CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

Dynamic Path Planning based on Neural Networks for Aerial Inspection

Gabriel Gelard Reis de Castro

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Orientadora:
Milena Faria Pinto

Rio de Janeiro,
Dezembro, 2021

Dynamic Path Planning based on Neural Networks for Aerial Inspection

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em nome do programa, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre.

Gabriel Gelard Reis de Castro

Aprovada por:

Presidente, Milena Faria Pinto, D.Sc. (orientadora)
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)

Diego Barreto Haddad, D.Sc.
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)

Ana Lucia Ferreira de Barros, PhD.
Centro Federal de Educação Tecnológica Celso Suckow da Fonseca (CEFET/RJ)

André Luis Marques Marcato, D.Sc.
Universidade Federal de Juiz de Fora (UFJF)

Raul Queiroz Feitosa, D.Sc.
Pontifícia Universidade Católica do Rio de Janeiro (PUC-RJ)

Rio de Janeiro,
Dezembro, 2021

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

C355 Castro, Gabriel Gelard Reis de
Dynamic path planning based on neural networks for aerial inspection / Gabriel Gelard Reis de Castro. — 2022
xi, 48f. : il. (algumas color.) , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia : f. 43-48
Orientadora: Milena Faria Pinto

1. Redes neurais (Computação). 2. Drone. 3. Algorítimos. 4. Sistema de controle inteligente. 5. Engenharia elétrica. I. Pinto, Milena Faria (Orient.). II. Título.

CDD 006.3

Elaborada pela bibliotecária Tania Mello – CRB/7 nº 5507/04

Acknowledgement

My sincere thanks go to my parents for always support me, my sister for being my example, for Thaiany Rodrigues, my girl, for all the understanding and support through this journey.
Also thanks to Luana Gantert and Victor Rinaldi, my friends, for always hearing my ramblings and complaining about life. And of course for Milena Pinto, my mentor, that always push me to my limit and believed in this work.
I would also like to thank you master's thesis panel for the invitation accepted to evaluate this work.

Gabriel Gelard Reis de Castro

"Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand.

RESUMO

Dynamic Path Planning based on Neural Networks for Aerial Inspection

Gabriel Gelard Reis de Castro

Orientadora:

Prof. Milena Faria Pinto, Dsc.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em nome do programa, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, como parte dos requisitos necessários à obtenção do título de mestre/doutor.

Veículos Aéreos Não Tripulados (VANTs) são uma solução adequada para realização de inspeções autônomas em grandes estruturas que necessitam ser vistoriadas periodicamente, devido à sua flexibilidade de voo e baixo risco de acidentes envolvendo seres humanos. Como o ambiente inspecionado pode ser complexo e altamente dinâmico, faz-se necessário o uso de planejadores de rotas tridimensionais, que são cruciais para a execução de tais inspeções. Diferentes tipos de estratégias foram desenvolvidos nos últimos anos, sendo a maioria baseadas em métodos heurísticos. A desvantagem desses métodos tradicionais é que o custo computacional aumenta, dependendo da complexidade e da escalabilidade do ambiente. O planejamento de rotas baseado em Redes Neurais (NNs) surge como alternativa para suprir as desvantagens desses algoritmos.Nesse sentido, este trabalho propõe uma nova estratégia utilizando redes neurais profundas para realizar o planejamento de rotas em tempo real em um ambiente tridimensional totalmente desconhecido. O método foi desenvolvido para gerar, inicialmente, um banco de dados a ser utilizado durante o treinamento da rede e a sua performance validada em um ambiente simulado utilizando o software Gazebo/ROS. Os resultados mostraram que a topologia da rede e o método de treinamento obtiveram desempenho satisfatório e possuem aplicabilidade em um cenário real.

Planejamento de Caminhos, Redes Neurais, VANTs, Inspeções Aéreas, Ambientes Desconhecidos.

ABSTRACT

Dynamic Path Planning based on Neural Networks for Aerial Inspection

Gabriel Gelard Reis de Castro

Advisor:

Prof. Milena Faria Pinto, Dsc.

Abstract of Master Dissertation submitted to Programa de Pós-Graduação em Engenharia Elétrica, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ, as partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Unmanned Aerial Vehicles (UAVs) are a suitable solution to automate inspections on large structures that require periodic inspections due to their flexibility of flight and preventing risks to humans. However, the inspected environment could be complex and highly dynamic. Therefore, tridimensional path planners are crucial in aerial tasks. Different kinds of 3D path planning were developed in the last years, and most of them are classified as heuristic methods. A drawback of these traditional methods is that computational time depends on the environment's complexity and scale. Path planning based on Neural Networks (NNs) arises from overcoming these algorithms' bottlenecks. In this sense, this work proposes a novel deep learning structure to perform real-time path planning in an unknown environment. The strategy was developed to generate an initial database used for training and validate the results in a simulated environment using Gazebo/ROS software. The results showed that the network topology and the training method obtained satisfactory performance to be applied in a real-world scenario.

Path Planning, Neural Network, UAV, Aerial Inspection, Unknown Environments.

Contents

I Introduction **13**

I.1 Main Contributions 16

I.2 Organization 17

II Background and Related Work **18**

II.1 Path Planning Algorithms 18

II.1.1 Artificial Potential Field 19

II.1.2 Probabilistic Road Map 19

II.1.3 Node-Based Optimal Algorithms 21

II.1.4 Neural Network 22

II.2 Related Works 27

III Proposed Methodology **30**

III.1 Problem Formulation 30

III.2 Dataset 31

III.3 Neural Network 33

III.3.1 Proposed model for the neural network 33

III.3.2 Training the model 35

IV Results and Discussion **37**

IV.1 Training Process 38

IV.1.1 Number of layers and neurons 38

IV.2 Simulation Results 41

IV.2.1 Proposed methodology simulated only with Python 41

IV.2.2 Proposed methodology tested in a flight simulator 47

V Conclusions and Future Work **53**

V.0.1 Conclusions 53

V.0.2 Future Works 54

References 55

List of Tables

II.1 Comparison among some contemporaneous path planning methods and the proposed algorithm. 29

III.1 Slice of the dataset generated by Dijkstra's algorithm. 33
III.2 Neural network input example. 35
III.3 Neural network output for the input of Table III.2. 35

IV.1 First of the three better NNs models generated using randomly initialization of hyperparameters 38
IV.2 Second of the three better NNs models generated using randomly initialization of hyperparameters 38
IV.3 Third of the three better NNs models generated using randomly initialization of hyperparameters 38
IV.4 Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.1. 40
IV.5 Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.2 40
IV.6 Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.3. 40
IV.7 Performance evaluation statistics 42
IV.8 Performance evaluation statistics for moving targets 45
IV.9 Performance evaluation statistics for kidnapped problem 46

List of Figures

I.1 Inspections made by UAVs: (a) Mining. (b) Solar Power Plants. 14

II.1 3D Path Planning Taxonomy. Adapted from (1). 19
II.2 Generated path from Artificial Potential Field (APF) 20
II.3 Simulation of a simple environment with a single static object. 20
II.4 Example of node-based algorithms. Adapted from (1). 21
II.5 A fully Connected Deep Neural Network structure. 23
II.6 Classes of networks architectures. (a) Feedforward network with a single layer of neurons. (b) fully connected feedforward network with one hidden layer and one output layer. (c) recurrent network with no self-feedback loops and no hidden neurons. 24
II.7 IRIS dataset taken from (2) 26

III.1 Proposed methodology flowchart. 30
III.2 Simulated first world,using python, to create the training dataset. (a) Isometric view from world 1. (b) Lateral view from world 1. 31
III.3 Simulated second world,using python, to create the training dataset. (a) Isometric view from world 2. (b) Lateral view from world 2. 31
III.4 Simulated third world,using python, to create the training dataset. (a) Isometric view from world 3. (b) Lateral view from world 3. 32
III.5 Example of path generated by Dijkstra's algorithm in world 2. 32
III.6 Sensor's view directions. 32
III.7 Proposed architecture of neural network. 34
III.8 Implementation of a ReLu function. 34
III.9 N-fold Cross-Validation train method. 36

IV.1 Infographic representation of the processing steps. 37
IV.2 Random models. 39
IV.3 Training results for Model 1,2 and 3. 41

IV.4 Simulated worlds to create training dataset. (a) Isometric view from simulated world 1. (b) Lateral view from simulated world 1. (c) Isometric view from simulated world 2. (d) Lateral view from simulated world 2. (e) Isometric view from simulated world 3. (f) Lateral view from simulated world 3. 42
IV.5 Simulations results for static targets. 43
IV.6 Histograms based on IV.5 43
IV.7 Isometric and top view of paths generated by RRT* and Neural networks in world 1. (a)Isometric view for path with start point = [15,15,10] and target = [25,15,15]. (b) Lateral view for path with start point = [15,15,10] and target = [25,15,15]. (c) Isometric view for path with start point = [15,10,10] and target = [15,25,20] (d) Lateral view for path with start point = [15,10,10] and target = [15,25,20] (e) Isometric view for path with start point = [20,10,5] and target = [15,20,25] (f) Lateral view for path with start point = [20,10,5] and target = [15,20,25]. 44
IV.8 Simulations results for moving targets. 45
IV.9 Histograms based on IV.8 45
IV.10 Simulations results for kidnapped problem 46
IV.11 47
IV.12 Simulation using Neural Network to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3. 48
IV.13 Simulation using Neural Network to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3. 49
IV.14 Simulation using rapidly exploring random tree to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3. . . . 49
IV.15 UAV model in Gazebo/ROS. 50
IV.16 Gazebo world used in the simulations. 50
IV.17 UAV performing inspection in hydro-power plant in gazebo simulated world. 51
IV.18 UAV vision during the pipe inspection in gazebo simulated world. 51
IV.19 Path generated from Neural Network to perform the inspection in gazebo simulated world. 52

List of Abbreviations

2.5D	Two And A Half Dimensional . . . 27
2D	Two-dimensional . . . 27
3D	Three-dimensional . . . 27
ANNS	Artificial Neural Networks . . . 22, 23
APF	Artificial Potential Field . . . ix, 19, 20, 27
CNN	Convolutional Neural Network . . . 15, 28
CPP	Coverage Path Planning . . . 13, 14
DNN	Deep Neural Network . . . 15
ECD	Exact Cellular Decomposition . . . 15
GPS	Global Positioning System . . . 46
LIDAR	Light Detection And Ranging . . . 28, 31
LPA	Lifelong Planning A^* . . . 22, 27
MAE	Mean Absolute Error . . . 34, 35
MSE	Minimum Square Error . . . 41
NNS	Neural Network . . . 15, 16, 22, 28, 31, 33, 34, 38, 39, 41, 44, 46, 47, 53
PRM	Probabilistic Road Map . . . 18, 19, 20, 22, 27, 28, 31, 41
PSO	Particle-Swarm-Optimization . . . 14
RELU	Rectified Linear Unit . . . 33
ROS	Robot Operating System . . . 16, 47, 53
RRT	Exploring Random Trees . . . 18, 41, 44, 46, 47
TDPP-NET	Three-Dimensional Path Planning Network . . . 15
UAV	Unmanned Aerial Vehicle . . . 13, 14, 19, 23, 27, 31, 33, 34, 46, 47, 54
VIN	Value Iteration Network . . . 15

Chapter I Introduction

In the last decades, UAVs have been used in many fields of industry for aerial inspection (3), such as in monitor crop field (4), delivery goods (5), civil structures inspection (3, 6), surveillance (7), archaeological exploration (8), among others. This wide usability is explained by the UAV's ability to perform complex activities with maneuvering flexibility and low-cost flight.

UAVs have the ability to easily reach places that human beings can't or are too risky for them. High/low temperatures or pressure, insalubrious/tiny spaces like swearers and mines, and higher altitudes are examples of these areas. Although all this danger, humanity needs to explore these locals in order to develop new technology, to fasten processes, and to maintain systems properly working (9). In the light of this last case, we have many applications that Unmanned Aerial Vehicle (UAV)s are responsible for making inspections, i.e., in hydro power plants, in the chemical industry, in oil and gas pumps, and telecommunications towers. Figure I.1 shows some of these robotic systems performing inspections. In Figure I.1 (a), the UAV is perfoming an inspection in a dam site. Figure I.1 (b) presents a UAV inspection a solar farm. In both applications, it is possible to note that the UAV should perform the inspection in a large area.

The path planning for UAV is considered a vital task, once it should move from one position to another safely, avoiding obstacles coming and taking the shortest path from all the available options. As stated by (10), secure and impressive path planning methods are usually made by solving related optimization problems. In the case of UAVs, path planning algorithms are a prominent solution to automate navigation and mapping. For instance, this kind of algorithm is extremely important for inspection applications, especially in large structures that periodic inspections are a mandatory requirement to verify possible structural changes preventing risks to humans and environmental disaster (11).

This kind of aerial inspection could help decrease the mission's complexity. For instance, due to their maneuvering flexibility, high flight versatility, and unique working, including search and rescue, real-time monitoring, Information Collection, Transmission and delivering goods (5, 12). Therefore, it is possible to observe that the UAV technology has attracted huge interest in many applications (3, 7, 13).

Note that the inspection using UAVs should guarantee flight time optimization due to battery time restrictions and best trajectory to cover a determined area (14, 15, 16). Besides, UAVs could easily data gathering and geometry land specifications, such as explained in (11).

In this sense, Coverage Path Planning (CPP) is a suitable technique to be applied in this kind of



(a)



(b)

Figure I.1: Inspections made by UAVs: (a) Mining. (b) Solar Power Plants.
Adapted from: https://www.doosanmobility.com/

task (17). CPP aims to find the best path to cover a particular area passing through all viewpoints in a collision-free trajectory. As a result of the inspection, the UAV acquires images from the local inspected area and sends them to a post-process stage to build the desired accurate 3D model. For more details of this process, the author refers to (6).

Optimal path planning is also an interesting research topic that many works have addressed in the last years. Most of the optimal path planning methods are classified as heuristic methods, such as Ant Colony Optimization (18) or Concorde (19), or as meta-heuristic methods, such as genetic algorithm (20), or discrete Particle-Swarm-Optimization (PSO) (21). As a drawback of these conventional approaches, their computational time depends on the environment complexity and scale (22). A promising solution to overcome these traditional algorithms' bottlenecks is the use of deep neural network architectures, such as feed-forward neural networks. This kind of architecture can also learn to plan efficiently under various conditions.

A widely used meta-heuristic method is particle swarm optimization (PSO). However, implementing the PSO to a discrete problem requires some adjustments (21, 23). It is important to note that meta-

heuristic and heuristic methods to UAV routing problem have also been conducted by several researchers. For instance, Biundini et al. (3) applied a genetic algorithm in a UAV for coverage path planning problems in inspection scenarios. In Lim et al. (24), the authors applied the traveling salesman problem with a time window and solved it with dynamic programming.

In the last decades, many researchers have been using Neural Network (NNs) to the path planning field to deal with nonlinearities and uncertainties of the environment. NNss are being used in a wide range of applications, semantic segmentation (25), object detection (26), action recognition (27), among others. Recently, they are widely applied to path planning problems (28, 29). As can be seen, Deep Neural Network (DNN)-based 2D-path planning is not a novelty (30). For instance, the work of Tamar et al. (31) proposed the Value Iteration Network (VIN) algorithm to approximate the Bellman update using Convolutional Neural Network (CNN). Different applications used VIN due to its ultimate ability to solve 2D path trajectories. Khan et al. (32) used this method for value map learning in partially observable environments. Another application was used in Gupta et al. (33), where the authors proposed a version of VIN to generate actions for a visual navigation system. Okada et al. (34) proposed a path integral network for optimal continuous controlling.

3D path planning is a crucial approach to be applied in complex and highly dynamic environments (35), especially in the robotics field (36). Different kind of 3D approaches were developed in the last years, such as Rapidly-exploring Random Tree (37), Exact Cellular Decomposition (ECD) (38), Probabilistic Roadmap (39), Visibility Graph (40), among others. Most of these conventional algorithms rely on heuristics, which may be trapped in local solutions, not finding the complete path. In this sense, DNN-based 3D path planners were developed to address this issue. As stated in (29), NNs have the advantages of capturing both global structures and local details of observed environments and generating spatially structured outputs.

It should be noted that although 3D path planning plays a vital role in autonomous robots, it is rarely handled by DNNs. However, in recent years, some works have been addressing this issue by employing DNN path planners. For instance, Wu et al. (41) proposed novel end-to-end neural network architecture, named Three-Dimensional Path Planning Network (TDPP-Net), to perform DNN-based 3D path planning. Their results showed an indecency of environmental scale and complexity for computational time. Besides, the experimental results demonstrated that their approach outperformed the work of Tamar et al. (31), which proposed a solution based on CNN for a 2D environment. In Wu et al. (42), the authors developed a DNN-based methodology for real-time online path planning designed to learn 3D local path planning policies and determine 3D space actions based on multiple value iteration computations approximated by recurrent 2D convolutional neural networks. Differently from the other works, they demonstrated their results employing virtual and real-world experimental results.

I.1 Main Contributions

This research proposes the use of a neural network structure to perform real-time path planning in 3D unknown environments. In the research, the authors developed a strategy to generate the initial database used for training and validate the results. In this sense, Dijkstra's algorithm was used to generate a proper optimization that will feed the path database from the different scenarios created using random characteristics, and therefore, the global solutions will be available to perform training on the intelligent method. The deep learning method was trained using a cross-validation method, an Adam optimizer algorithm, a categorical cross-entropy as the loss function, and binary accuracy as the metric to evaluate the training performance. This work presents the most simple structure capable of using deep learning and perform localize planning. Further methods can be developed and enhanced based on this structure. In light of this previously mentioned context, this research work's main contributions can be summarized as:

- A dataset based on information produced by a LIDAR sensor and based on a classic offline global path planning algorithm to train a NNs;
- The use of a deep neural network structure to perform local path planning capable of dealing with kidnapping, unknown environments, and moving targets;
- Testing the proposed path planning algorithm in the Iris UAV by starting PX4 SITL and Gazebo software in Robot Operating System (ROS) to prove its functionality in a real application.

I.2 Organization

Chapter II will address the theoretical foundation necessary for the development of this research work. Also, in this chapter, all functioning and construction of neural networks and path planning algorithms are detailed.

Chapter III presents the methodological foundation for the proposed path planning based on neural networks for aerial vehicles. Also, in this chapter, the entire working process of the developed methodology will be explained in detail.

The results obtained through the methods discussed are shown in Chapter IV. In addition, the necessary discussions will be held about the results to demonstrate the efficiency of the obtained method.

The conclusions obtained throughout this dissertation, as well as the various possibilities for continuing the research, are presented in Chapter V.

Chapter II Background and Related Work

This Chapter will introduce some important aspects of the most used path planning algorithms for UAVs, aiming at demonstrating the drawback and positive aspects of them. Then, this Chapter will detail the mathematical foundations of Neural Networks to support the aspects addressed in the subsection Related Works.

II.1 Path Planning Algorithms

According to (43), path planning can be defined as "...the motion planning problem is to find a sequence of control inputs so as the drive the robot from its initial state to one of the goal states while obeying the rules of the environment, e.g., not colliding with the surrounding obstacles." Another definition is found in (44), where the authors state that the path planning problem is" Given a three-dimensional rigid body with a set of configuration parameters and a known set of stationary obstacles, find a collision-free path from a start configuration to a goal configuration". Based on these two statements, this work presents a definition established on (44), since it better defines the problem that this work intends to tackle.

There are a lot of algorithms that have been studied in the literature that can handle this problem. Every one of them has its applications, depending on what kind of robot and environment the problem is facing. Note that other features are relevant too, for instance, worlds simplifications such as grids, and of course, the dimension of the problem, i.e., three-dimensional (3D) one or two-dimensional (2D). This chapter focus on methods that successfully accomplished the path planning problem in a 3D world, such as Exploring Random Trees (RRT) (45), Probabilistic Road Map (PRM) (46), RRT, Bio-inspired algorithms, graph-based algorithms, and their variations.

3D path planning algorithms can be divided into five categories, which are distinguished from each other by their unique properties. A complete survey of 3D path planning can be found in (1). Figure II.1 illustrates the current approaches for 3D path planning. The following subsections will detail the most used successful 3D path planning methods.

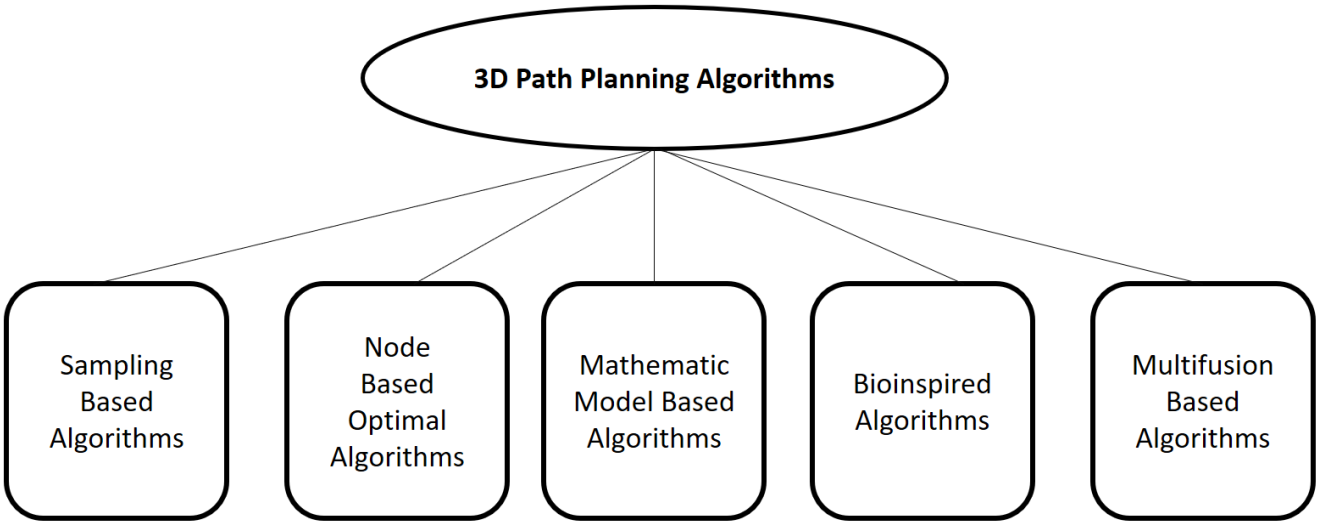


Figure II.1: 3D Path Planning Taxonomy. Adapted from (1).

II.1.1 Artificial Potential Field

APF is a well-known algorithm applied in robots. This method uses attractive magnetic forces to reach the goal and repulsive ones for obstacles. There are a few problems with that implementation. One of them is the local minima area. The other one is the Goal Not Reachable when Obstacles are Nearby (GNRON). That application was first used by (45) and (47) proposed applying it in a 3D environment, for UAV path planning. The problems encountered by (45) were outlined by (47) using a graph algorithm to explore the area around the obstacles and creating a potential virtual force surround them. It's important to say that in order to apply this kind of algorithm, it is necessary to have known environments with static objects. Figure II.2 exemplifies the APF algorithm. This technique can overcome unknown scenarios, taking into account the realities of the current environment of the robot motion. As stated before, this method is based on the attractive potential field due to the target and the repulsive potential field due to the world's obstacles.

II.1.2 Probabilistic Road Map

Another important group that has been extensively studied is the PRM method. It was implemented by (46), and it has two stages: (i) learning; and (ii) the query. The roadmap is constructed in the learning phase by taking random points on the map, checking if they are collision-free, and connecting them using a motion planner. After that, these points and connections are saved as an undirected graph. In the next step of the method, a query asks for a path between two free nodes on the roadmap. Thus, a graph search algorithm is used to finding the solution if it exists. As explained by (46), it is important to note that the execution of these two phases, i.e., learning and query, do not have to be sequential.

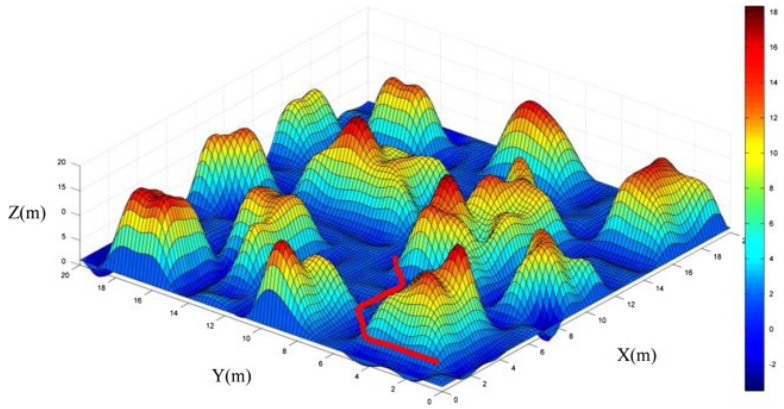


Figure II.2: Generated path from APF .
Taken from :
https://medium.com/@rymshasiddiqui/path-planning-using-potential-field-algorithm-a30ad12bdb08

Instead, these phases can be interwoven to adapt the size of the roadmap to difficulties that the robot can encounter during the query phase, increasing the method's learning flavor. One of the problems with this implementation is the time that the algorithm takes to achieve the solution. Depending on how complex the map is, the learning stage can time exhausting because of the number of collisions.

Therefore, in order to minimize time spent in the learning phase, another PRM was proposed by (48). In their methodology, the learning phase has no collision check. The algorithm takes the shortest path between start and goal positions. Then, it checks only if that path is collision-free. Supposing a node in that path is blocked, the algorithm would delete it and replace it with a neighbor node. Then, repeat the process until it returns a free-collision path. Figure II.3 shows a map with only one object in a 3D world, Equation II.1 is its respectively Probabilistic road map, where free spaces are 0, and occupied ones are 1.

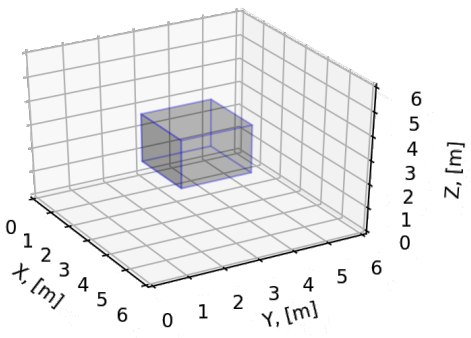


Figure II.3: Simulation of a simple environment with a single static object.

$$PRM = \left[\begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{bmatrix}\right] \tag{II.1}$$

II.1.3 Node-Based Optimal Algorithms

According to (49), this kind of method explores through the decomposed graph. This algorithm shares the same property that they explore among a set of nodes in the map. Figure II.4 illustrates examples of node-based optimal algorithms.

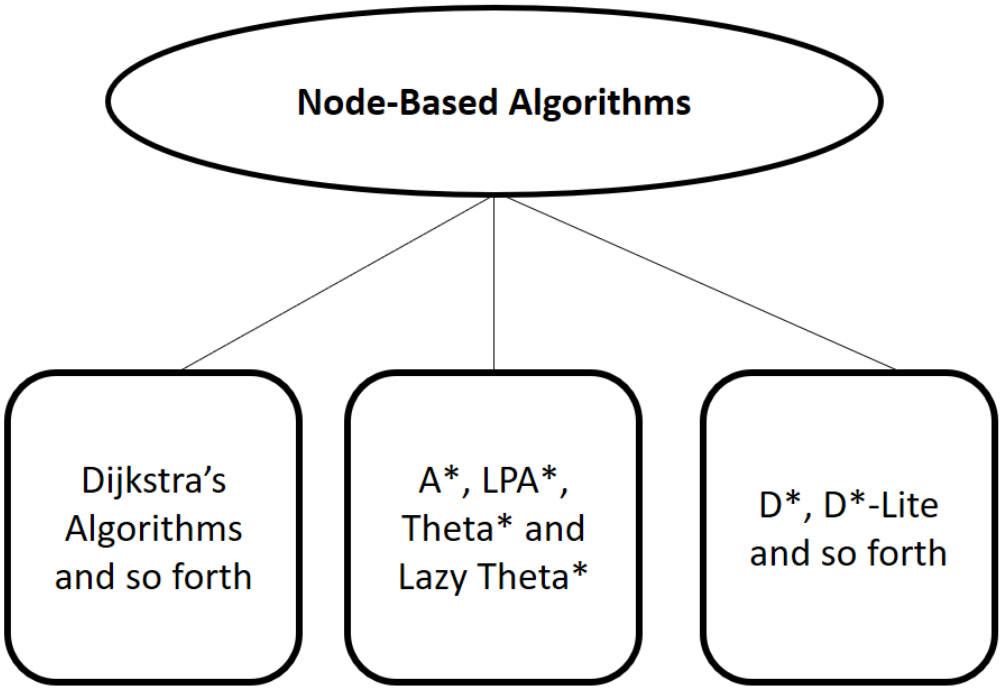


Figure II.4: Example of node-based algorithms. Adapted from (1).

This kind of algorithm explores through a graph searching for optimal paths. The drawback of this kind of method is the need for previous information about the map, like a PRM. On the other hand, it always finds an optimal solution if it exists, according to map decomposition (grid). In the works of Dijkstra (50), A* (51) ,Lifelong planning A^* (LPA) (52), D* (53), D*-lite (54), it is possible to find examples of these algorithms, commonly classified as discrete optimal planning or road map algorithms.

An important node-based optimal algorithm is the Dijkstra. This algorithm was firstly implemented by (50), and it was developed to solve two problems. The first one is, given n nodes in a direct graph, find the minimum path between them, the second one is given n nodes in an undirect graph, also find the minimum path. It's important to note that Dijkstra always returns a globally optimal solution, or in other words, the smaller possible path among the given nodes and that in order to perform the edges, weights must be already known. A pseudo-code of the method is shown in Algorithm 1.

Algorithm 1 Dijkstra's algorithm generalization.

```
for Each node in graph do
    Distance[V] := ∞// Initial distance from source to target V is infinite
    Previous[V] := -1
end for
Distance[source] := 0
S := all nodes in Graph
while S is not empty do
    u := node in S with smallest Distance[]
    remove u from S
    for neighbor v of u do
        new distance = Distance[u] + distance(u,v)
        if new distance < Distance[V] then
            Distance[V]:= new distance
            Previous[V] := u
        end if
    end for
end while
return Previous[]
```

In the past years, this method has been extensively studied and improved to solve more complex problems, with dynamic obstacles, unknown maps, and so on (55, 56, 57).

II.1.4 Neural Network

A neural network is a class of bio-inspired algorithms that intends to reflect the behavior of the human brain, allowing it to mimic features of other algorithms, recognize patterns in datasets and solve problems. NNs or Artificial Neural Networks (ANNs) are composed of a set of layers, and these layers are composed of a set of neurons. There are three kinds of layers: input, output, and a hidden layer. Figure II.5 illustrates this structure. Note that the input layer receives the data information, and the output returns the solution based on the mathematical operations performed on the hidden layers.

ANNs can be applied in many practical applications, such as *i)* data analysis for business intelligence, medical applications, support financial decisions, military applications, 4.0 industry; *ii)* When the task has dynamic elements, and the algorithm needs to be adapted; *iii)* In face and object recognition; and *iv)* When handling huge online data to make decisions, like self-driven cars and some UAV applications.

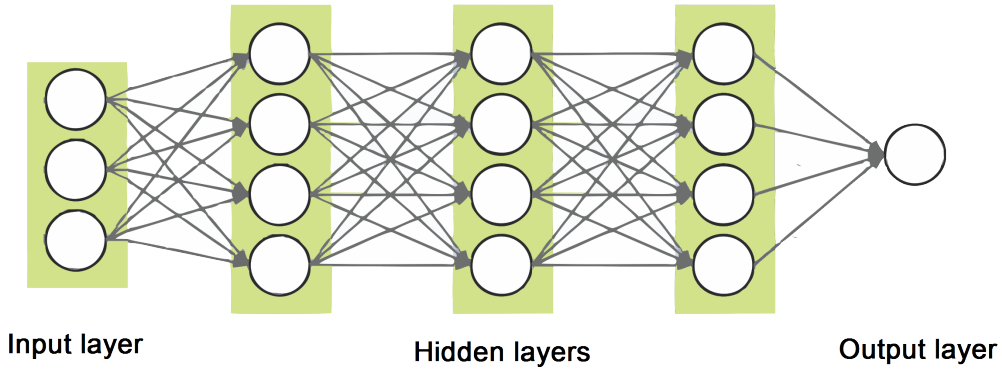


Figure II.5: A fully Connected Deep Neural Network structure.

There are many different structures for neural networks. Each one of them has its own characteristics, and they are used in different scenarios. They can be divided into three fundamentally different classes of networks architectures (58): *(i)* Singles layer Feedforward Networks; *(ii)* Multilayer Feedforward Networks; and *(iii)* Recurrent Networks. Figure II.6 presents their topology.

Each node of the NN works like a perceptron (59). The input is the vector $x_1, x_2, x_3, ..., x_n$, the weights are denoted by $\omega_1, \omega_2, \omega_3, ..., \omega_n$, the bias as b, and the output is y. The goal of the perceptron is to, by using the input information, classify the problem analysing the output into θ_1 or θ_2. The operation can be better understand through Equation II.2.

$$y = \sum_{i=1}^{n} \omega_i \; x_i + b \tag{II.2}$$

The decision rule for the classification is to obtain a linear combination of input and weights that would correspond to class θ_1 if y is $+1$ and to class θ_2 if it is -1. The processing of finding these weights in order to accomplish that task is called training the NN.

II.1.4.1 Training phase

The training phase is the part of the algorithm that tries to mimic or learn behavior from a dataset. (60) defines the processing of learning as "A computer program is said to learn from experience E concerning some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E". The task T that this work proposes to tackle is to find a path between two points in a 3D unknown map by using the kinematic of a UAV. The performance P

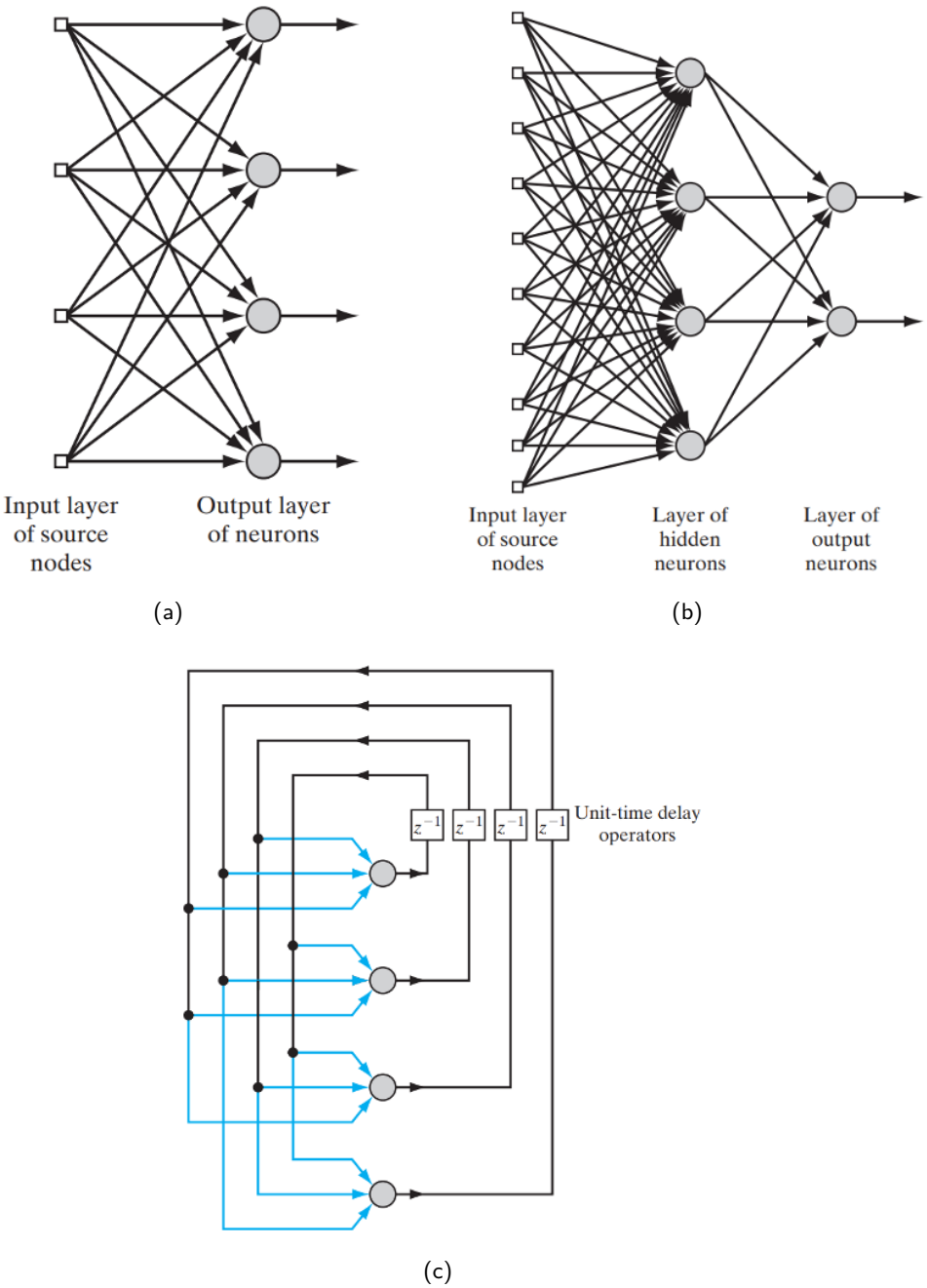


Figure II.6: Classes of networks architectures. (a) Feedforward network with a single layer of neurons. (b) fully connected feedforward network with one hidden layer and one output layer. (c) recurrent network with no self-feedback loops and no hidden neurons.
Adapted from : (58)

and the Experience E can assume different shapes to accomplish this task. Note that Section Related Works will discuss different E and P changes performance. In this section, it will be presented a brief understanding of some techniques.

II.1.4.1.1 The performance measure P

The choice of a metric is normally associated with the task. For example, suppose the task is a classification problem. In that case, accuracy is commonly used as a measure, since it is just a direct relation between the expected output and the given one, if they are equal or not. If the task is a regression problem, the mean square error could be a good metric. In the case of density estimation, it is usually applied a probabilistic metric like binary cross-entropy, Poisson, or categorical cross-entropy would give the needed information. It is very important to choose wisely this variable. In the machine learning field, it is often desired to evaluate how good a model is when facing data that the NN has never been exposed to, simulating a real-world experience. The wrong choice of metrics can give the illusion that the algorithm performs well or that the training phase results are inaccurate. Another use for metrics is to be a control variable during training in order to try to reduce overfitting. Note that Chapter III will better explain this process. Thus, to summarize, it is possible to see how different metrics could generate different interpretations. There are lots of metrics with many variations. It's often a difficult task to choose what should be the measure to improve the NN performance.

II.1.4.1.2 The experience E

The experience E is usually obtained through a collection of examples, which is called a dataset. These examples could be labeled or not. "Machine learning algorithms can be broadly categorized as unsupervised or supervised by what kind of experience they are allowed to have during the learning process" (61). Supervised learning is the group of algorithms that experience or learn from a dataset that has labeled examples. In this case, the NN tries to mimic the comportment from the examples. On the other hand, unsupervised learning is the group that learns from data without labels. A case of label examples is the iris dataset (2), which shows that using a linear combination of the measurement of the sepal width/length and the petal width/length is possible to determine if the iris is setosa, versicolor or virginica. Table II.7 shows the studied cases of (2).

II.1.4.2 Overfitting and Underfitting

As stated by (61), "The factors determining how well a machine learning algorithm will perform are its ability to: make the training error small and make the gap between training and test error small." These two measures are essential to determine the capacity of generalization of the NN, which is one of the main features of this method. Generalization is the ability that the model has to respond properly when facing data points that have never been exposed during the training phase, this data is usually labeled as a test dataset.

Iris setosa				*Iris versicolor*				*Iris virginica*			
Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
5·1	3·5	1·4	0·2	7·0	3·2	4·7	1·4	6·3	3·3	6·0	2·5
4·9	3·0	1·4	0·2	6·4	3·2	4·5	1·5	5·8	2·7	5·1	1·9
4·7	3·2	1·3	0·2	6·9	3·1	4·9	1·5	7·1	3·0	5·9	2·1
4·6	3·1	1·5	0·2	5·5	2·3	4·0	1·3	6·3	2·9	5·6	1·8
5·0	3·6	1·4	0·2	6·5	2·8	4·6	1·5	6·5	3·0	5·8	2·2
5·4	3·9	1·7	0·4	5·7	2·8	4·5	1·3	7·6	3·0	6·6	2·1
4·6	3·4	1·4	0·3	6·3	3·3	4·7	1·6	4·9	2·5	4·5	1·7
5·0	3·4	1·5	0·2	4·9	2·4	3·3	1·0	7·3	2·9	6·3	1·8
4·4	2·9	1·4	0·2	6·6	2·9	4·6	1·3	6·7	2·5	5·8	1·8
4·9	3·1	1·5	0·1	5·2	2·7	3·9	1·4	7·2	3·6	6·1	2·5
5·4	3·7	1·5	0·2	5·0	2·0	3·5	1·0	6·5	3·2	5·1	2·0
4·8	3·4	1·6	0·2	5·9	3·0	4·2	1·5	6·4	2·7	5·3	1·9
4·8	3·0	1·4	0·1	6·0	2·2	4·0	1·0	6·8	3·0	5·5	2·1
4·3	3·0	1·1	0·1	6·1	2·9	4·7	1·4	5·7	2·5	5·0	2·0
5·8	4·0	1·2	0·2	5·6	2·9	3·6	1·3	5·8	2·8	5·1	2·4
5·7	4·4	1·5	0·4	6·7	3·1	4·4	1·4	6·4	3·2	5·3	2·3
5·4	3·9	1·3	0·4	5·6	3·0	4·5	1·5	6·5	3·0	5·5	1·8
5·1	3·5	1·4	0·3	5·8	2·7	4·1	1·0	7·7	3·8	6·7	2·2
5·7	3·8	1·7	0·3	6·2	2·2	4·5	1·5	7·7	2·6	6·9	2·3
5·1	3·8	1·5	0·3	5·6	2·5	3·9	1·1	6·0	2·2	5·0	1·5
5·4	3·4	1·7	0·2	5·9	3·2	4·8	1·8	6·9	3·2	5·7	2·3
5·1	3·7	1·5	0·4	6·1	2·8	4·0	1·3	5·6	2·8	4·9	2·0
4·6	3·6	1·0	0·2	6·3	2·5	4·9	1·5	7·7	2·8	6·7	2·0
5·1	3·3	1·7	0·5	6·1	2·8	4·7	1·2	6·3	2·7	4·9	1·8
4·8	3·4	1·9	0·2	6·4	2·9	4·3	1·3	6·7	3·3	5·7	2·1
5·0	3·0	1·6	0·2	6·6	3·0	4·4	1·4	7·2	3·2	6·0	1·8
5·0	3·4	1·6	0·4	6·8	2·8	4·8	1·4	6·2	2·8	4·8	1·8
5·2	3·5	1·5	0·2	6·7	3·0	5·0	1·7	6·1	3·0	4·9	1·8
5·2	3·4	1·4	0·2	6·0	2·9	4·5	1·5	6·4	2·8	5·6	2·1
4·7	3·2	1·6	0·2	5·7	2·6	3·5	1·0	7·2	3·0	5·8	1·6
4·8	3·1	1·6	0·2	5·5	2·4	3·8	1·1	7·4	2·8	6·1	1·9
5·4	3·4	1·5	0·4	5·5	2·4	3·7	1·0	7·9	3·8	6·4	2·0
5·2	4·1	1·5	0·1	5·8	2·7	3·9	1·2	6·4	2·8	5·6	2·2
5·5	4·2	1·4	0·2	6·0	2·7	5·1	1·6	6·3	2·8	5·1	1·5
4·9	3·1	1·5	0·2	5·4	3·0	4·5	1·5	6·1	2·6	5·6	1·4
5·0	3·2	1·2	0·2	6·0	3·4	4·5	1·6	7·7	3·0	6·1	2·3
5·5	3·5	1·3	0·2	6·7	3·1	4·7	1·5	6·3	3·4	5·6	2·4
4·9	3·6	1·4	0·1	6·3	2·3	4·4	1·3	6·4	3·1	5·5	1·8
4·4	3·0	1·3	0·2	5·6	3·0	4·1	1·3	6·0	3·0	4·8	1·8
5·1	3·4	1·5	0·2	5·5	2·5	4·0	1·3	6·9	3·1	5·4	2·1
5·0	3·5	1·3	0·3	5·5	2·6	4·4	1·2	6·7	3·1	5·6	2·4
4·5	2·3	1·3	0·3	6·1	3·0	4·6	1·4	6·9	3·1	5·1	2·3
4·4	3·2	1·3	0·2	5·8	2·6	4·0	1·2	5·8	2·7	5·1	1·9
5·0	3·5	1·6	0·6	5·0	2·3	3·3	1·0	6·8	3·2	5·9	2·3
5·1	3·8	1·9	0·4	5·6	2·7	4·2	1·3	6·7	3·3	5·7	2·5
4·8	3·0	1·4	0·3	5·7	3·0	4·2	1·2	6·7	3·0	5·2	2·3
5·1	3·8	1·6	0·2	5·7	2·9	4·2	1·3	6·3	2·5	5·0	1·9
4·6	3·2	1·4	0·2	6·2	2·9	4·3	1·3	6·5	3·0	5·2	2·0
5·3	3·7	1·5	0·2	5·1	2·5	3·0	1·1	6·2	3·4	5·4	2·3
5·0	3·3	1·4	0·2	5·7	2·8	4·1	1·3	5·9	3·0	5·1	1·8

Figure II.7: IRIS dataset taken from (2)

Overfitting happens when there is a big difference between training error and test error. Underfitting happens when the model performed badly during the training phase, i.e., when returning a large error on training data. There are many ways to reduce these two errors, one of them is to change the capacity model, lower capacity can underfit and higher overfit. Others strategies are regularization like early stopping and weight decay, augment dataset, adding noise on input or target, bootstrap aggregating.

II.2 Related Works

An important aspect during the inspection is path planning. As exposed in the initial part of Chapter II, path planning has been extremely studied in the past years for Two-dimensional (2D) and Three-dimensional (3D) environments. However, simple 2D path planning algorithms are not able to deal with complex 3D environments. Therefore, 3D path planning algorithms for UAV navigation are needed to support navigation in complex environments. Note that these difficulties increase exponentially with dynamic and kinematic constraints in path planning in 3D environments, making this kind of problem very difficult. It is relevant to say that many of 3D path plannings are in reality Two and a half dimensional (2.5D) since they take height as a constant and then solve path planning for a 3D sliced world.

3D path planning have different characteristics and can be applied to different kinds of robots and environments. In the last year, many algorithms methods were proposed to deal with a such kind of problem, including Visibility Graph (62), randomly sampling search algorithms like Rapidly-exploring Random Tree (37), and Probabilistic Roadmap (39), optimal search algorithms like Dijsktra's algorithm (50), A* (51), and D* (63) bio-inspired planning algorithms (20), among others.

Classic node search algorithms, such as Dijkstra's (50) have been sophisticated through the past years. For instance, the method A^*, proposed in the work of (51), is just a Dijkstra algorithm with a heuristic approach to reduce the number of states. This algorithm was used by (64) in a 3D environment, constructing a road map and then applying the method. The authors in (39) applied A^* for a UAV along with an octree-based PRM. This method divides the 3D space into voxels, reducing the time spent during A^* search. The methods D^* (53) short for dynamic A^* and Lifelong planning A^* LPA (52), are branch's of A^*. The algorithm D^* uses sensors to update a temporal map, changing in real-time the edges' weights and returning a free-collision path. The algorithm LPA can recalculate the path when the environment changes using previous information and interactive replanning through sensors.

An improved version of APF was used for UAV path planning in a dynamic environment by Chen et al. (65). This approach eliminates classic APF problems, such as minimum local. The issues that graph search methods rely on are closely the same. They take too long in constructing PRM and then searching among the nodes. In simple environments, this method returns great results but facing more complex ones, it tends to spend lots of time. On the other hand, APF methods can give great results for partially known environments, and as exposed by (65), with non-static worlds too.

Over the years, Neural Network algorithms for robotic systems path planning are growing due to the many different applications and advantages of using the Bioinspired Neural Network algorithm for robot path planning (66, 67, 68). Neural Networks were first applied to avoid obstacles and to perform path planning by (69). They used a Hopfield structure (70) in a dynamic online application.

Sophisticated approaches have been developed in the last years, multiple-UAV problems (71), hybrid algorithms like (10) that uses genetic algorithms with NNs and (72). More recently, works use CNN to analyze camera images and then to generate a free collision path (41) into an unknown dynamic environment. Others use reinforcement learning (73) to achieve better results than Q-learning-based algorithms. At last, there are applications that use feedforwards NNs that are trained based on search node algorithms (74). The problem about CNN is the the computational time spent in the convolutions layers.

Other similar methods have been proposed. In (75) a Deep Learning method is proposed using Light Detection And Ranging (LIDAR) input for 2D environment. However this results in a large input vector and also in a more complex training. The method proposed here reduces this factors by using a more simplified input vector containing only information about the closer elements. In (76) GA is used to train a CNN. In this implementation however only simple UAV position information is used instead, reducing the CNN potential. As can be noticed many of the similar methods proposed still rely on complex network strategies or still need some characteristics to enable deployment in real world scenarios.

Different from previously mentioned works, this research proposes a hybrid algorithm by using PRM to generate a graph road map and Dijkstra algorithm to find the path, and then to create the training and test dataset to feed and train the feed-forward neural network. These NNs are so used to work with aerial inspections. This methodology differs from the others presented before because it can be applied to a totally unknown dynamic world with a low computational cost. Another advantage is the capacity of handled kidnapped problems and moving targets. Table II.1 shows a brief comparison among some contemporaneous path planning methods and the proposed algorithm. The advantages and drawbacks are highlighted.

Table II.1: Comparison among some contemporaneous path planning methods and the proposed algorithm.

Method	**Advantages**	**Drawbacks**
RRT	Simple Implementation, low computational time	Only works in static environments, Non-optimal paths
PRM	Can handle complex environments	Only works in static environments, Non-optimal Time expensive in collisions check
Dijkstra's Algorithm	Easy implementations in any environment	High time complexity Static environment only
A^*	Fast searching ability, enabling implementation on-line Fast	Static environment only, heavy time burden, Non-smoothness
APM	Fast convergence	Local minima
Genetic algorithm	Able to solve NP-hard and multiobjectives problems	High time complexity Premature convergence
CNN	Can handle dynamic environments and unknown environments	High complexity in convolution operations, high number of hyperparameters, Non-optimal
Proposed Deep Neural Network algorithm	Can handle dynamic and unknown environments, low complexity	Non-Smoothness, Non-optimal

Chapter III Proposed Methodology

III.1 Problem Formulation

This work has three important methods to be discussed: *(i)* the proposed deep learning model itself, the implementation, and the design have to be evaluated; *(ii)* the strategy used to generate a database and perform training of the proposed model; *(iii)* and the methodology used to apply the model developed and how results were gathered. Figure III.1 presents the proposed methodology diagram to generate this database and train the model.

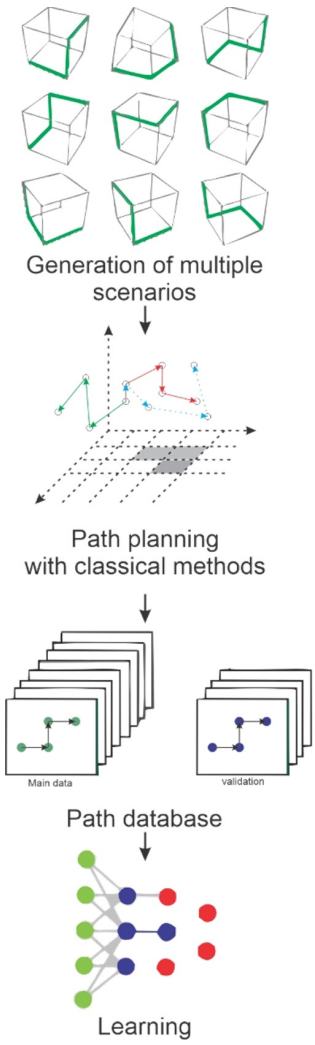
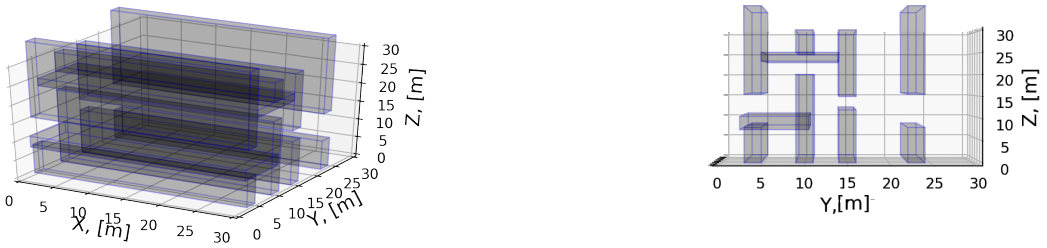


Figure III.1: Proposed methodology flowchart.

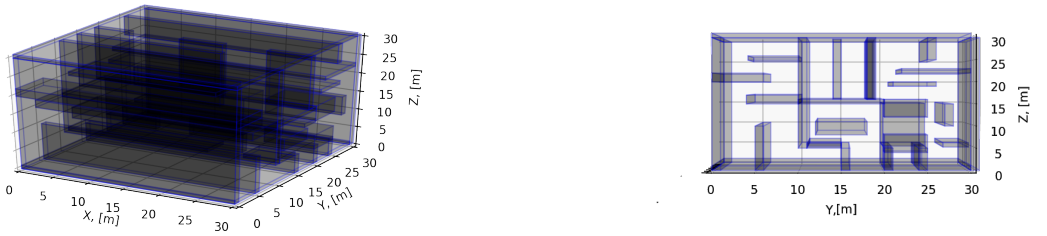
III.2 Dataset

The dataset that will be use for training is crucial, since NNs aims to generalize the dataset behavior. In this work, the data should inform us what the actions the UAV should take based on the environmental conditions obtained by its sensors and its pose. The sensor utilized was a LIDAR producing a cloud point that is equivalent to the surface inspected area. In order to obtain a dataset that reflects this behavior, a series of possible situations were simulated in three different worlds, as shown in Figure III.2, Figure III.3, and Figure III.4, respectively. A PRM was created using python3, in order to reflect the objects and constrains of these worlds and passes this information as a graph.



(a) (b)

Figure III.2: Simulated first world,using python, to create the training dataset. (a) Isometric view from world 1. (b) Lateral view from world 1.



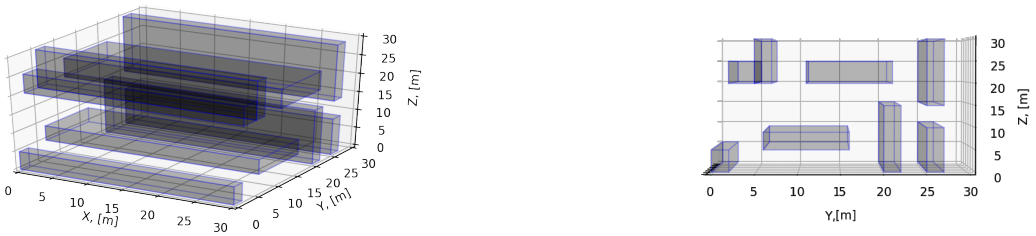
(a) (b)

Figure III.3: Simulated second world,using python, to create the training dataset. (a) Isometric view from world 2. (b) Lateral view from world 2.

Figure III.5 shows a complete path generated by Dijkstra's algorithm in of these, where the 'x' is the target pose, and the dot the start point.

The UAV movement is limited in six directions, as shown in Figure III.6, where the cubes are the possibles next UAV positions.

The minimum distance between two objects are at least three grids of distance, in order to avoid collisions due to the UAV dimensions. The dataset contains information extracted from the path generated, that is, in every grid that the robot passes, the sensors readings, the pose and the target are stored.



(a) (b)

Figure III.4: Simulated third world,using python, to create the training dataset. (a) Isometric view from world 3. (b) Lateral view from world 3.

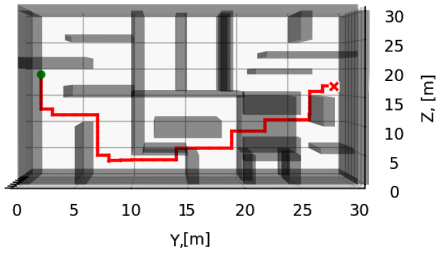


Figure III.5: Example of path generated by Dijkstra's algorithm in world 2.

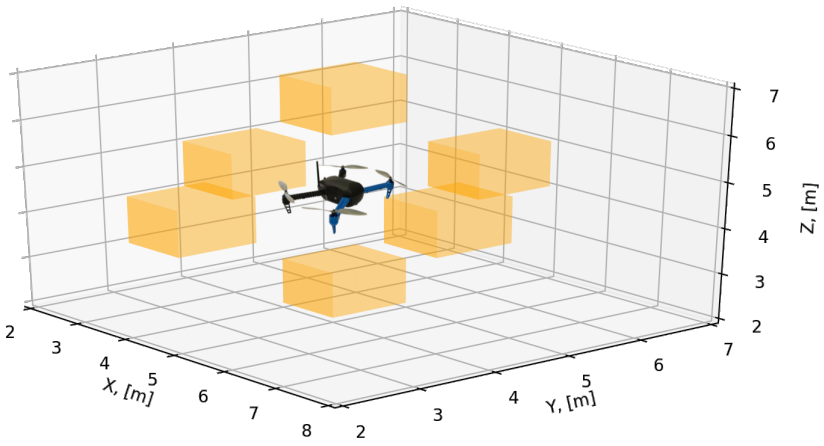


Figure III.6: Sensor's view directions.

Figure III.6 shows the sensor's directions, which is the same as the constraint movement of the drone, now the cubes can be seen as the grids where we extract information from the sensors. In other words,

the UAV movements can be described as a vector $movement = M_z+, M_z-, M_y+, M_y-, M_x+, M_x-$ and so is the sensors reading $sensors = S_z+, S_z-, S_y+, S_y-, S_x+, S_x-$. The last pieces of the dataset are the actual pose of the UAV and the target. Both has the shape of a three dimensional vector, represented by $pose = p_x, p_y, p_z$ and $target = t_x, t_y, t_z$. Table III.1 presents a slice of the dataset for one of these 30.000 simulated situations.

Table III.1: Slice of the dataset generated by Dijkstra's algorithm.

Pose			Target			Sensors						Movement					
p_x	p_y	p_z	t_x	t_y	t_z	S_z+	S_z-	S_y+	S_y-	S_x+	S_x-	M_z+	M_z-	M_y+	M_y-	M_x+	M_x-
11	14	1	13	1	23	0	0	1	0	0	0	0	0	0	1	0	0
11	13	1	13	1	23	0	0	0	0	0	0	0	0	0	1	0	0
11	12	1	13	1	23	1	0	0	0	0	0	0	0	0	1	0	0
11	11	1	13	1	23	1	0	0	0	0	0	0	0	0	1	0	0
11	10	1	13	1	23	1	0	0	0	0	0	0	0	0	1	0	0

The number of simulated scenarios were 30.000, which gives the total of 180.684 lines that represents all the decisions taken from the node algorithm in every grid the agent passed. All these data composes the dataset developed in this work (77). Its important to say that even if we have this large data, only a part of it were used for train and test the proposed Neural Network. The number of simulations that actually were used, will be discusses in chapter IV, since it plays a important role to avoid overfitting.

III.3 Neural Network

III.3.1 Proposed model for the neural network

The number of layers, neurons, batch size, the type of optimizer to use, metrics, loss function, learning rate, and regularization techniques are called hyperparameters. These parameters are some of the choices that the programmer needs to face when training NNs. In the past years, many attempts have been made in order to create techniques to optimize this choice, particle swarm optimization (78) (79), genetic algorithm (80) are examples of these techniques. This work applied the hyperparameters search manually using rules-of-thumb discussed by Hinton et al. (81) and Hsu et al. (82). In Chapter IV are presented the different results when changing some hyperparameters.

The Neural Network's proposed architecture is a fully connected feed-forward configuration with four layers, two are hidden, and the others are input and output ones. Figure III.7 illustrates this architecture. The input layer has eighteen neurons that are built based on the sensor readings, the target, and the local pose. The second one has one hundred twenty-eight neurons, the third has sixty-four, and the output six.

A Rectified Linear Unit (ReLU) function is used as an activation function for the two hidden layers. This function was introduced by (83), and it simply works as a zero threshold, for values bellow zero output is zero and for $x > 0$ it outputs a linear function.ReLU have been vastly applied together with softmax cross-entropy function to solve different classifications problems using NNs (84, 85, 86). Figure III.8 shows ReLU behavior for a specific interval.

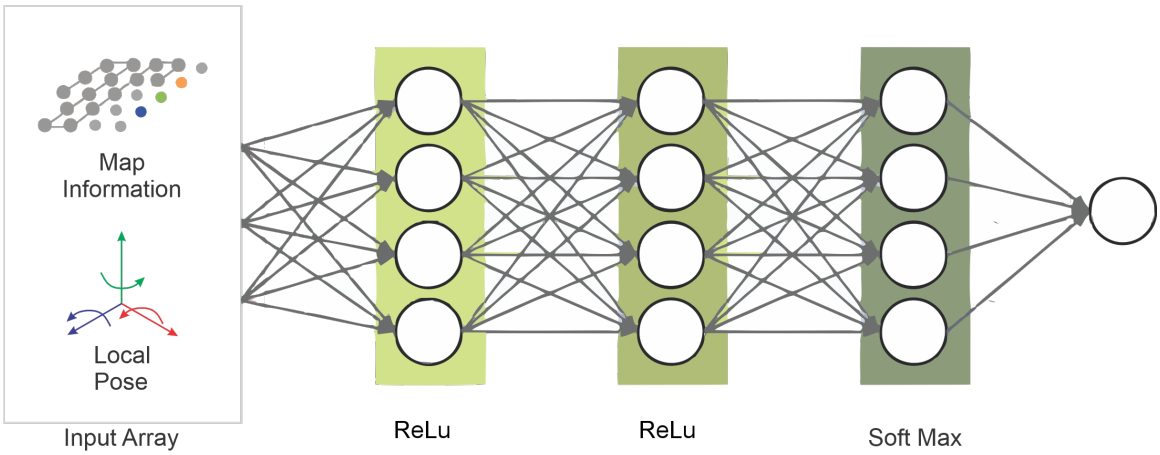


Figure III.7: Proposed architecture of neural network.

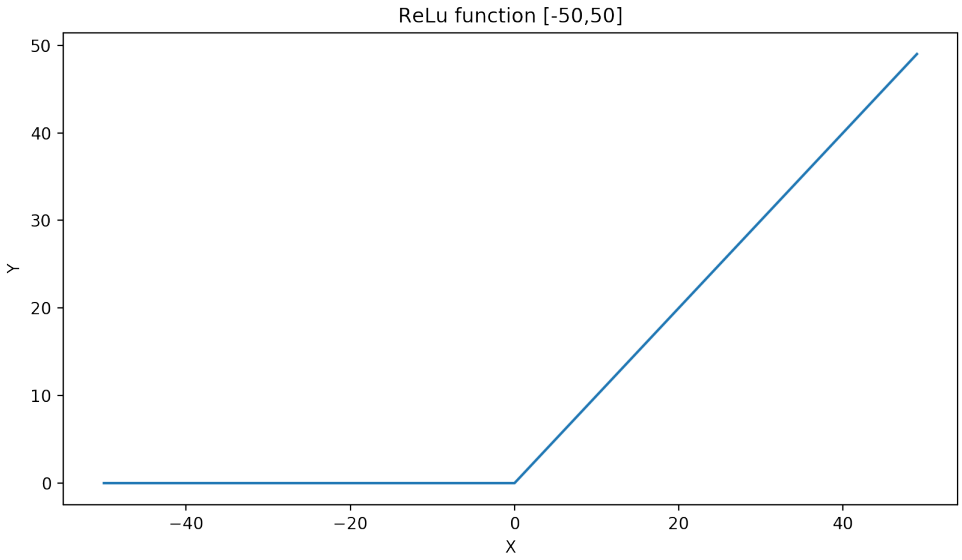


Figure III.8: Implementation of a ReLu function.

Softmax was applied on the output layer as an activation function, since the problem is tackled as a multi-class categorical problem. In other words, our NNs will return a probability of which is the best action (i.e., direction) that the UAV should go. For instance, suppose an input like Table III.2, based on this particular situation, the best possible output is to go up (i.e., M_z+) since it will reduce the distance from the source to the target and it is a free-collision path. When our NN faces this input, it will return the output showed in Table III.3. It is clearly that every possible action has a probability, M_z+ has 98% ,M_z- 0.2%,M_y+ 0.3%,M_y- 0.5% ,M_x+ 0.01% and M_x- has 0.09%. The action that the UAV will take is the one with higher probability.

The model uses Mean Absolute Error (MAE) function during the training phase as a cost function. As shown by (87) MAE is used as a robust loss function to train NNs under noise datasets in multi-classification problems. Although the dataset used in this work is free noise, and even if as shown

Table III.2: Neural network input example.

Pose			**Target**			**Sensors**					
p_x	p_y	p_z	t_x	t_y	t_z	S_z+	S_z-	S_y+	S_y-	S_x+	S_x-
0	0	0	0	0	3	0	1	0	1	0	0

Table III.3: Neural network output for the input of Table III.2.

Output					
M_z+	M_z-	M_y+	M_y-	M_x+	M_x-
0.98	0.002	0.003	0.005	0.001	0.009

by (88) MAE can increase difficulty during training in Chapter IV will be shown that the opposite occurred, MAE performs better than categorical cross-entropy based algorithms. At last, it was used Adam optimizer, that was first introduced by (89). The method was design to combine RMSProp (90) and AdaGrad (91) advantages, the first handling with sparses gradients and the second with on-line non-stationary settings.

III.3.2 Training the model

Estimating the accuracy of a model (classifier) induced by supervised learning algorithms is important not only to predict its future prediction accuracy, but also for choosing a classifier from a given set (model selection), or combining classifiers (92). In the light of this affirmation, we choose cross validation, as the method applied for measure the efficiency of the generated models, although its known that no accuracy estimation can be correct all the time (93, 94), cross-validation has low bias and low variance, when comparing with bootstrap (95) and other methods (96). In this work we sliced the dataset (77) into two parts, using a ratio of 70% for training and validation purposes and 30% for testing. Then we sliced again the training data into 10 folds, this division was stratified, in order that the folds had approximately the same labels proportion as the original dataset. This split can change randomly through each interaction which provides a better Monte-Carlo estimate to the complete cross-validation at an added cost (96). In this work we used a cross-validation method named n-fold cross-validation. The idea behind it, is to only divided data once and changes its function (validation and train) at each interaction III.9, this action reduces lots of time during the models evaluation, although when compared with complete k-fold cross-validation it is less precised.

Together with cross validation,an early-stopping function was applied. It worked like a callback function, that is evoked at the end of each epoch during the training phase.This function would interrupt the training process if the cost function presented worst results in a row. The parameter that stipulate the quantity of these results is called patience, and when correctly tuned optimizes the model accuracy, preventing underfitting and overfitting.

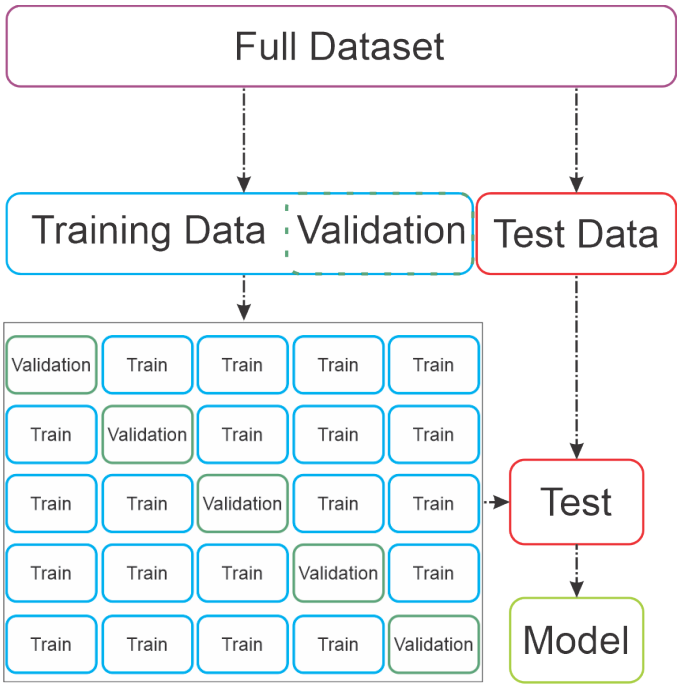


Figure III.9: N-fold Cross-Validation train method.

Chapter IV Results and Discussion

All the simulations are performed on a notebook with 8GB DDR3 RAM and a 2.7 GHz Core-i5-5200 processor. The operating system distro is a 64 bits Ubuntu 16.04 xenial with a desktop environment LXDE. All algorithms were implemented using Python 2.7.12. Figure IV.1 describes the processing steps of the proposed methodology. As can be seen, the UAV position and environment data from the depth camera are combined to produce a precise environment map. These maps will be updated during the inspection process. The algorithm processes this information to produce the UAV input vector. The input vector will contain basic information about the UAV location and its surroundings. The path planning can be applied from the input vector and generate the next position where the UAV should be located.

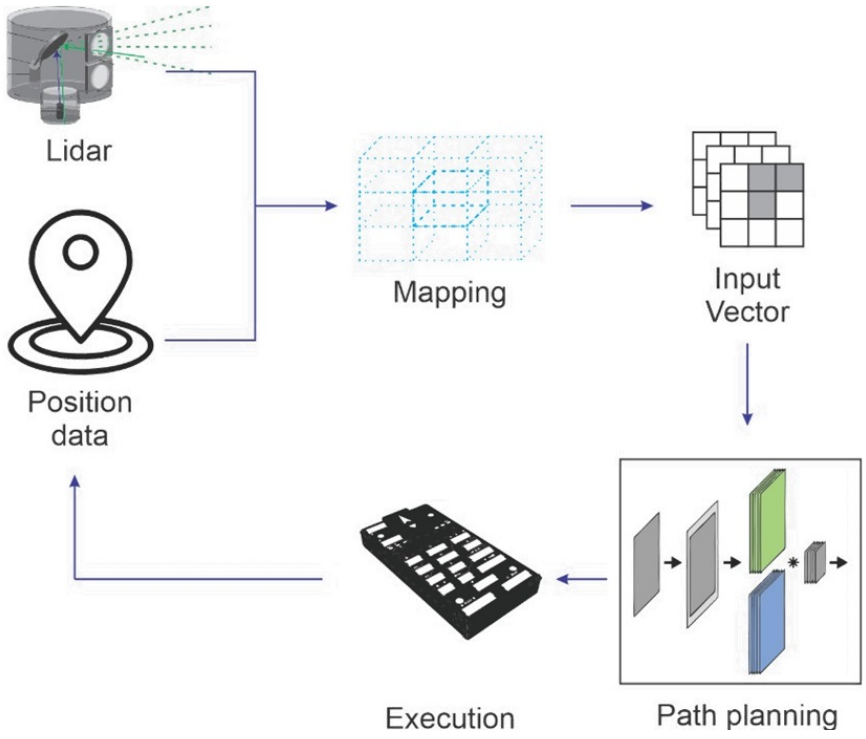


Figure IV.1: Infographic representation of the processing steps.

IV.1 Training Process

IV.1.1 Number of layers and neurons

Regarding the training process, the cross-validation training makes it possible to test different NNs models and gives trustful results, as discussed in Chapter III. The feedfoward NNs models are create using Keras (97) library and uses the Adam (89) optimizer with default learning rate $\alpha = 1103$, $\beta_1 = 0.9$, $\beta_2 = 0.999$,$\epsilon = 1108$, and no decay. Table IV.1 shows a example of a neural network structure based in this library.

Table IV.1: First of the three better NNs models generated using randomly initialization of hyperparameters

Batch size = 1000		
Layer (type)	Number of nodes	Activation
Dense	18	ReLU
Dense	128	ReLU
Batch Normalization	—	—
Dense	64	ReLU
Batch Normalization	—	—
Dense	6	Softmax

Table IV.2: Second of the three better NNs models generated using randomly initialization of hyperparameters

Batch size = 750		
Layer (type)	Number of nodes	Activation
Dense	8	ReLU
Batch Normalization	—	—
Dense	32	ReLU
Batch Normalization	—	—
Dense	16	ReLU
BatchNormalization	—	—
Dense	6	Softmax

Table IV.3: Third of the three better NNs models generated using randomly initialization of hyperparameters

Batch size = 900		
Layer (type)	Number of nodes	Activation
Dense	69	ReLU
Batch Normalization	—	—
Dense	190	ReLU
Batch Normalization	—	—
Dense	166	ReLU
BatchNormalization	—	—
Dense	6	Softmax

Using rules of thumb method (81, 82), discussed in chapter II, we sample sixty randomly models

using sets of different hyperparameters. In this sense, tables IV.1,IV.2 and IV.3 shows the structure of the three NNs models, that have better performance.In figure IV.2 shows the training and validation of the models. These results give to us a hint of what is gonna be the final model structure, since its possible to see that the models are learning through the epochs. In other other words, the loss is getting smaller and the accuracy bigger as much as the epochs are passing.

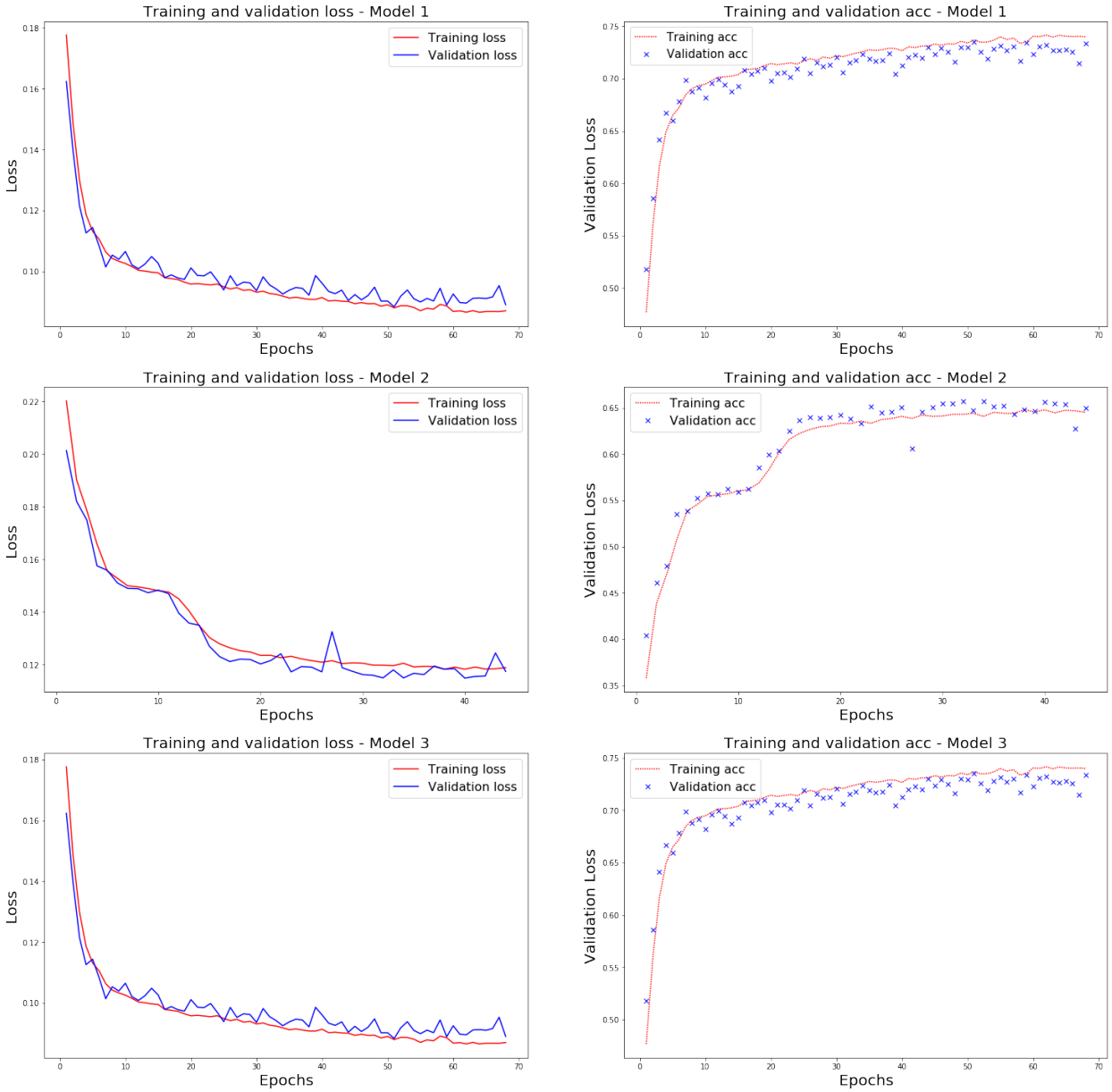


Figure IV.2: Random models.

Another piece of information that this analysis gave to us, is that models with three layers and using the batch size between 800 and 1000 units gives great results. In order to better understand this data and compare themselves performance we can analyze the tables IV.4, IV.5 and IV.6, that show the models performance through each fold during the cross-validation train. These tables show that depending on the the fold, we can expected better, or not, results since they take different pieces of the

dataset to train and validate data. In model 1 table IV.4 for example,the folds nine and ten, take better performance than the others, the same happen in fold 5,10,4,5 in model 2 and model 3 respectively.

Table IV.4: Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.1.

Fold	Loss	Accuracy(x100%)	Mse
1	0.08226080950746933	0.7538	0.07981030692011118
2	0.08726617128600676	0.7389166666666667	0.08468605899761121
3	0.08583215553611517	0.7428666666666667	0.08324645183881124
4	0.08456029720405737	0.7465666666666667	0.08195482400208712
5	0.08481584132810434	0.7457833333333334	0.0823706709186236
6	0.08488099710891645	0.74555	0.08264241950201491
7	0.08292948759098848	0.7514166666666666	0.08079301984558503
8	0.08197796434213718	0.7542666666666666	0.0799420260116458
9	0.08219373828570048	0.7535166666666666	0.08020280018250148
10	0.08012960558285316	0.7598833333333334	0.0782468729664882

Table IV.5: Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.2

Fold	Loss	Accuracy(x100%)	Mse
1	0.09772442011386157	0.7078333333333333	0.09449852476914723
2	0.09596467327028513	0.7126833333333333	0.09255503638436396
3	0.09761417149752379	0.7078166666666666	0.0944758033161362
4	0.09728292270600795	0.7085	0.09403711899966001
5	0.0956315401809911	0.7135833333333333	0.0925789721495239
6	0.09998196794937055	0.7010333333333333	0.0969394253542026
7	0.09738239938697467	0.7081833333333334	0.09418266956160853
8	0.09844471259216468	0.70495	0.09574538338979086
9	0.0988040993998448	0.7041333333333334	0.09593132083018621
10	0.09480653842786949	0.7162666666666667	0.09211008421977361

Table IV.6: Training accuracies, mse and losses per fold in 10-fold using the NN model describe in table IV.3.

Fold	Loss	Accuracy(x100%)	Mse
1	0.07161953251883388	0.7866166666666666	0.06773633756370594
2	0.07976075526839123	0.7622166666666667	0.07472513816976571
3	0.07930026321808498	0.7629666666666667	0.07471375742306312
4	0.07908299187272787	0.7638	0.07461225947588682
5	0.07839935580392679	0.7655	0.07423253265048067
6	0.0779168124333024	0.7672	0.07360960410485665
7	0.07822249878781537	0.7661	0.07428377850343629
8	0.07695374436924855	0.7694333333333333	0.07322869099751114
9	0.07324521374106407	0.7809333333333334	0.06977380297270915
10	0.07177296830018362	0.7853	0.06852749990820885

An even better way to understand the results of these models is creating a box plot, where the graph presents the mean and the distribution of values. Figure IV.3 displays the values contained in tables

IV.4, IV.5 ,IV.6 on the box plot shape. Looking at these graphs it possible to see that clearly model 3 has the best performance, since the mean value of Minimum square error (MSE) and loss are the lowest and the accuracy is the bigger when compared with the other two NNs models. Since model 3 achieve a satisfactory performance, the experiments in the next sections will processed using it in the simulations.

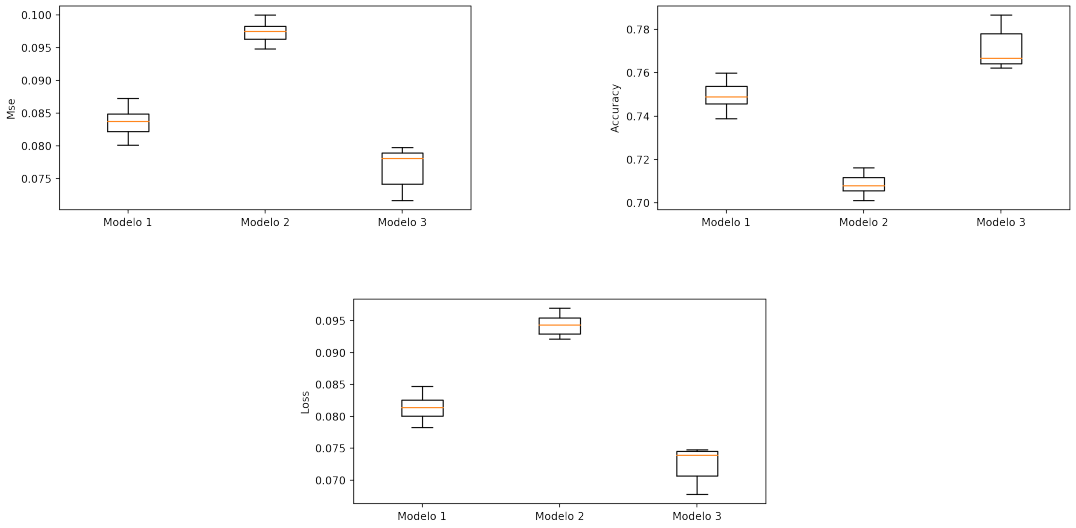


Figure IV.3: Training results for Model 1,2 and 3.

IV.2 Simulation Results

The whole idea of this work, is to test the proposed neural network model in different scenarios that approach to a real world experience, ant test its effectiveness. In other to do that, we splited the simulations into two groups, the first using Python 2.7 and the other running the model in a flight simulator. In the first case the environment is created like a PRM and has no physical proprieties, like wind, gravity, the motors and model inertia and etc. The only thing that matters is the dimensions of the uav, the objects size and the length of the path. On the other hand using the flight simulator all this physical proprieties are applied to the model, which gives to us a much better idea of what would be the effectiveness of the model in a real scenario. Also in other to compare and quantify how good is our strategy, we compare the results founded in the Python simulations with a established path planning algorithm discussed in chapter II, the RRT algorithm.

IV.2.1 Proposed methodology simulated only with Python

Three different worlds were used during these tests, Figure IV.4 shows all of them. These worlds were build thinking in different levels of challenges, world 1 can be seen as an easy task, with only four objects, world 2 was more complex with extra interest obstacles and world 3 is the most difficulty, since it has the complexity of world 2 but the agent was trapped inside a box.

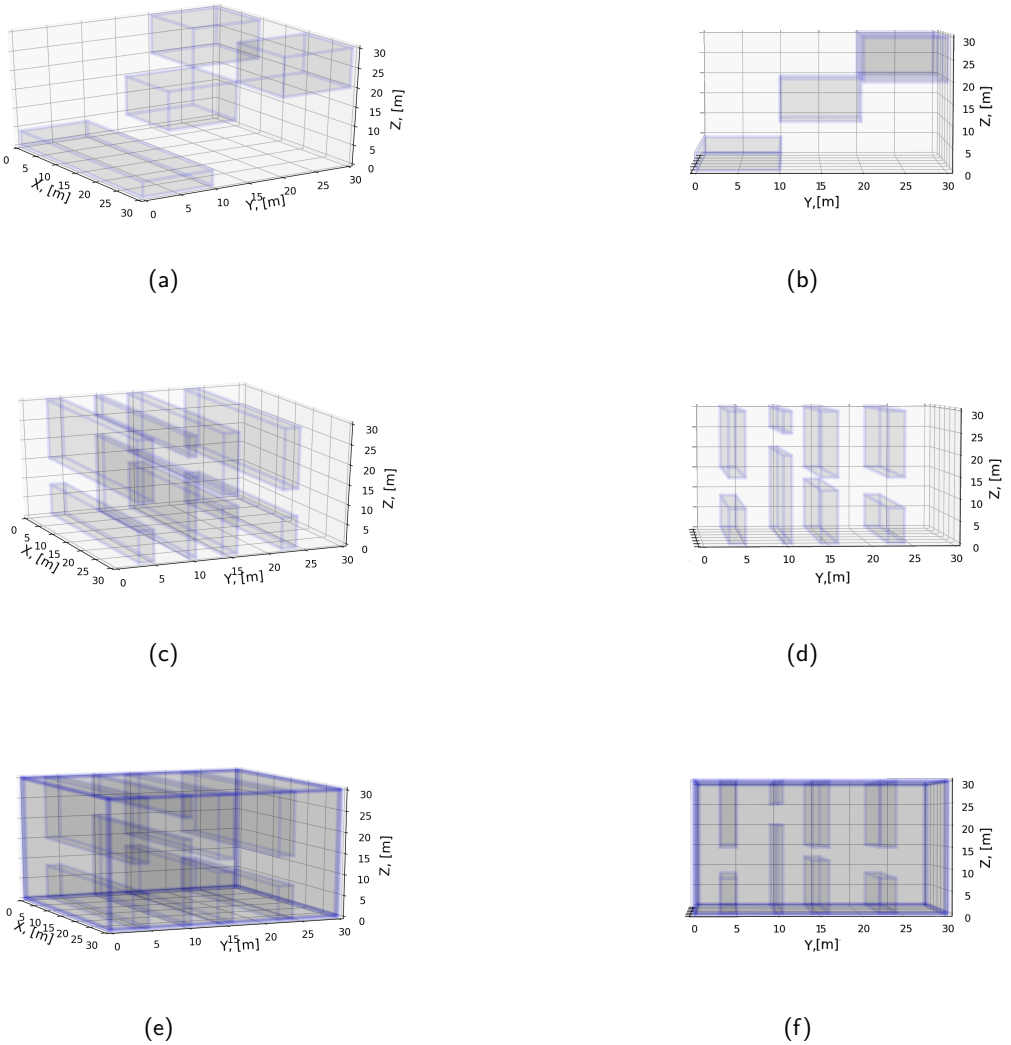


Figure IV.4: Simulated worlds to create training dataset. (a) Isometric view from simulated world 1. (b) Lateral view from simulated world 1. (c) Isometric view from simulated world 2. (d) Lateral view from simulated world 2. (e) Isometric view from simulated world 3. (f) Lateral view from simulated world 3.

In each world was randomly generated twenty pairs of start and target points, and then paths are created with RRT and the proposed Neural Network Model. The metrics used for comparing paths generated from them were the time spent finding the path and its distance. Figure IV.5 shows the results of this simulations.

Table IV.7: Performance evaluation statistics

Case	Mean value	STD
World 1	38.85	37
World 2	56.75	71
World 3	34.27	65
Total	42.62	59

To analyze every single result in Figure IV.5 may be difficult. So we subtracted the path sizes for

every single case in Figure IV.5 and plotted these differences as an histogram as shown in Figure IV.6. Note in the Figure that the distribution center is positive, i.e.the proposed method generated smaller path sizes. The worst scenario were, as expected, in world 3, it happens because when during the training phase, a similar situation wasn't presented in the dataset. This information was also compiled in Table IV.7. Note that despite the deviation from the mean in most cases the method was produced a smaller path then the RTT. The data in Figure IV.5 we also estimated that the proposed method generated a path on average 47% smaller than RTT.

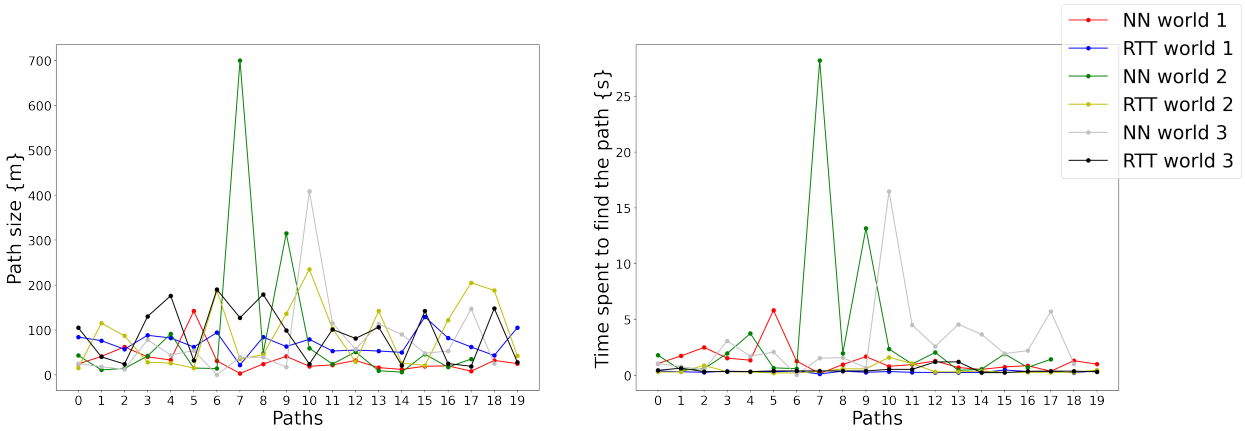


Figure IV.5: Simulations results for static targets.

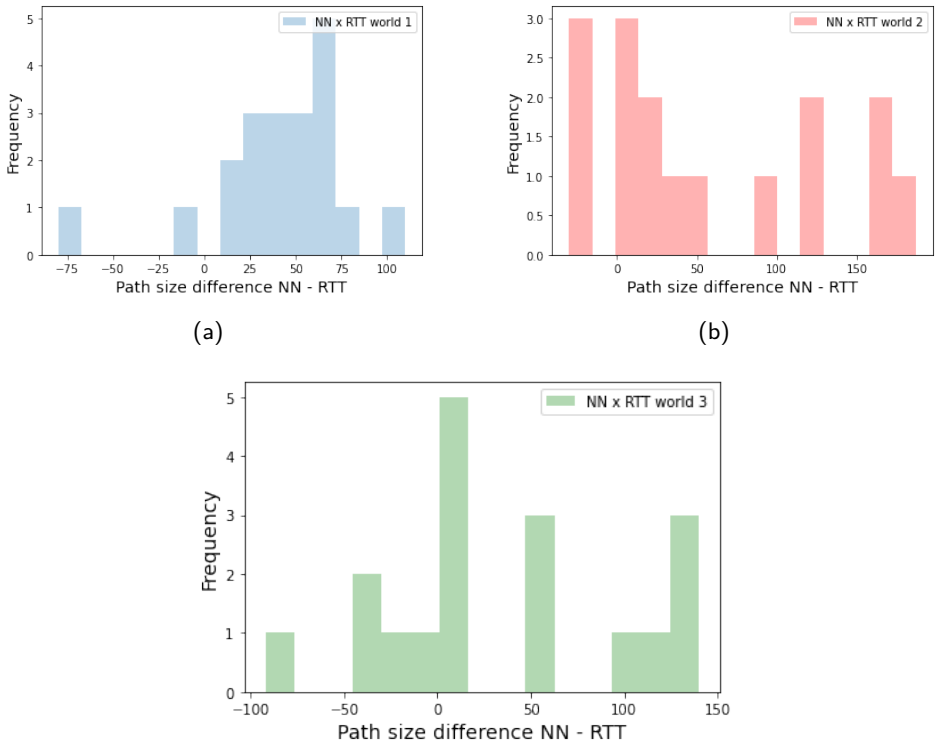


Figure IV.6: Histograms based on IV.5

Some paths generated in world 1 are shown in Figure IV.7, where the green dot is the start point,

the red one is the target, and the blue line the path generated.

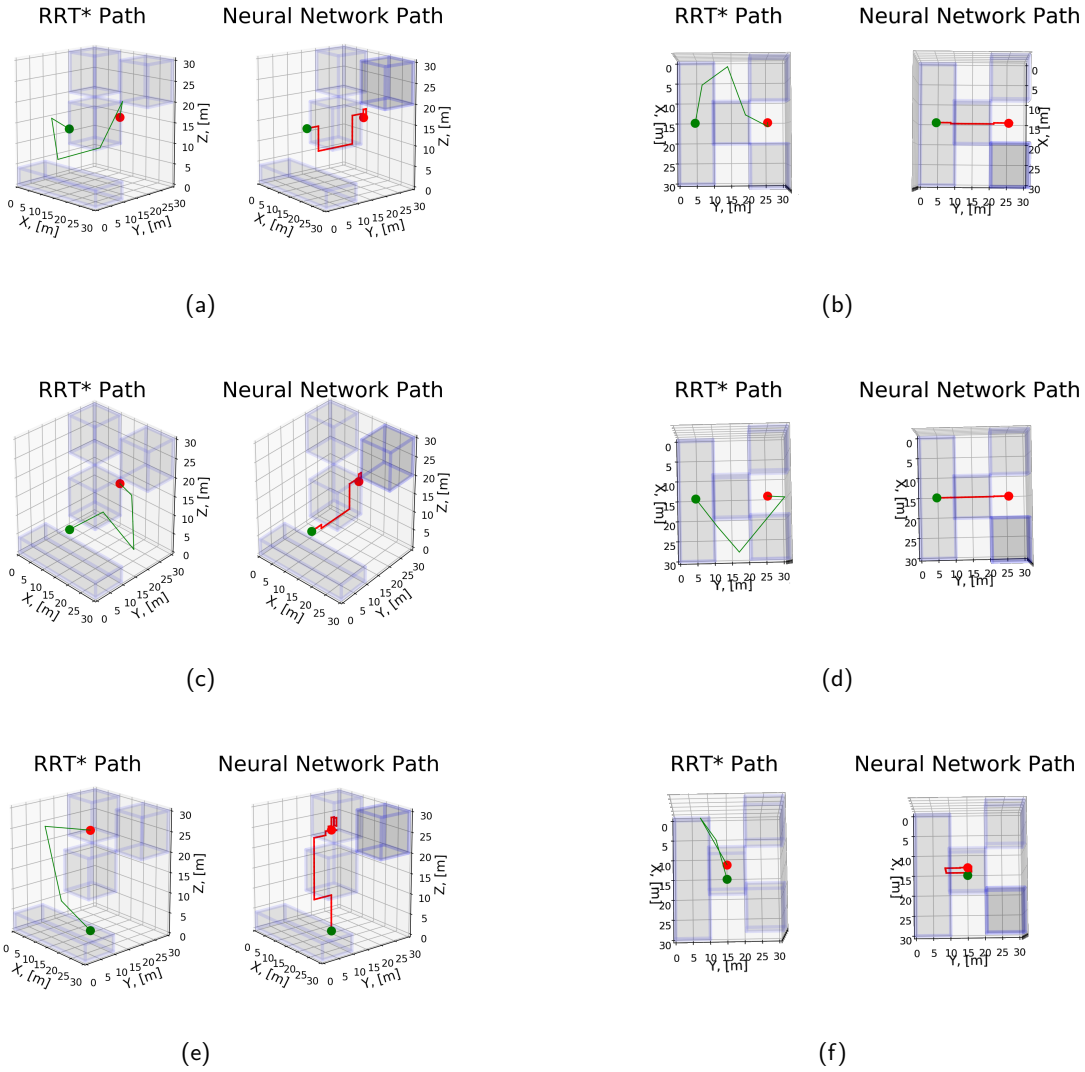


Figure IV.7: Isometric and top view of paths generated by RRT* and Neural networks in world 1. (a)Isometric view for path with start point = [15,15,10] and target = [25,15,15]. (b) Lateral view for path with start point = [15,15,10] and target = [25,15,15]. (c) Isometric view for path with start point = [15,10,10] and target = [15,25,20] (d) Lateral view for path with start point = [15,10,10] and target = [15,25,20] (e) Isometric view for path with start point = [20,10,5] and target = [15,20,25] (f) Lateral view for path with start point = [20,10,5] and target = [15,20,25].

IV.2.1.1 Moving targets

Is Sometimes interesting while doing industrial inspections to lock a moving target and follow it, like when making a ship hull inspection while it is on the water or looking for cracks on a moving pipeline. With that motivation, we simulated twenty other scenarios in each world, but this time the target moves aleatory one meter in any direction every one second. Like the last section, we compared the paths generated by the RRT algorithm and our proposed NNs model. Figure IV.8 shows these simulations results. And as discussed in the last section, we subtracted the path sizes for every single case in figure

IV.8 and plotted these differences as a histogram as shown in IV.9. Like in static objects, in moving targets we have positive mean values table IV.8, which shows that the proposed model in most of the simulated scenarios return better results than RRT. From these results we also estimated that the neural network model produces 37% smaller paths than RRT.

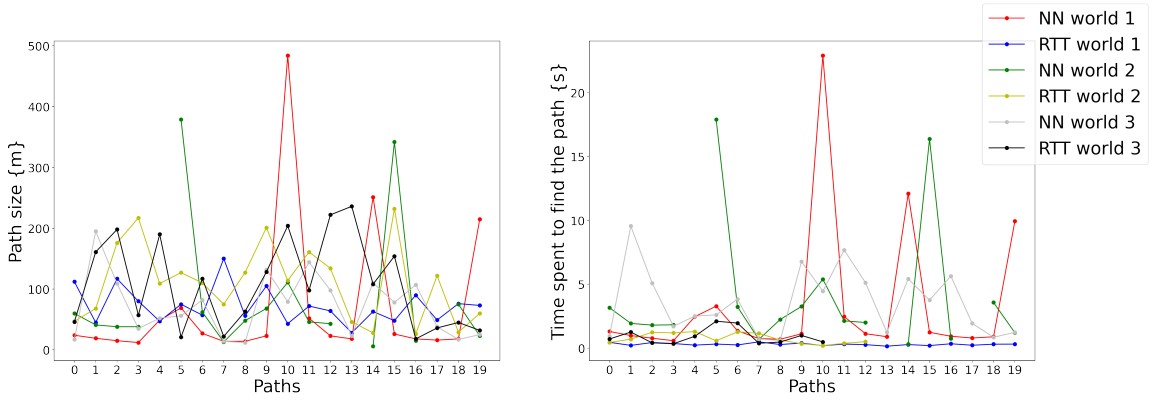
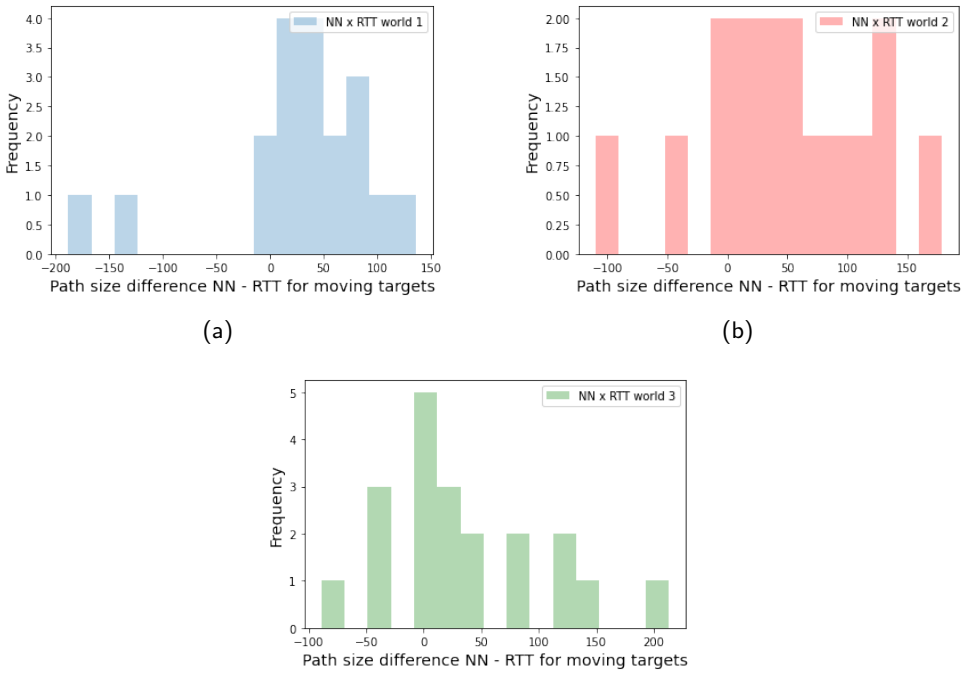


Figure IV.8: Simulations results for moving targets.



(a) (b)

(c)

Figure IV.9: Histograms based on IV.8

Table IV.8: Performance evaluation statistics for moving targets

Case	Mean value	STD
World 1	26.52	74.62
World 2	48.5	71.64
World 3	36.65	70.97
Total	36.6	72.97

IV.2.1.2 Kidnapped problem

Another interesting problem facing by autonomous robots is the kidnapped problem. It consists of when the robot loses his pose for a period of time and gets back it in another place. That is very usual when handling with UAVs that uses Global Positioning System (GPS). GPS tend to have signal attenuation/loss depending on its contact with the satellites, weather, and buildings can create this difficulty. In light of this reason, the proposed NNs model and RRT were used to find a path between a source and a target. Different from the last sections, this time, the simulated UAV lost its pose at the fifth algorithm iteration and then regained its pose in an aleatory place in the world. The simulations were generated in the same three words used in the last sections, and the results are shown in Figure IV.10

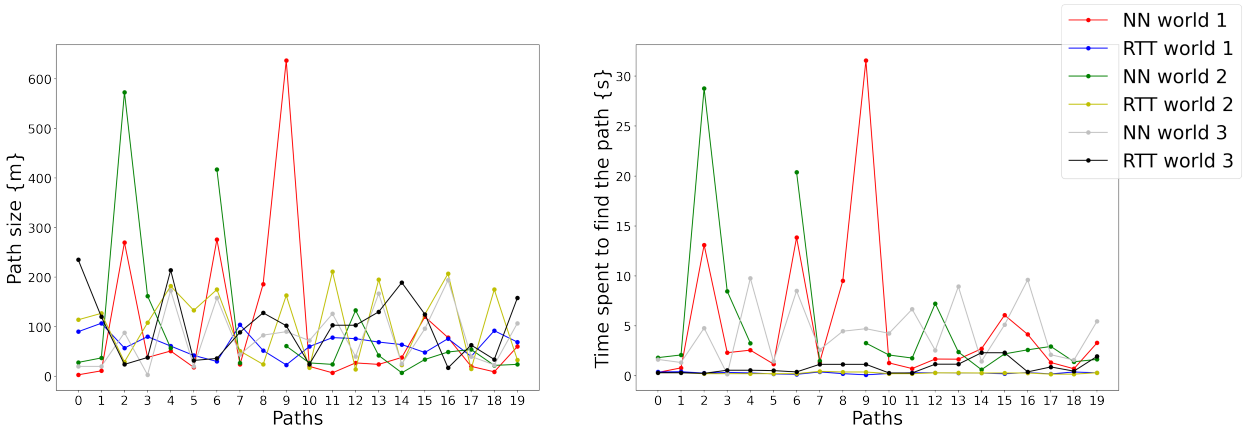


Figure IV.10: Simulations results for kidnapped problem

Using Figure IV.10, were created Table **??** and the histogram in figure IV.11. By looking at this results, again the proposed model is better in average, it is confirmed observing the mean values in Table **??** and the way the histograms were centered in positive values. The neural networks performs in terms of size path 27% better than the RRT in the simulated scenarios.

Table IV.9: Performance evaluation statistics for kidnapped problem

Case	Mean value	STD
World 1	0.78	95.62
World 2	60.75	85.55
World 3	18.75	85.41
Total	24.76	92.35

IV.2.1.3 Dynamic environments

Since the motivation of this work is to create a NNs model, in other to use it as pathfinding for UAVs during industrial inspections, its needed to face that real-world application are usually non-static. A simulation in world three is created to reflect that behavior. Figure IV.12 shows world 3 and how its

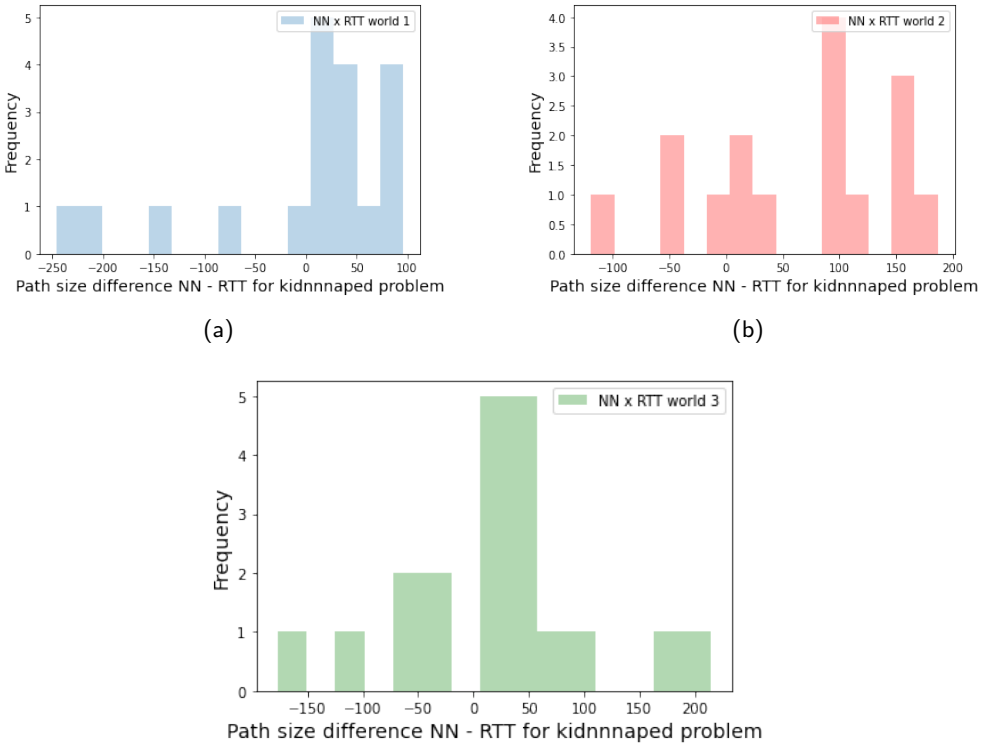


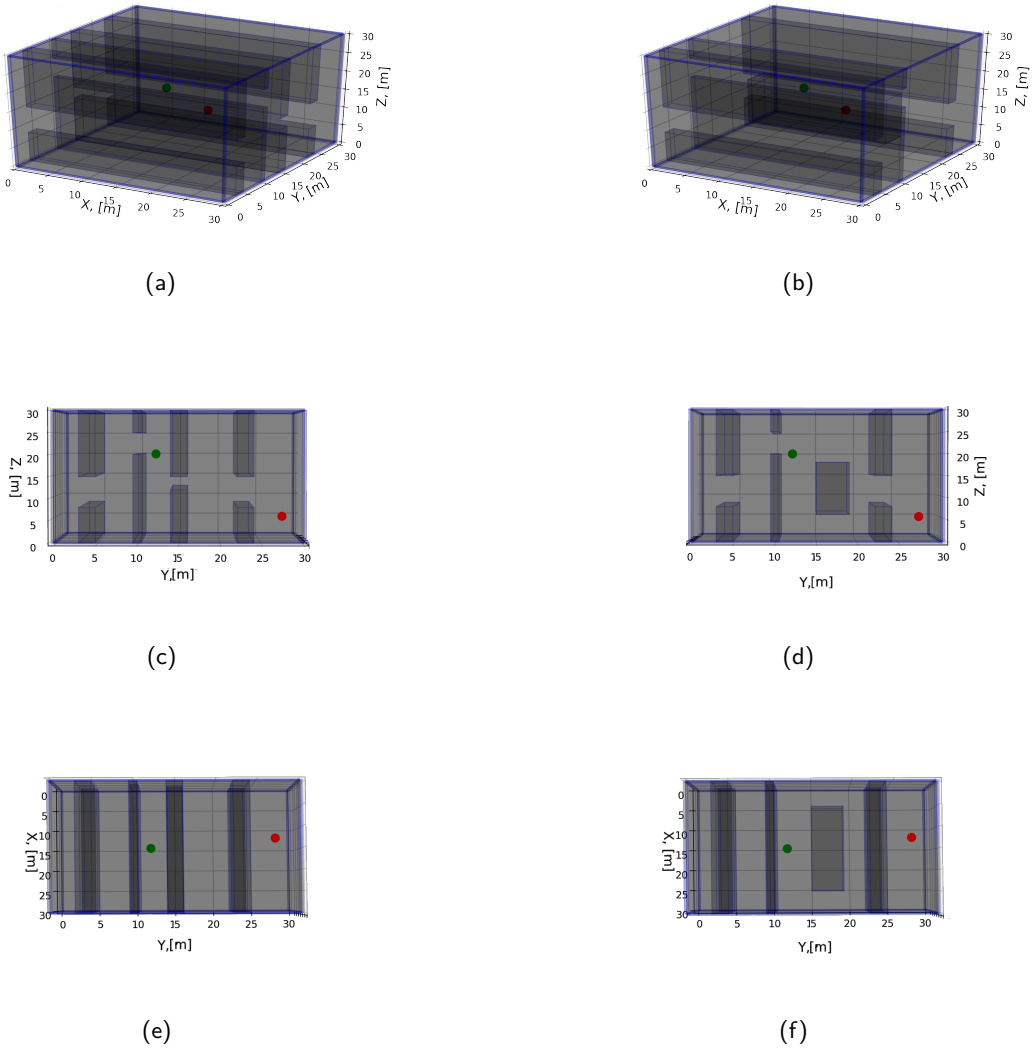
Figure IV.11

changes during simulations. The changing world will be presented to the UAV in every five iterations. The results for the proposed NNs model is presented in Figure IV.13 and for RRT in Figure IV.14. It is important to say that this is a naive implementation. Simulating and predicting dynamic objects is way more complex, although the results show how our proposed model NNs has no problem when facing this situation and guarantees, in this case, a free collision path. On the other hand, the path generated by the RRT algorithm can't guarantee the UAV safety since it's an offline method, and when the drone actually goes on this path, it could be changed.

IV.2.2 Proposed methodology tested in a flight simulator

For this experiment, the robotic system in the Gazebo simulation environment was used. The simulation using ROS was built in a world created with the software Gazebo, as shown in Figure IV.16. For the tests, the UAV model used was the 3DR IRIS, presented in Figure IV.15. In other to test the efficiency of the proposed strategy developed in this work the UAV must complete a curve through the barrage to perform an inspection in a specific section on the hydro power plant.

Figure IV.17 shows the UAV during the hydro-power plant inspection, figure IV.18 how the operator will visualize the information acquired from it and figure IV.19 the path taken until one of the inspection points.

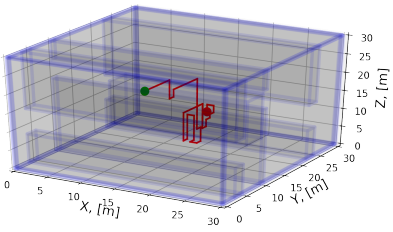


(a) (b)

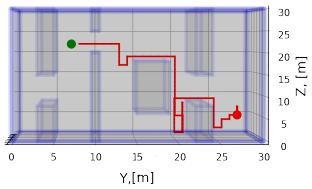
(c) (d)

(e) (f)

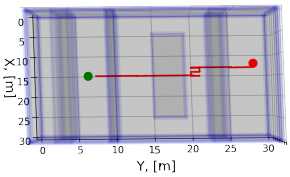
Figure IV.12: Simulation using Neural Network to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3.



(a)

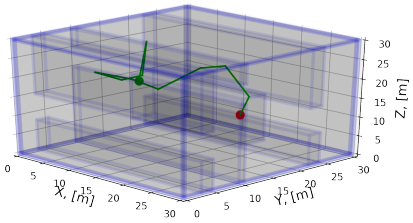


(b)

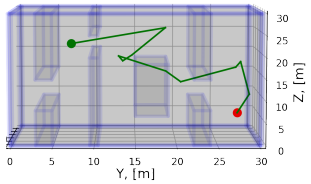


(c)

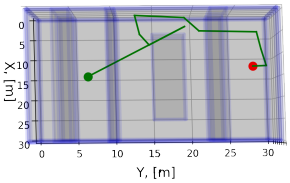
Figure IV.13: Simulation using Neural Network to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3.



(a)



(b)



(c)

Figure IV.14: Simulation using rapidly exploring random tree to find path in a dynamic world, source = [15,12,20] (green dot), target = [12,28,5] (red dot). (a) Isometric view of world 3. (b) Isometric view of change of world 3. (c) Top view of world 3. (d) Top view of of change of world 3. (e) Lateral view of world 3. (f) Lateral view of change world 3.



Figure IV.15: UAV model in Gazebo/ROS.

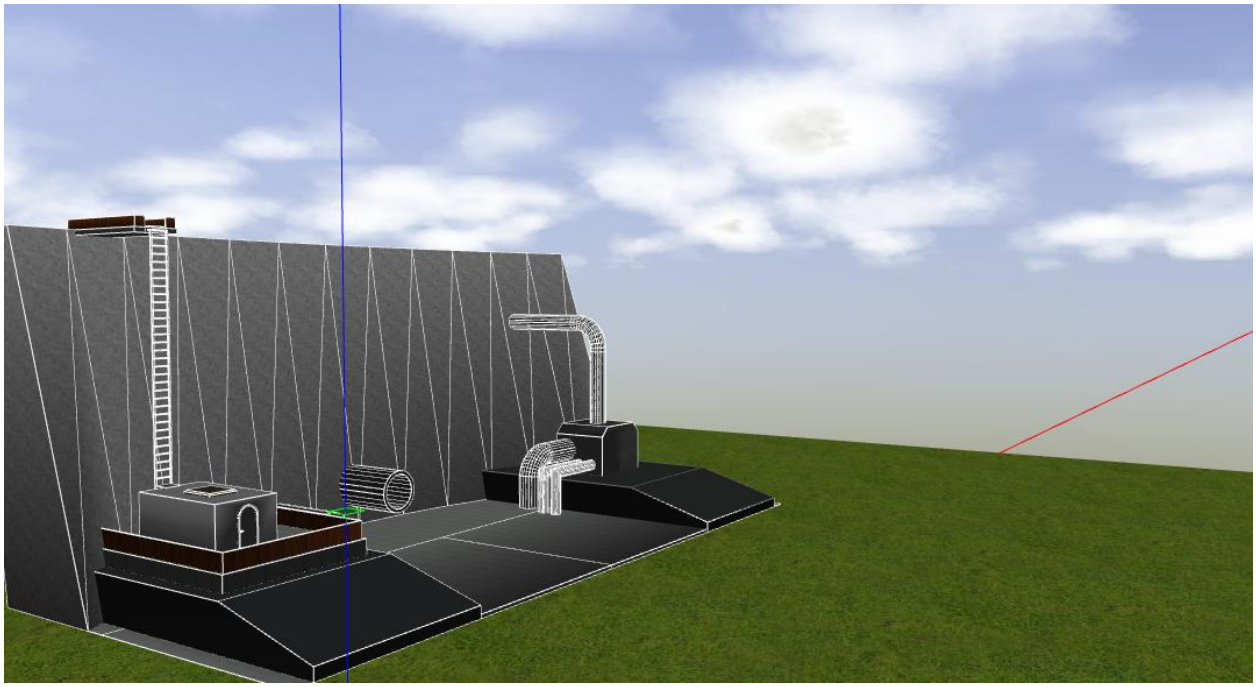


Figure IV.16: Gazebo world used in the simulations.

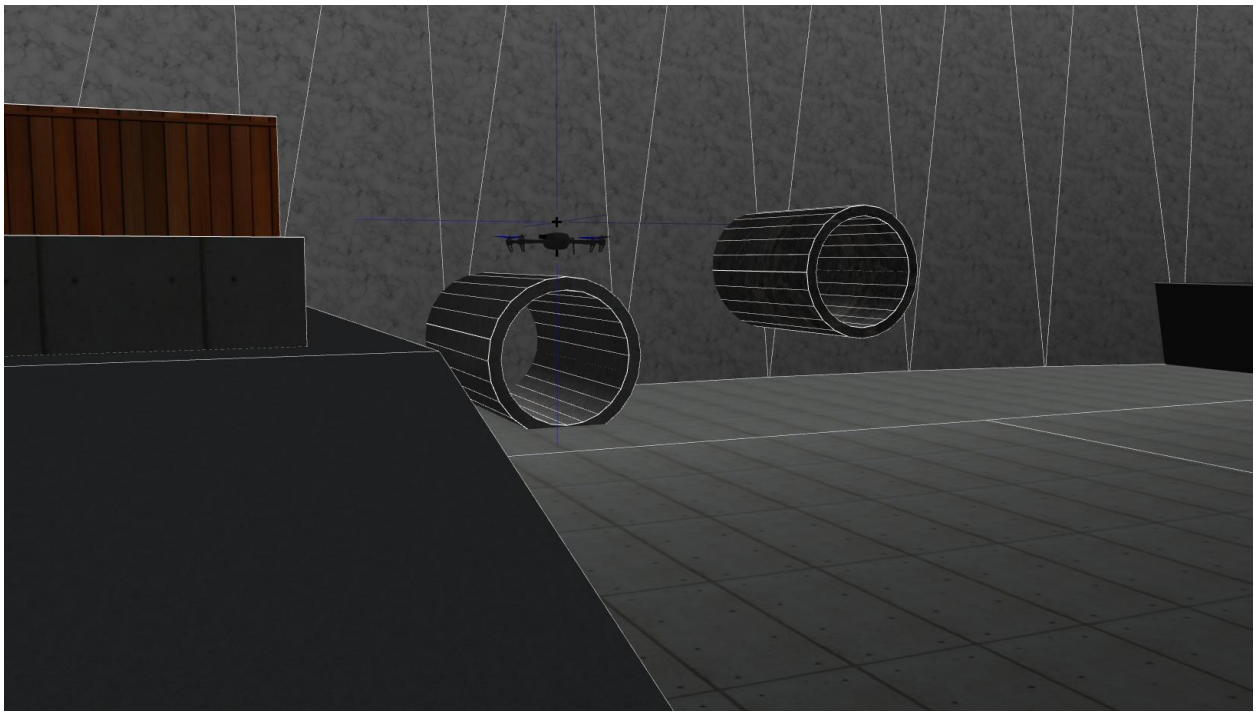


Figure IV.17: UAV performing inspection in hydro-power plant in gazebo simulated world.

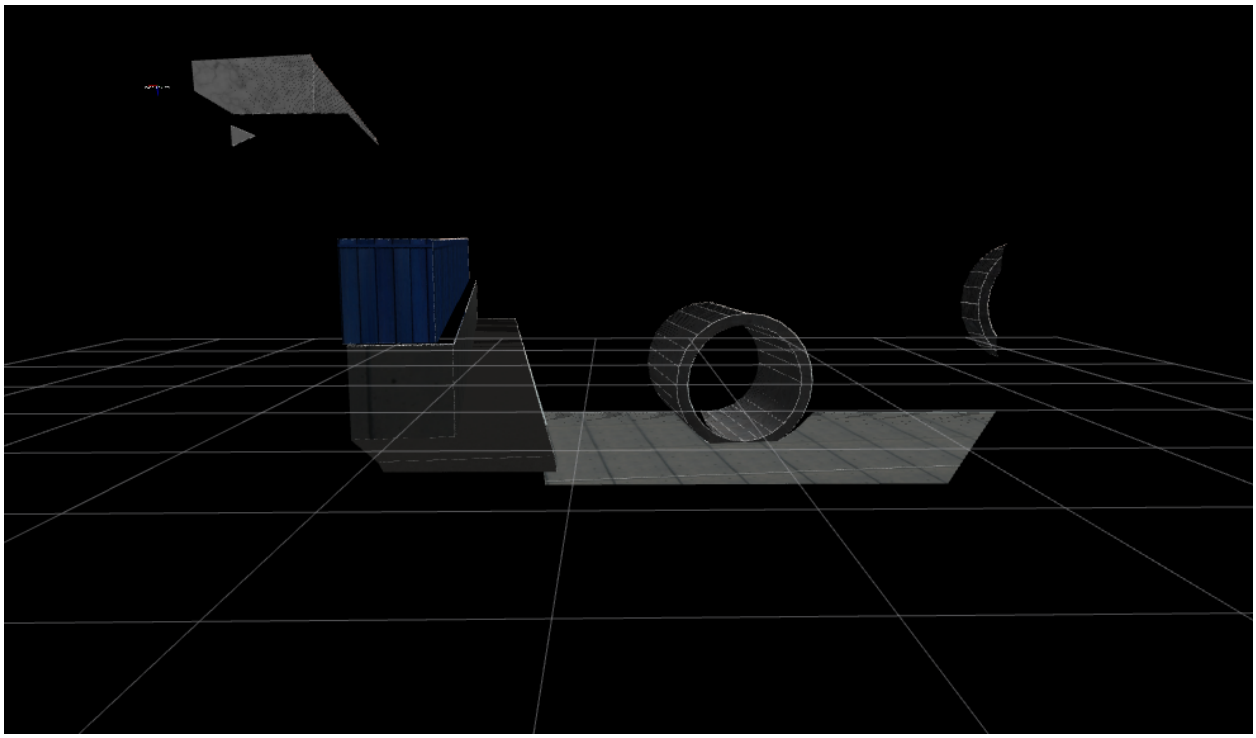


Figure IV.18: UAV vision during the pipe inspection in gazebo simulated world.

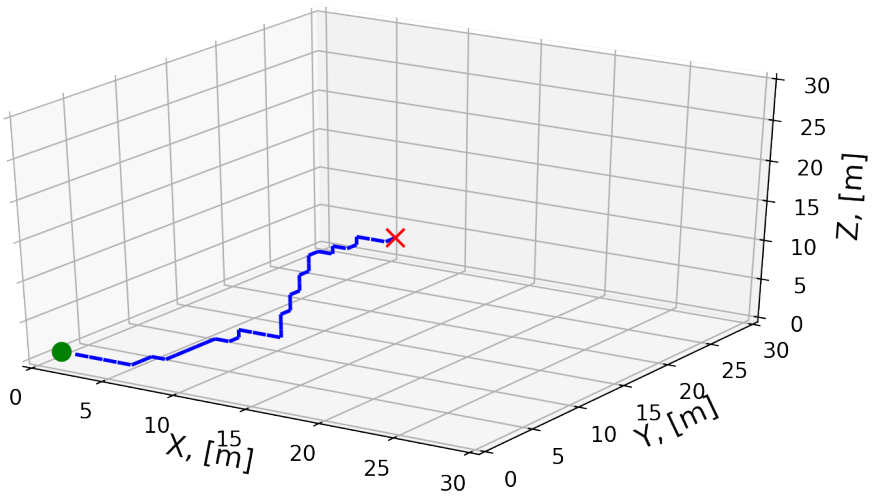


Figure IV.19: Path generated from Neural Network to perform the inspection in gazebo simulated world.

Chapter V Conclusions and Future Work

V.0.1 Conclusions

UAVs have been used in many fields of industry for aerial inspection, such as hydropower plants, chemical industry, oil and gas pumps, telecommunications towers, among others. This is explained by their versatility, the flexibility of flying, and the fact that they can easily reach places that human beings can't or are too risky for them. A vital role for UAV inspection is path planning, responsible for positioning the UAV from one position to another safely, avoiding obstacles, and taking the shortest path from all the available options.

Over the years, many works were developed for optimal path planning, being classified as heuristic, meta-heuristic, discrete, among others. As a drawback of these conventional approaches, their computational time depends on the environment's complexity and scale. Thus, a promising solution to overcome these traditional algorithms' bottlenecks is using deep neural network architectures, such as feed-forward neural networks. This kind of architecture can also learn to plan efficiently under various conditions.

Regarding 3D path planning, in recent years, some works have been addressing the use of NNs to this issue. Therefore, this research proposed using a neural network structure to perform real-time path planning in unknown environments. In this study, the authors developed a strategy to generate the initial database used for training and validate the results in a simulated environment using Gazebo and ROS. In this sense, different scenarios were created using random characteristics, and Dijkstra's algorithm was used to generate the proper optimization that will feed the path database. Thus, global solutions will be available to perform training on the intelligent method. The database was split into training, validation, and evaluation. Therefore, the deep learning method was trained using a cross-validation method, an Adam optimizer algorithm, a categorical cross-entropy as the loss function, and binary accuracy as the metric to evaluate the training performance.

V.0.2 Future Works

In terms of evaluation, this research work opens up several future possibilities to develop sophisticated path planning capable of work in global and local environments. Since Neural Networks are a field that have been extensively studied through the last years, there will be always room for improvement the performance of it. New approaches, tools and better strategies to understand and optimize the training phase are being developed every day. The dataset can also be much more complex with other important information like the battery of the UAV, a bigger amount of sensor and communication from others vehicles. Another improvement that can be done in dataset is the algorithm used to feed it. In this work the authors applied Dijistrik's algorithm to find the minimum path, but in order to use that, a number of constrains in the environment must be set. Using more complex algorithms can avoid this problem and still find better solutions, not only for minimum path, but for managing the UAV resources, like fuel or safety. At last, after all the discussion and results presented in this work, a series of experiments with a real UAV performing visual inspections in different structures can be done.

Bibliography

1 YANG, L. et al. Survey of robot 3d path planning algorithms. *Journal of Control Science and Engineering*, Hindawi, v. 2016, 2016. Citado 4 vezes nas páginas ix, 18, 19 e 21.

2 FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, Wiley Online Library, v. 7, n. 2, p. 179–188, 1936. Citado 3 vezes nas páginas ix, 25 e 26.

3 BIUNDINI, I. Z. et al. Coverage path planning optimization for slopes and dams inspection. In: SILVA, M. F. et al. (Ed.). *Robot 2019: Fourth Iberian Robotics Conference*. Cham: Springer International Publishing, 2020. p. 513–523. Citado 2 vezes nas páginas 13 e 15.

4 GAGO, J. et al. Uavs challenge to assess water stress for sustainable agriculture. *Agricultural water management*, Elsevier, v. 153, p. 9–19, 2015. Citado na página 13.

5 SHAKHATREH, H. et al. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, IEEE, v. 7, p. 48572–48634, 2019. Citado na página 13.

6 MELO, A. G. et al. 3d correspondence and point projection method for structures deformation analysis. *IEEE Access*, IEEE, v. 8, p. 177823–177836, 2020. Citado 2 vezes nas páginas 13 e 14.

7 PINTO, M. F. et al. Arcog: An aerial robotics cognitive architecture. *Robotica*, Cambridge University Press, p. 1–20, 2020. Citado na página 13.

8 STEK, T. D. Drones over mediterranean landscapes. the potential of small uav's (drones) for site detection and heritage management in archaeological survey projects: A case study from le pianelle in the tappino valley, molise (italy). *Journal of Cultural Heritage*, Elsevier, v. 22, p. 1066–1071, 2016. Citado na página 13.

9 PINTO, M. F. et al. A robotic cognitive architecture for slope and dam inspections. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 20, n. 16, p. 4579, 2020. Citado na página 13.

10 GAUTAM, S. A.; VERMA, N. Path planning for unmanned aerial vehicle based on genetic algorithm & artificial neural network in 3d. In: IEEE. *2014 International Conference on Data Mining and Intelligent Computing (ICDMIC)*. [S.l.], 2014. p. 1–5. Citado 2 vezes nas páginas 13 e 28.

11 BIUNDINI, I. Z. et al. Coverage path planning optimization for slopes and dams inspection. In: SILVA, M. F. et al. (Ed.). *Robot 2019: Fourth Iberian Robotics Conference*. Cham: Springer International Publishing, 2020. p. 513–523. Citado na página 13.

12 HUANG, Z.; CHEN, C.; PAN, M. Multiobjective uav path planning for emergency information collection and transmission. *IEEE Internet of Things Journal*, IEEE, v. 7, n. 8, p. 6993–7009, 2020. Citado na página 13.

13 PINTO, M. F. et al. A framework for analyzing fog-cloud computing cooperation applied to information processing of uavs. *Wireless Communications and Mobile Computing*, Hindawi, v. 2019, 2019. Citado na página 13.

14 CABREIRA, T. M.; BRISOLARA, L. B.; JR, P. R. F. Survey on coverage path planning with unmanned aerial vehicles. *Drones*, Multidisciplinary Digital Publishing Institute, v. 3, n. 1, p. 4, 2019. Citado na página 13.

15 CHOI, Y. et al. Energy-constrained multi-uav coverage path planning for an aerial imagery mission using column generation. *Journal of Intelligent & Robotic Systems*, Springer, v. 97, n. 1, p. 125–139, 2020. Citado na página 13.

16 SHANG, Z.; BRADLEY, J.; SHEN, Z. A co-optimal coverage path planning method for aerial scanning of complex structures. *Expert Systems with Applications*, Elsevier, p. 113535, 2020. Citado na página 13.

17 GLORIEUX, E.; FRANCIOSA, P.; CEGLAREK, D. Coverage path planning with targetted viewpoint sampling for robotic free-form surface inspection. *Robotics and Computer-Integrated Manufacturing*, Elsevier, v. 61, p. 101843, 2020. Citado na página 14.

18 DORIGO, M.; BIRATTARI, M.; STUTZLE, T. Ant colony optimization. *IEEE computational intelligence magazine*, IEEE, v. 1, n. 4, p. 28–39, 2006. Citado na página 14.

19 HOFFMAN, K. L. et al. Traveling salesman problem. *Encyclopedia of operations research and management science*, Springer New York, v. 1, p. 1573–1578, 2013. Citado na página 14.

20 JUNEJA, S. S. et al. Travelling salesman problem optimization using genetic algorithm. In: IEEE. *2019 Amity International Conference on Artificial Intelligence (AICAI)*. [S.l.], 2019. p. 264–268. Citado 2 vezes nas páginas 14 e 27.

21 HADIA, S. K. et al. Solving city routing issue with particle swarm optimization. *International Journal of Computer Applications*, Citeseer, v. 47, n. 15, 2012. Citado na página 14.

22 KANEZAKI, A.; NITTA, J.; SASAKI, Y. Goselo: Goal-directed obstacle and self-location map for robot navigation using reactive neural networks. *IEEE Robotics and Automation Letters*, IEEE, v. 3, n. 2, p. 696–703, 2017. Citado na página 14.

23 CHEN, W.-N. et al. A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on evolutionary computation*, IEEE, v. 14, n. 2, p. 278–300, 2009. Citado na página 14.

24 LIM, C.-W. et al. A path planning algorithm for surveillance uavs with timing mission constrains. In: IEEE. *ICCAS 2010*. [S.l.], 2010. p. 2371–2375. Citado na página 15.

25 VIANA, F. X. et al. Aerial image instance segmentation through synthetic data using deep learning. Citado na página 15.

26 CAI, Z. et al. A unified multi-scale deep convolutional neural network for fast object detection. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 354–370. Citado na página 15.

27 SARGANO, A. B. et al. Human action recognition using transfer learning with deep representations. In: IEEE. *2017 International joint conference on neural networks (IJCNN)*. [S.l.], 2017. p. 463–469. Citado na página 15.

28 KIM, W.-S. et al. Path detection for autonomous traveling in orchards using patch-based cnn. *Computers and Electronics in Agriculture*, Elsevier, v. 175, p. 105620, 2020. Citado na página 15.

29 TERASAWA, R. et al. 3d-cnn based heuristic guided task-space planner for faster motion planning. In: IEEE. *2020 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2020. p. 9548–9554. Citado na página 15.

30 REHDER, E. et al. Cooperative motion planning for non-holonomic agents with value iteration networks. *arXiv preprint arXiv:1709.05273*, 2017. Citado na página 15.

31 TAMAR, A. et al. Value iteration networks. *Advances in neural information processing systems*, v. 29, p. 2154–2162, 2016. Citado na página 15.

32 KHAN, A. et al. Memory augmented control networks. *arXiv preprint arXiv:1709.05706*, 2017. Citado na página 15.

33 GUPTA, S. et al. Cognitive mapping and planning for visual navigation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2017. p. 2616–2625. Citado na página 15.

34 OKADA, M.; RIGAZIO, L.; AOSHIMA, T. Path integral networks: End-to-end differentiable optimal control. *arXiv preprint arXiv:1706.09597*, 2017. Citado na página 15.

35 AGGARWAL, S.; KUMAR, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, Elsevier, v. 149, p. 270–299, 2020. Citado na página 15.

36 PU, X. et al. 3d path planning for a robot based on improved ant colony algorithm. *Evolutionary Intelligence*, Springer, p. 1–11, 2020. Citado na página 15.

37 YANG, K.; SUKKARIEH, S. Real-time continuous curvature path planning of uavs in cluttered environments. In: IEEE. *2008 5th International Symposium on Mechatronics and Its Applications*. [S.l.], 2008. p. 1–6. Citado 2 vezes nas páginas 15 e 27.

38 DEBNATH, S. K. et al. Different cell decomposition path planning methods for unmanned air vehicles-a review. In: SPRINGER. *Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019*. [S.l.], 2020. p. 99–111. Citado na página 15.

39 YAN, F.; LIU, Y.-S.; XIAO, J.-Z. Path planning in complex 3d environments using a probabilistic roadmap method. *International Journal of Automation and computing*, Springer, v. 10, n. 6, p. 525–533, 2013. Citado 2 vezes nas páginas 15 e 27.

40 SCHØLER, F.; COUR-HARBO, A. la; BISGAARD, M. Generating configuration spaces and visibility graphs from a geometric workspace for uav path planning. *Autonomous Robots, 2012*, Citeseer, 2012. Citado na página 15.

41 WU, K. et al. Tdpp-net: Achieving three-dimensional path planning via a deep neural network architecture. *Neurocomputing*, Elsevier, v. 357, p. 151–162, 2019. Citado 2 vezes nas páginas 15 e 28.

42 WU, K. et al. Achieving real-time path planning in unknown environments through deep neural networks. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2020. Citado na página 15.

43 KARAMAN, S.; FRAZZOLI, E. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, Sage Publications Sage UK: London, England, v. 30, n. 7, p. 846–894, 2011. Citado na página 18.

44 SCHØLER, F. 3d path planning for autonomous aerial vehicles in constrained spaces. Section of Automation & Control, Department of Electronic Systems, Aalborg , 2012. Citado na página 18.

45 KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. In: *Autonomous robot vehicles*. [S.l.]: Springer, 1986. p. 396–404. Citado 2 vezes nas páginas 18 e 19.

46 KAVRAKI, L. E. et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, IEEE, v. 12, n. 4, p. 566–580, 1996. Citado 2 vezes nas páginas 18 e 19.

47 ISWANTO, A. M.; WAHYUNGGORO, O.; CAHYADI, A. I. Artificial potential field algorithm implementation for quadrotor path planning. *Int. J. Adv. Comput. Sci. Appl*, v. 10, n. 8, p. 575–585, 2019. Citado na página 19.

48 BOHLIN, R.; KAVRAKI, L. E. Path planning using lazy prm. In: IEEE. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. [S.l.], 2000. v. 1, p. 521–528. Citado na página 20.

49 MEYER-DELIUS, D.; BEINHOFER, M.; BURGARD, W. Occupancy grid models for robot mapping in changing environments. In: *Twenty-Sixth AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2012. Citado na página 21.

50 DIJKSTRA, E. W. et al. A note on two problems in connexion with graphs. *Numerische mathematik*, v. 1, n. 1, p. 269–271, 1959. Citado 2 vezes nas páginas 22 e 27.

51 HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, IEEE, v. 4, n. 2, p. 100–107, 1968. Citado 2 vezes nas páginas 22 e 27.

52 KOENIG, S.; LIKHACHEV, M. Improved fast replanning for robot navigation in unknown terrain. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. [S.l.: s.n.], 2002. v. 1, p. 968–975 vol.1. Citado 2 vezes nas páginas 22 e 27.

53 STENTZ, A. Optimal and efficient path planning for partially known environments. In: *Intelligent unmanned ground vehicles*. [S.l.]: Springer, 1997. p. 203–220. Citado 2 vezes nas páginas 22 e 27.

54 KOENIG, S.; LIKHACHEV, M. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics*, IEEE, v. 21, n. 3, p. 354–363, 2005. Citado na página 22.

55 LUO, M.; HOU, X.; YANG, J. Surface optimal path planning using an extended dijkstra algorithm. *IEEE Access*, IEEE, v. 8, p. 147827–147838, 2020. Citado na página 22.

56 ZHU, D.-D.; SUN, J.-Q. A new algorithm based on dijkstra for vehicle path planning considering intersection attribute. *IEEE Access*, IEEE, v. 9, p. 19761–19775, 2021. Citado na página 22.

57 LIU, L.-s. et al. Path planning for smart car based on dijkstra algorithm and dynamic window approach. *Wireless Communications and Mobile Computing*, Hindawi, v. 2021, 2021. Citado na página 22.

58 HAYKIN, S. *Neural networks and learning machines, 3/E*. [S.l.]: Pearson Education India, 2010. Citado 2 vezes nas páginas 23 e 24.

59 ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 23.

60 MITCHELL, T. M. et al. Machine learning. McGraw-hill New York, 1997. Citado na página 23.

61 GOODFELLOW, I. et al. *Deep learning*. [S.l.]: MIT press Cambridge, 2016. v. 1. Citado na página 25.

62 SCHØLER, F.; COUR-HARBO, A. la; BISGAARD, M. Generating configuration spaces and visibility graphs from a geometric workspace for uav path planning. In: CITESEER. *Proceedings of the AIAA Guidance, Navigation, and Control Conference, Portland, OR, USA*. [S.l.], 2011. p. 8–11. Citado na página 27.

63 CARSTEN, J.; FERGUSON, D.; STENTZ, A. 3d field d: Improved path planning and replanning in three dimensions. In: IEEE. *2006 IEEE/RSJ international conference on intelligent robots and systems*. [S.l.], 2006. p. 3381–3386. Citado na página 27.

64 AMATO, N. M. et al. Obprm: An obstacle-based prm for 3d workspaces. In: *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*. [S.l.: s.n.], 1998. p. 155–168. Citado na página 27.

65 CHEN, X.; ZHANG, J. The three-dimension path planning of uav based on improved artificial potential field in dynamic environment. In: IEEE. *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*. [S.l.], 2013. v. 2, p. 144–147. Citado na página 27.

66 WANG, Z.; LI, H.; ZHANG, X. Construction waste recycling robot for nails and screws: Computer vision technology and neural network approach. *Automation in Construction*, Elsevier, v. 97, p. 220–228, 2019. Citado na página 27.

67 NI, J.; YANG, S. X. Bioinspired neural network for real-time cooperative hunting by multirobots in unknown environments. *IEEE Transactions on Neural Networks*, IEEE, v. 22, n. 12, p. 2062–2077, 2011. Citado na página 27.

68 LUO, C. et al. Neural-dynamics-driven complete area coverage navigation through cooperation of multiple mobile robots. *IEEE Transactions on Industrial Electronics*, IEEE, v. 64, n. 1, p. 750–760, 2016. Citado na página 27.

69 GLASIUS, R.; KOMODA, A.; GIELEN, S. C. Neural network dynamics for path planning and obstacle avoidance. *Neural Networks*, Elsevier, v. 8, n. 1, p. 125–133, 1995. Citado na página 27.

70 HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 79, n. 8, p. 2554–2558, 1982. Citado na página 27.

71 HORN, J.; GEIGER, B.; SCHMIDT, E. Use of neural network approximation in multiple-unmanned aerial vehicle trajectory optimization. In: *AIAA guidance, navigation, and control conference*. [S.l.: s.n.], 2009. p. 6103. Citado na página 28.

72 WANG, N. et al. A hybrid neural network method for uav attack route integrated planning. In: SPRINGER. *International Symposium on Neural Networks*. [S.l.], 2009. p. 226–235. Citado na página 28.

73 CUI, Z.; WANG, Y. Uav path planning based on multi-layer reinforcement learning technique. *IEEE Access*, IEEE, v. 9, p. 59486–59497, 2021. Citado na página 28.

74 SUNG, I.; CHOI, B.; NIELSEN, P. On the training of a neural network for online path planning with offline path planning algorithms. *International Journal of Information Management*, Elsevier, v. 57, p. 102142, 2021. Citado na página 28.

75 LEI, X.; ZHANG, Z.; DONG, P. Dynamic path planning of unknown environment based on deep reinforcement learning. *Journal of Robotics*, Hindawi, v. 2018, 2018. Citado na página 28.

76 PAN, Y.; YANG, Y.; LI, W. A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-uav. *IEEE Access*, IEEE, v. 9, p. 7994–8005, 2021. Citado na página 28.

77 CASTRO, G. *Dijkstra 3d path*. IEEE Dataport, 2021. Disponível em: <https://dx.doi.org/10-.21227/5re6-ep10>. Citado 2 vezes nas páginas 33 e 35.

78 LIN, S.-W. et al. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, Elsevier, v. 35, n. 4, p. 1817–1824, 2008. Citado na página 33.

79 MEISSNER, M.; SCHMUKER, M.; SCHNEIDER, G. Optimized particle swarm optimization (opso) and its application to artificial neural network training. *BMC bioinformatics*, Springer, v. 7, n. 1, p. 1–11, 2006. Citado na página 33.

80 TSAI, J.-T.; CHOU, J.-H.; LIU, T.-K. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *IEEE Transactions on Neural Networks*, IEEE, v. 17, n. 1, p. 69–80, 2006. Citado na página 33.

81 HINTON, G. E. A practical guide to training restricted boltzmann machines. In: *Neural networks: Tricks of the trade*. [S.l.]: Springer, 2012. p. 599–619. Citado 2 vezes nas páginas 33 e 38.

82 HSU, C.-W. et al. *A practical guide to support vector classification*. [S.l.]: Taipei, Taiwan, 2003. Citado 2 vezes nas páginas 33 e 38.

83 HAHNLOSER, R. H. et al. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, Nature Publishing Group, v. 405, n. 6789, p. 947–951, 2000. Citado na página 33.

84 YANG, Z. et al. Hierarchical attention networks for document classification. In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. [S.l.: s.n.], 2016. p. 1480–1489. Citado na página 33.

85 CHOROWSKI, J. et al. Attention-based models for speech recognition. *arXiv preprint arXiv:1506.07503*, 2015. Citado na página 33.

86 KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, v. 25, p. 1097–1105, 2012. Citado na página 33.

87 GHOSH, A.; KUMAR, H.; SASTRY, P. Robust loss functions under label noise for deep neural networks. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2017. v. 31, n. 1. Citado na página 34.

88 ZHANG, Z.; SABUNCU, M. R. Generalized cross entropy loss for training deep neural networks with noisy labels. In: *32nd Conference on Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2018. Citado na página 35.

89 KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Citado 2 vezes nas páginas 35 e 38.

90 TIELEMAN, T.; HINTON, G. et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, v. 4, n. 2, p. 26–31, 2012. Citado na página 35.

91 DUCHI, J.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, v. 12, n. 7, 2011. Citado na página 35.

92 WOLPERT, D. H. Stacked generalization. *Neural networks*, Elsevier, v. 5, n. 2, p. 241–259, 1992. Citado na página 35.

93 WOLPERT, L. *The unnatural nature of science*. [S.l.]: Harvard University Press, 1994. Citado na página 35.

94 SCHAFFER, C. A conservation law for generalization performance. In: *Machine Learning Proceedings 1994*. [S.l.]: Elsevier, 1994. p. 259–265. Citado na página 35.

95 TIBSHIRANI, R. J.; EFRON, B. An introduction to the bootstrap. *Monographs on statistics and applied probability*, Chapman and Hall New York, v. 57, p. 1–436, 1993. Citado na página 35.

96 KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: MONTREAL, CANADA. *Ijcai*. [S.l.], 1995. v. 14, n. 2, p. 1137–1145. Citado na página 35.

97 CHOLLET, F. et al. *keras*. 2015. Citado na página 38.