I don't much care where—" said Alice.
"Then it doesn't matter which way you go," said the Cat.
"—so long as I get somewhere," Alice added as an explanation.
"Oh, you're sure to do that," said the Cat, "if you only walk long enough."
—Chapter 6, Pig and Pepper

6.1 Contributions

One of the contributions this work has provided is the variable quantity of genes, with the T parameter optimization (4.14) inserted on the fitness function (4.7). This allowed the method to better converge and to be less susceptible to hyperparameters fine-tuning. The results sections show the efficacy of the method.

A big contribution to the whole LCR system is the design and formulation of the data definitions to better fit the system structures.

It was also developed the GA in the ROS system. That enabled the algorithm to be executed on a real flight.

The use of the Ray Casting algorithm in path planning problems being optimized by a genetic algorithm is a novel method, as far as this research literature review went.

6.2 Future Work

The development of this work occurred in a brief time, future work to be done is the translation of the 2D world representation to the 3D, by adding one dimension on the subject's DNA definition, hence, adding one more variable to the construction of the vectors defining the UAV's controllers.

The output of the method is the geographical waypoints that can be read by Mavros. Improvements in this part could be changing the output in vector form. This approach could simplify the connection between the nodes for the autopilot by reducing the processing time in converting the waypoints in control movements and giving directly those vectored controllers.

The genetic algorithm has been compared with the reinforcement learning (RL). The main conceptual difference is that RL is an inter-life optimization method, as for the GA is an extra-life optimization, requiring many generations to improve. A possible continuity for this work is to compare the performance of these two approaches.

6.3 Project and Course Connection

Many of the disciplines were important for the success of this research. The main subjects used were: Evolutionary Systems Applied to Robotics (SSC0713 2017/2), Artificial Intelligence (SCC0530 2018/1), Statistical Data Mining (SME0878 2019/1), Algorithms and Data Structures (2016/2). Many of the Bachelor in Information Systems' first and second-year disciplines were essential to the creation and strengthening of my computer basis.

Not only the disciplines contributed to my development but also my participation in extension groups was fundamental to the satisfactory accomplishment that I consider having in this research. From the beginning of my undergraduate course, I have been involved with the Fellowship of the Game (FoG), the ICMC-USP's game development group. My participation in this group taught me interpersonal skills, the called, soft-skills (non-technical). Such skills contributed that I could better position myself before the problems and challenges I experienced in this work. Even though I was a member of the group collaborating with part of the project, I positioned myself so I could offer interesting new insights to improve the whole LCR project.

6.4 Course Considerations

The Bachelor in Information Systems offered by ICMC-USP is a great course that covers the computation fundamentals as well as software development. I consider the ICMC's installation and the physical spaces one of the bests public research institutes. Its classrooms and laboratories are well equipped and the team of employees and professors are one of the bests.

The main point to be improved on the course is the following up of modern technologies and the knowledge that those technologies requires. My first final graduation project was an internship on a software development company, focused on innovation. The database technologies I used there comprehended non-relational databases, although the course we learn about it, it was never the central point nor the modelling and documentation of this type of databases were covered in any mandatory class.

For the course maintain its high standards and be recognized by it is important that the topics covered by the disciplines are covering the state-of-the-art research and emphasizing on what the market is using since a broad part of the students go to the market after finishing the graduation.

Another point I believe is fundamental for the improvement of the course is the specific area the professors are in. Many times during the course, the professor responsible for the discipline were either negligent or the main topic were not his main area. As a suggestion, I recommend the choice of professors for each discipline be made more carefully.

Research groups are organic organizations that grow with not much structure for what I could witness in my time at the university. Many department issues and discussions make professors to not get along with each other causing the birth of new groups for mere political reasons. If the university could hold a strong organization and methodology for its research groups the quality of the work provided and the non-redundant work would increase. I could see a lot of related research being made from different groups that, if joined, could output really strong and impacting research, instead it ends when the researcher finish their degree.

A very strong point for the course being nocturne is the availability to work during the day. For me, I could only do an internship and even employed work while on the course because of the night classes. Despite the work on the market, I consider the "free" day a really strong point, as the student can join groups, practice sports, play some instrument or take extra classes. Those activities are fundamental for the complete development of one’s capability.

I like to say that the strongest part of being in a public university is the availability of a lot of extra courses, extension and culture groups, sports and the academic life outside the classroom. Those extra-class activities are fundamental for the student’s intellectual and technological development growth, as they provide a clear way to improve one’s soft and hard skills, as well as they provide an accepting and comfortable environment to socialize and interact. I strongly recommend the continuity of investments in those areas, particularly for the extension groups, who collaborate to share science with the society and to create new techniques and even areas of knowledge.

BIBLIOGRAPHY

ARANTES, J. da S. **Route planning for UAVs with risk of critical failure: A security-based approach**. Dissertação (Mestrado) — University of Sao Paulo, 2016. Cited 4 times on pages 26, 30, 41, and 45.

ARANTES, M. da S. **Hybrid qualitative state plan problem e o planejamento de missão com VANTs**. Monografia (Doctorade) — Universidade de São Paulo, São Carlos, 2017. Cited 2 times on pages 26 and 41.

BALACHANDRAN, S.; NARKAWICZ, A.; MUÑOZ, C.; CONSIGLIO, M. A path planning algorithm to enable well-clear low altitude uas operation beyond visual line of sight. In: **Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)**. [S.l.: s.n.], 2017. Cited 2 times on pages 26 and 30.

BESADA-PORTAS, E.; TORRE, L. D. L.; MORENO, A.; RISCO-MARTÍN, J. L. On the performance comparison of multi-objective evolutionary uav path planners. **Inf. Sci.**, Elsevier Science Inc., New York, NY, USA, v. 238, p. 111–125, jul 2013. ISSN 0020-0255. Disponível em: <http://dx.doi.org/10.1016/j.ins.2013.02.022>. Cited on page 30.

BLACKMORE, L.; ONO, M.; WILLIAMS, B. C. Chance-constrained optimal path plan- ning with obstacles. **IEEE Press**, 2011. Cited 4 times on pages 29, 33, 34, and 35.

CHEN, H.; WANG, X.-m.; LI, Y. A survey of autonomous control for uav. In: IEEE. **2009 International Conference on Artificial Intelligence and Computational Intelligence**. [S.l.], 2009. v. 2, p. 267–271. Cited on page 25.

CIVIL, A. N. de A. **Regras da ANAC para uso de drones entram em vigor**. [S.l.], 2017. Acesso em: 30/10/2019. Cited on page 25.

DARWIN, C. **On the origin of species, 1859**. [S.l.]: Routledge, 2004. Cited on page 42.

DEFENSE, O. of the Secretary of. **Unmanned Aircraft Systems Roadmap 2005-2030**. 2005. Cited on page 25.

KRUGER, J.; WESTERMANN, R. Acceleration techniques for gpu-based volume rendering. In: IEEE COMPUTER SOCIETY. **Proceedings of the 14th IEEE Visualization 2003 (VIS'03)**. [S.l.], 2003. p. 38. Cited on page 30.

LI, H. X. **Kongming: A Generative Planner for Hybrid Systems with Temporally Extended Goals**. Dissertação (Mestrado) — Massachusetts Institute of Technology, 2010. Cited on page 29.

LU, K.; XIE, J.; WAN, Y.; FU, S. Toward uav-based airborne computing. **IEEE Wireless Communications**, IEEE, 2019. Cited on page 25.

MILLER, B. L.; GOLDBERG, D. E. *et al.* Genetic algorithms, tournament selection, and the effects of noise. **Complex systems**, [Champaign, IL, USA: Complex Systems Publications, Inc., c1987-, v. 9, n. 3, p. 193–212, 1995. Cited 2 times on pages 45 and 47.

ONO, M.; WILLIAMS, B. C.; BLACKMORE, L. Probabilistic planning for continuous dynamic systems under bounded risk. **J. Artif. Int. Res.**, AI Access Foundation, USA, v. 46, n. 1, p. 511–577, jan 2013. ISSN 1076-9757. Disponível em: <http://dl.acm.org/citation.cfm?id=2512538.2512551>. Cited on page 29.

PATTERSON, T.; MCCLEAN, S.; MORROW, P.; PARR, G. Modelling safe landing zone detection options to assist in safety critical {UAV} decision making. **Procedia Computer Science**, v. 10, p. 1146 – 1151, 2012. ISSN 1877-0509. {ANT} 2012 and MobiWIS 2012. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050912005212>. Cited on page 29.

ROTH, S. D. Ray casting for modeling solids. **Computer graphics and image processing**, Elsevier, v. 18, n. 2, p. 109–144, 1982. Cited on page 30.

TARBUTTON, J. A.; KURFESS, T. R.; TUCKER, T. M. Graphics based path planning for multi-axis machine tools. **Computer-Aided Design and Applications**, Taylor & Francis, v. 7, n. 6, p. 835–845, 2010. Cited on page 30.

TUNCER, A.; YILDIRIM, M. Dynamic path planning of mobile robots with improved genetic algorithm. **Comput. Electr. Eng.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 38, n. 6, p. 1564–1572, nov 2012. ISSN 0045-7906. Disponível em: <http://dx.doi.org/10.1016/j.compeleceng.2012.06.016>. Cited on page 30.

ZHANG, X.; DUAN, H. An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning. In: . [S.l.: s.n.], 2015. p. 270–284. Cited on page 30.