PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

Ministry of Higher Education and Scientific Research

UNIVERSITY OF ECHAHID HAMMA LAKHDAR EL OUED

Faculty of exact sciences

Department: Computer science

Speciality: Artificial intelligence and distributed system

Design and realization of a Drone for optimizing the trajectory between two points

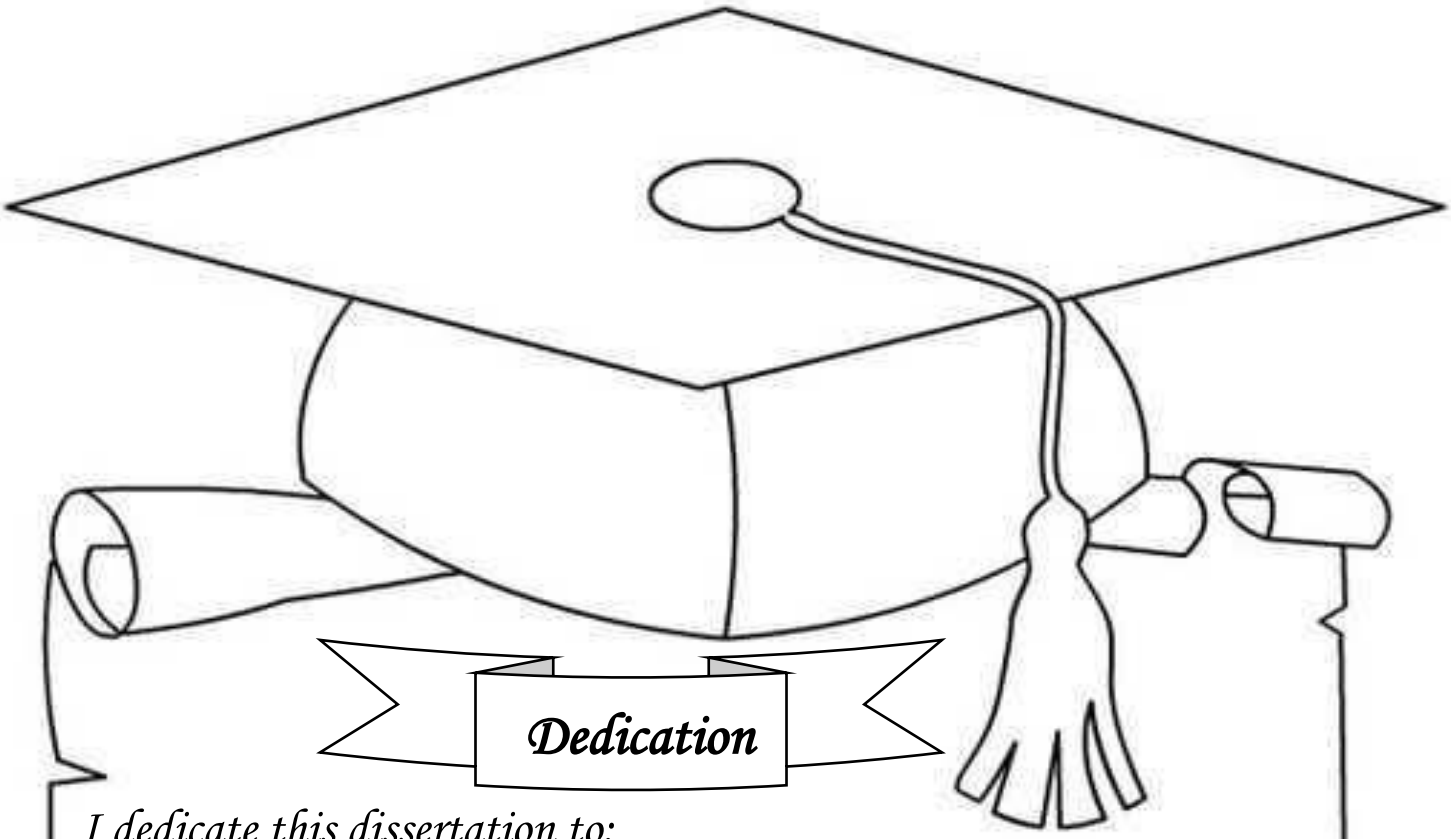
Presented by :

- ✓ MEDAOUI Rachid
- ✓ SADALLAH Younes

Supervised by:

Dr. LEJDEL Brahim

2019/2020



Dedication

I dedicate this dissertation to:

*My **mother** and my **sister** may Allah have mercy on their souls*

My father may Allah protect him

*My brothers (**Mounir, Mohamed, Oussama** and **Kamal**) whose have never left my side.*

*My dear sister, the sunshine of our family **Yasmine***

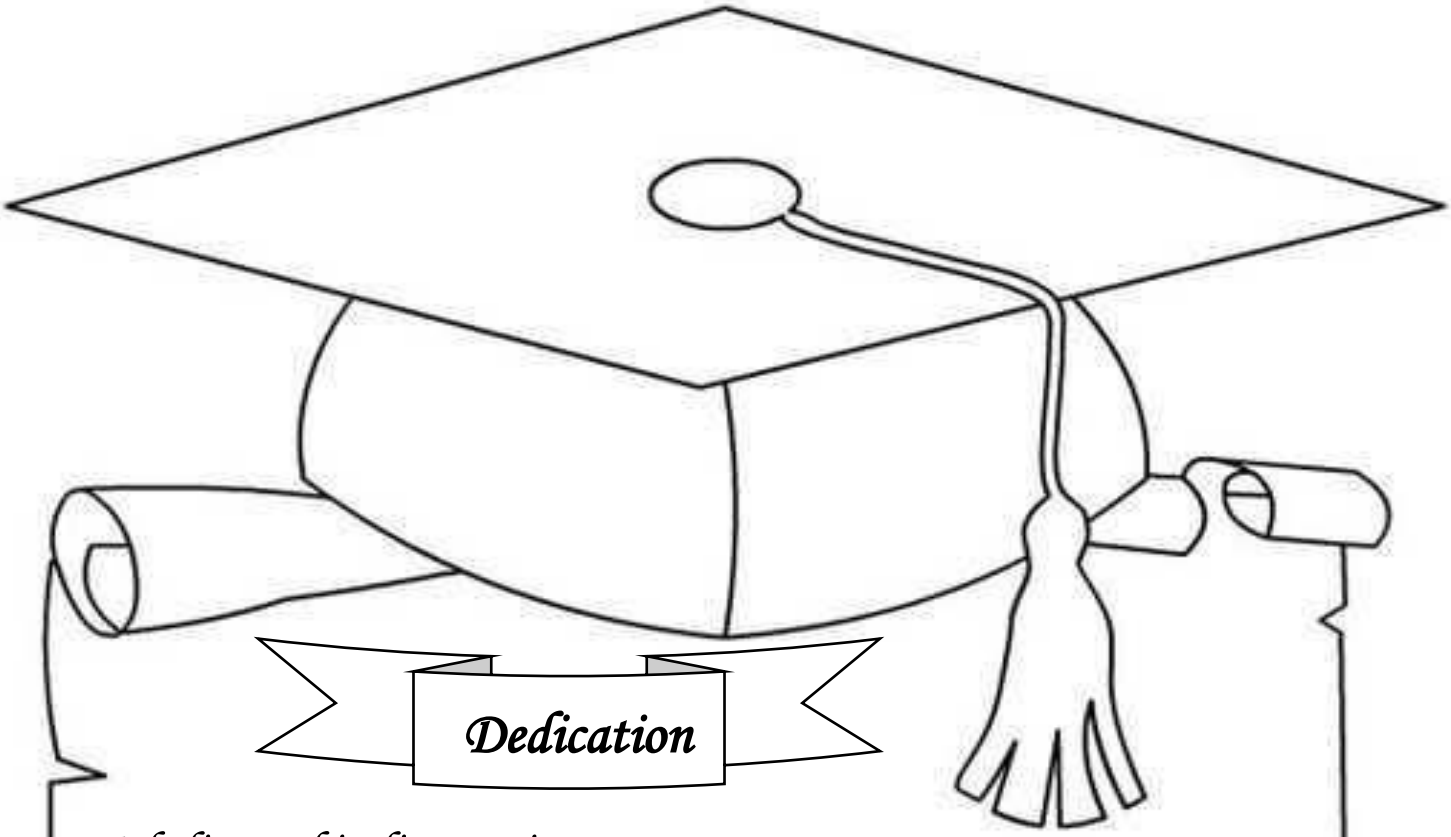
*My partner **Younes** who motivated me to work harder*

*My honest friends who helped me (**Djaballah Souhaib, Boubaker Soltani, Yacine Soltani**)*

To All the members of my family and to those who have helped me

To realize this project.

RACHID



Dedication

I dedicate this dissertation to:

*My dear **parents** whose have blessed and supported me throughout the ordeal fulfilling our work*

*My brothers **Lakhdar** and **Haithem** have never left my side.*

*My dear sisters **Zineb** and **Hadjer** whose words of encouragement and push for tenacity ring in my ears.*

*My partner **Rachid** who gave me the energy to work harder*

*My honest friends who helped me (**Djaballah Souhaib, Maamoun Oussama, Oubiri Radhouane**)*

To All the members of my family and to those who have helped me to realize this project.

YOUNES

Acknowledgment

Our greatest gratitude and our deepest thankfulness are to the One who should be praised first, to the Almighty ALLAH who gives us strength and guidance that enabled us to fulfill this study.

A very profound debt of gratitude is owed to our supervisor *Dr Lejdel Brahim* for his continuous encouragement, guidance, support, insightful criticisms and patience.

Also, we are thankful to all the examiners who spared us some of their precious and busy time to read, correct and criticize our work which is not immune from mistakes and imperfections.

We would like to acknowledge and express our heartfelt gratefulness to all our teachers without exception.

Finally, we would like to thank those who helped us to accomplish this work

Abstract

Drones are considered one of the important tools in many fields (photography, film-making, etc.). Overtime their use has developed quickly nowadays, as they are used in other fields such as research and investigation, facial recognition ... etc. This widespread use has led to the development of methods of using drones due to the artificial intelligence algorithms. Among them are the machine learning and deep learning algorithms where drones have become autonomous, which makes them have good flight capabilities and adapt to the environment in which they are, in addition to the optimization algorithms, which give them the ability to cross a trajectory in shortest path. For this purpose, we aim to design, model and automate a drone that is able to fly and follow a predetermined path by using two main techniques, the first one is the artificial neural network which is used for the obstacle avoidance and the second one is the A* optimization algorithm to search for the shortest path.

Keywords: Drones, artificial intelligence, facial recognition, machine learning, deep learning, and optimization algorithm.

مـــلـــخـــص

تعد الطائرات بدون طيار أحد الوسائل المهمة في العديد من المجالات (التصوير الفوتوغرافي، صناعة الافلام ...الخ) ومع مرور الوقت تطور استخدامها في الوقت الحاضر بشكل رهيب حيث اصبحت تستعمل في مجالات اخرى مثل البحث والتحري، التعرف على الوجه ...الخ. حيث ادى هذا الاستخدام الواسع الى تطوير اساليب استعمال الطائرات بدون طيار وذلك بفضل خوارزميات الذكاء الاصطناعي. ومن بينها خوارزميات التألية حيث اصبحت الطائرات بدون طيار ذاتية التحكم مما جعلها تتمتع بقدرات طيران جيدة و تتأقلم مع البيئة الموجودة فيها، اضافة الى خوارزميات تحسين التعرف على المسار والتي بدورها اصبحت تسلك أقصر مسار . ولهذا الغرض فإننا نهدف الى تصميم، نمذجة و تألية طائرة بدون طيار و التي تكون قادرة على الطيران و اتباع مسار محدد مسبقا. باستخدام تقنيتين رئيسيتين ، الأولى هي الشبكة العصبية الاصطناعية التي تُستخدم لتجنب العوائق والثانية هي خوارزمية البحث بأولوية الافضل للبحث عن أقصر مسار.

الكلمات المفتاحية: طائرات بدون طيار، الذكاء الاصطناعي، خوارزميات التألية.

Résumé

Les drones sont considérés comme l'un des outils importants dans beaucoup de domaines (photographie, réalisation de films, etc.). Au fil du temps, leur utilisation se développer rapidement, car ils sont utilisés dans d'autres domaines tels que la recherche et les enquêtes, la reconnaissance faciale ... etc. Cette utilisation répandue a conduit au développement de méthodes d'utilisation de drones grâce aux algorithmes de l'intelligence artificielle. Parmi eux se trouvent les algorithmes d'apprentissage automatique et d'apprentissage profond où les drones sont devenus autonomes, ce qui leur confère de bonnes capacités de vol et de s'adapter à l'environnement dans lequel ils se trouvent, en plus des algorithmes d'optimisation, qui leur donnent la capacité de traverser la trajectoire la plus courte. Pour cela, nous visons à concevoir, modéliser et automatiser un drone qui est capable de voler et de suivre un chemin prédéterminé, en utilisant deux techniques principales, la première est le réseau de neurones artificiels qui est utilisé pour éviter les obstacles et la deuxième c'est l'algorithme d'optimisation A * pour rechercher le plus court chemin.

Mots-clés: drone, reconnaissance faciale, apprentissage automatique, apprentissage profond, algorithmes d'optimisation

Table of contents

Chapter 2: State of the art

Chapter 3: Optimization algorithms and artificial intelligence techniques

Chapter 4: Modeling and implementation

List of figures

List of tables

Acronyms

ACKTR: Actor-Critic using Kronecker-facotred Trust Region.

AGL: Above Ground Level.

AirSim: Aerial Informatics and Robotics Simulation.

ANN: Artificial Neural Network.

API: Application Programming Interface.

BAE: British Aerospace.

CIA: Central Intelligence Agency.

CNN: Convolutional Neural Network.

CPU: Central Processing Unit.

DDPG: Deep Deterministic Policy Gradient.

DJI: Da-Jiang Innovations.

DoD: Department of Defense.

DPG: Deep Policy Gradient.

DQN: Deep Q-Network.

FAA: Federal Aviation Administration.

GPS: Global Positioning System.

GPU: Graphics Processing Unit.

IoT: Internet of Things.

K-NN: K Nearest Neighbors.

LQR: Linear Quadratic Regulator.

MSL: Mean Sea Level.

PPO: Proximal Policy Optimization.

PSO: Particle Swarm Optimization.

RAM: Random Access Memory.

RC: Radio Control.

ROS: Robot Operating System.

TRPO: Trust Region Policy Gradient.

UAV: Unmanned Aerial Vehicle.

General introduction

General introduction

Two decades ago, the use of drones was restricted to some companies or military uses, its uses were intended only for a specific mission. Nowadays, it moved from private to civilian use, this quick switch led to more creativity in their use, in which became a source of money for a lot of content makers (such as filmmakers, video bloggers… etc..). The civilian use of drones offer many job opportunities, especially for those who have skills in photography, it stimulates the users of drones to be more creative in their content so that they left the basic and traditional skills and move to use drones which give them more options in order to have a great work.

Artificial Intelligence had the main role for multiplicity of used methods. Due to its techniques (Machine learning and deep learning). Machine learning algorithms have opened the doors for UAVs applications in many fields such as smart cities, agriculture, smart industrial, IoT (Internet of Things), data collection and processing ..etc. In addition to the deep learning algorithms which made it smarter (object detection and recognition),

This tremendous development has not prevented researchers from having some problems and challenges. The facial recognition is one of the most popular challenges, the head motion can cause changes in facial appearance in which it effects on the ratio of automated facial recognition. Also, the obstacle avoidance is considered as another challenge, the algorithms used are not enough to detect all the collisions in the environment, moreover it requires a lot of training in the same environment. Consequently, the drone takes too long to be trained and detect the obstacles more quickly. Further, the path optimization is considered as another challenge, most of the test performed on this issue did not achieve a good results for many reasons such as the lack of the environment data or they did not use the appropriate optimization algorithm.

In our work, we will focus on the path optimization and obstacle avoidance. For this purpose, our system is based on capture images, collecting data from a specific environment. These collected data will help us to achieve the desired objective by using an appropriate optimization path algorithm and a convenient method for the obstacle avoidance.

The purpose of the work

The aim of this dissertation is to design, model, automate and build a quadrotor who is able to fly and follow predefined paths. Because of the COVID-19, we could not get the different parts

of the Drone from Provider. Thus, we cannot construct a real drone in this project. For realizing our project, we use the simulation platform.

Work plan

The dissertation is divided into four chapters:

- **The first chapter:** this chapter is about a general overview of drones and the different uses and applications of it.
- **The second chapter:** it includes some related works, explain the different methods of these works and then we will discuss the results of each one.
- **The third chapter:** this chapter has two main parts: the first one is a comparison between the optimization algorithm, however the second one is about the different techniques of the artificial intelligence
- **The fourth chapter:** we will propose a general architecture of our model, an appropriate environment, also it contains the implementation and results phase.

Chapter 1:

General overview of drones

1.1 Introduction

Drones or UAVs (unmanned aerial vehicles) are unmanned flying vehicles on board capable of carrying out a mission more or independently. Their main use is military for reconnaissance or surveillance missions, without risk of human loss. In fact, they are well suited for carrying out missions that require performance in the area which would be tedious for a crew on board. Their use began with everything related to observation, then was extended by the acquisition of objectives as well as electronic warfare, and the destruction of targets. Today, civil applications are emerging, such as monitoring motorway traffic, preventing forest fires, collecting meteorological data, and human intervention.

In this chapter, we will present a general overview of drones, its uses and its application fields.

1.2 The evolution of drone overtime

1.2.1 The earliest breakthroughs

In the beginning, Jaques and Louis Breguet are the first invent the quadcopter. Their invention had so many troubles being manageable and needed man power. The Breguet brother's invention is the beginning of the quadcopter project. [1]

Figure 1.1 represent the first quadcopter in the world.

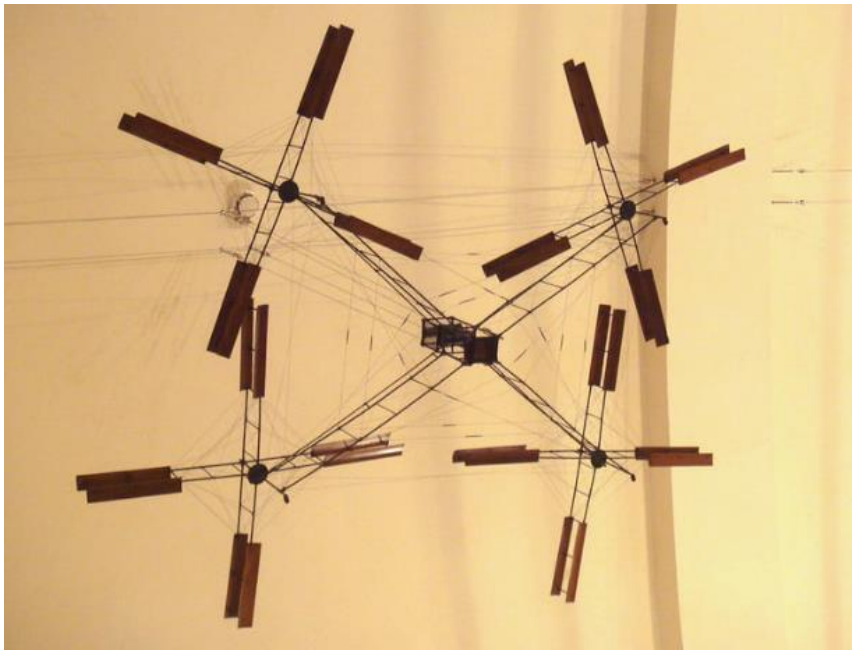


Figure 1.1 The world's first quadcopter

1.2.2 The first military drones

In 1917**,** the Ruston Proctor Aerial Target was the first unmanned aircraft in the world. This aircraft used RC technology from Nikola Tesla and controlled by radio. The purpose of the aircraft mentioned above is to drive it into the enemy territory and explode without taking casualties. The unmanned aircraft was never used at that time but it gave the first idea for today's unmanned aerial military vehicles. [1]

In 1943**,** FX-1400 was the first German armed drone capable of carrying 2.200 pounds of explosives in order to sink enemy ships. It was the first military drone that had the anti-ship missile and accurate weapons. [1]

Figure 1.2 shows the first military drone.

Figure 1.2 The Arial target drone.

1.2.3 The RC plane

In 1960s, drones were available for customers for a good price, there were different types of drones with different sizes and different job, in the US. [1]

Figure 1.3 represents the RC plane.

Figure 1.3 The RC plane

1.2.4 The first armed drone strikes

In 2001, the Central Intelligence Agency (CIA) were actively attacking the Taliban insurgents with armed drones operating both as a killing and surveillance machine. The first US drone kill was in Afghanistan, the target was Bin Laden but it turned out to be an innocent man named Daraz Khan, the effectiveness and accuracy of the unmanned drones were suspicious in modern warfare. [1]

1.2.5 FAA creates commercial drone permits

In 2006, the federal administration sees the importance of non- militarized drones and started the commercial drone project. The FAA opened the door for special businesses and professionals to use the drone for their inquiries. [1]

1.2.6 The Parrot AR Drone:

In 2010, parrot AR drone (which is a four-rotor aircraft which combines many of the new and advanced technologies in R/C Flight) was the first drone controlled by a smartphone via WI-FI. The French invention was a success and sold 500.000 thousand copy. [1]

1.2.7 Amazon Prime Air

In 2013, Amazon expressed interest in aerial deliveries by drones, the founder Jeff Bezos said that it is possible to use the technology to make half-hour deliveries but it requires a new federal rule. [1]

1.2.8 The Lily drone debacle

In 2015, a fail attempt to make a drone with a high camera capability happened with Lily Camera Drone (see Figure 1.4). The company bankrupted and shut down because of delays. [1]

Figure 1.4 The Lily Camera drone

1.2.9 Drone get smarter

In 2016, DJI PHANTOM 4 was the great success in new generation drones, it has a smart computer that helped avoiding obstacles without GPS guiding. This Incredible drone helped photographers precisely and consumers generally [1]

Figure 1.5 represents one of the most popular DJI drones

Figure 1.5 DJI PHANTOM Drone

1.3 Classification of drones

There is not a specific classification of drones. It can be classified according to serval criteria: Size, range and endurance. [2]

1.3.1 According to Size

1.3.1.1 Very small UAVs

In this class, we have the UAVs which has 30-50 cm of dimension ranging, rotary wings and a very light weight. It is used for spying and biological warfare.

Here are some examples of very small UAVs:

- The Israeli IAI Malat Mosquito (with wing span of 35 cm and endurance of 40 minutes).
- The US Aurora Flight Sciences Skate (with wing span of 60 cm and length of 33 cm).
- The Australian Cyber Technology CyberQuad Mini (with 42x42 cm square), and their latest model, CyberQuad Maxi. [2]

Figure 1.6 shows some examples of very small UAVs.

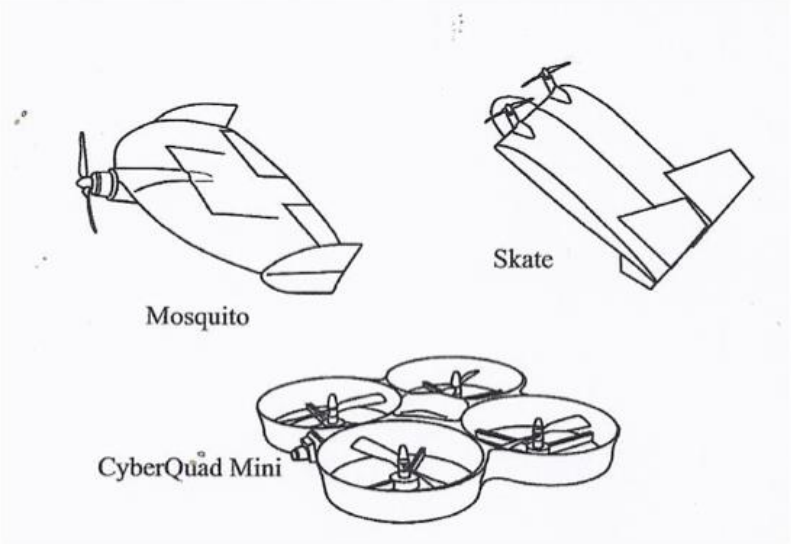


Figure 1.6 Examples of very small UAVs.

1.3.1.2 Small UAVs

The small UAVs or mini-UAVs class includes the UAVs with dimension equal to 50 cm or more while its larger have to be less than 2m. Figure1.7 shows hand-lunched small UAVs. In this context, we cite some examples of this class:

- The 1 meter long RQ-11 Raven, by US Aero Vironment with a wingspan of 1.4 m;
- The US Army RQ-7 Shadow (see Figure 1.8); [2]
- The Turkish Bayraktar (see Figure 1.9), which weighs about 5 kg and has a data link range of 20 km.

Figure 1.7 Hand-launched small UA

Figure 1.8 A US Army RQ-7 Shadow

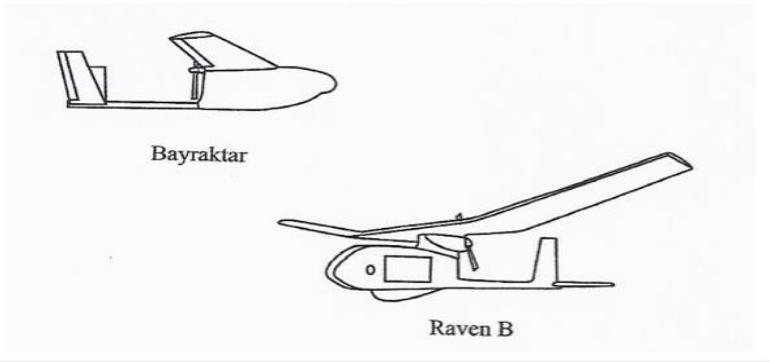


Figure 1.9 The Turkish Bayraktar

1.3.1.3 Medium UAVs

The Medium drone is smaller than a light aircraft but still has an interesting payload of 100 to 200 kilograms. Usually, the wingspread of this drone is somehow between 5 to 10 meters. The medium unmanned aerial vehicle's model are:

- UK watch keeper,
- US Boeing Eagle Eye,
- RQ-2 Pioneer,
- Israeli-US Hunter,
- BAE systems Skyeye R4E,
- RQ-5A Hunter.

The Hunter is 6.9 meters long and 10.2 meters for wingspread, while the RS-20 is a crossover drone that has the specifications of a small and medium drone. [2]

Figure 1.10 shows some examples of medium UAVs.

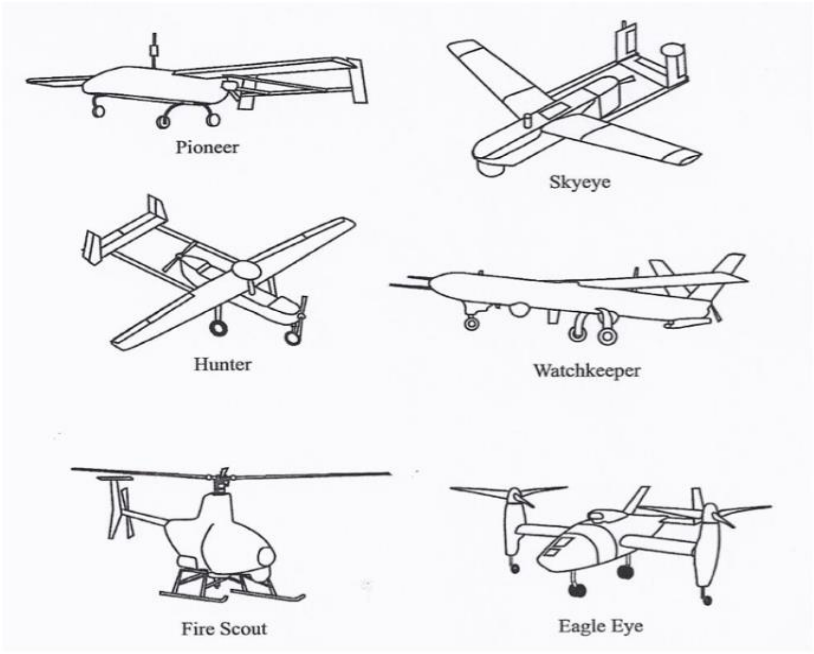


Figure 1.10 Medium UAVs

1.3.1.4 Large UAVs

The large UAV class applies to the large UAVs used mainly for combat operations by the military. Some examples of large UAVs are:

- The US General Atomics Predator A and B (see Figure 1.11)
- The US Northrop Grumman Global Hawk (see Figure 1.12). [2]

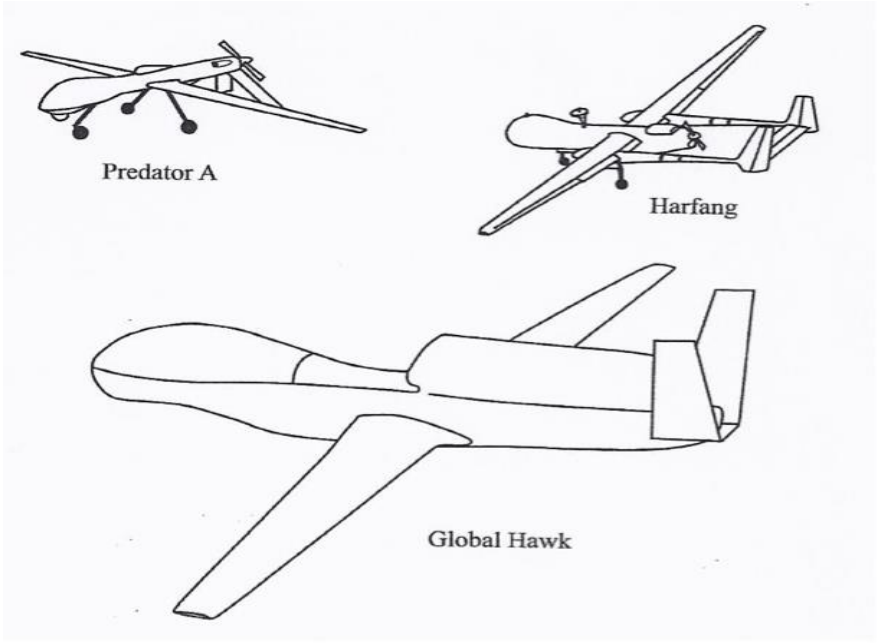


Figure 1.11 Large UAVs

Figure 1.12 NASAs Global Hawk

1.3.2 According to range

The range and the endurance are another criterion that could be adopted in the classification of UAVs. We cite some examples of its sub-classes.:

- Very low cost close-range UAVs
- Close-range UAVs
- Short-range UAVs
- Mid-range UAVs
- Endurance UAVs

Table 1.1 shows the classification of the UAVs categories according to the U.S.Departement of Defense [2].

Table 1.1 UAVs Classification

Category	**Size**	**Maximum Gross Takeoff Weight (MGTW) (lbs)**	**Normal Operating Altitude (ft)**	**Airspeed (knots)**
Group 1	Small	0-20	<1,200 AGL*	<100
Group 2	Medium	21-55	<3,500	<250
Group 3	Large	<1320	<18,000 MSL**	<250
Group 4	Larger	>1320	<18,000 MSL	Any airspeed
Group 5	Largest	>1320	>18,000	Any airspeed

1.3.3 According to Range and Endurance

1.3.3.1 Very low-cost, close range UAVs

This class includes UAVs with:

- Range of 5 km,
- Endurance time of 20 to 45 minutes,
- Cost of about $10,000 ,
- UAVs are similar to model airplanes.

There are many types of UAVs in this class some of them: The Dragon Eye and Raven. [2]

1.3.3.2 Close range UAVs

This class includes UAVs with:

- Range of 50 km
- Endurance time of 1 to 6 hours.

They are generally used for reconnaissance and surveillance tasks. [2]

1.3.3.3 Short range UAVs

This class includes UAVs with:

- range of 150 km or longer
- Endurance times of 8 to 12 hours.

They are mainly used for reconnaissance and surveillance purposes. [2]

1.3.3.4 Mid-range UAVs

This class includes UAVs with:

- Super high speed and a working radius of 650 km.

They are used for reconnaissance, surveillance purposes and gathering meteorological data. [2]

1.3.3.5 Endurance UAVs

This class includes UAVs with:

- Endurance of 36 hours and a working radius of 300 km.
- It can operate at altitudes of 30,000 feet.

They are used for reconnaissance and surveillance purposes. [2]

1.4 Uses of drones

1.4.1 Military use

It presents lower risk to combatants, saves their lives by keeping information available in real time. Also**,** it can be used in different missions such as espionage, identifying an area … etc.

1.4.2 Civil use

This type of drone can contain many systems as:

- Gas and petrol pipeline monitoring,
- Surveillance of dangerous cargoes, water sources,
- Forest fire monitoring: It is important to note that drones are very useful and effective on the subject of observation and controlling forest fire.

1.5 Application of drones

There are serval fields that drones can be applied, here are some of the most important application fields:

- **Investigations**: it helps the security agencies in a lot of issues for example: tracking car,
- **Industry**: risk zone control, System monitoring (solar panel, wind turbine),
- **Architecture and urbanism:** A good study of the site (dimensions), which allows for a good blueprint,
- **Tourism**: Filmmaking and photography,
- **Chemical hazard detection**: detection of toxic gases, radiation.

1.6 Conclusion

In this chapter, we described some features, uses and applications of drones in order to have more information about them. But, having an idea about the used method for these applications is required. This is what led up to explain some methods in the next chapter.

Chapter 2:

State of the art

2.1 Introduction

The widespread use of drones led to the development of its methods, most of these methods are related to the artificial intelligence algorithms. In order to know more about these methods, the related works are useful for it.

In this chapter we will take some related works. In each related work, we explain the used methods and we present the results and their discussions.

2.2 Obstacle avoidance drone

2.2.1 Overview

In this context, we can cite the work of Khosrow-Pour et al.[3] which study the obstacle avoidance drone. In this work, the authors use Deep learning algorithm which has achieved a fast progression. The Deep learning algorithm allows us to study an autonomous drone which has the capacity to avoid obstacles by using two machine learning paradigms: supervised learning and reinforcement learning [3].

In order to achieve the goals of this study, it proposes a new framework where the labels made by the actor-critic network in a reward-driven manner are trained by the supervised segmentation network, wherein this U-Net based network infers the next moving direction from the sequence of input images. For the actor-critic part, several recent policy gradient algorithms have been tested for controlling the drone with the continuous action space [9], so the main principle of this study is:

- Monocular vision-based drone with obstacle avoidance capability in the mazes.
- Reward driven Actor-Critic provides a depth map to U-Net, saving manual labeling.
- Segmentation model with U-Net infers the next moving direction for a flying drone.
- It can fly not only in the trained maze but also in the untrained mazes.

2.2.2 Used method

2.2.2.1 Critic-dependent segmentation model

Actor- critic network seeks to accommodate the analyzed network through developing a label map. Two stages are essential for developing the training data regarding the anticipated

reward. The first stage is for the agent to choose a direction to go inside an image. The second one is to develop a label map for the analyzed model based on the anticipated reward utilizing the optical flow vectors. [3]

2.2.2.2 Deep deterministic policy gradient

Deep Deterministic Policy Gradient (DDPG) is an off-policy actor-critic algorithm that combines DQN with Deep Policy Gradient (DPG) for the learning in continuous action space.[3]

2.2.2.3 Trust region policy gradient and proximal policy optimization

Trust Region Policy Gradient (TRPO) and Proximal Policy Optimization (PPO) are fundamentals methods in neural network but the main challenge is that these methods are very sensitive to choose steps size, the progress is too small or too large which make the response will be too noisy.[3]

2.2.2.4 Actor-Critic using Kronecker-factored trust region

Actor-Critic using Kronecker-facotred trust region (ACKTR) is a method similar to A2C architecture (Advantage Actor Critic without asynchronous), this method is proposed to apply the Kronecker-factored approximation when each network updates the gradient in the actor-critical model.[3]

2.2.2.5 Actor and critic networks

Although applying actor-critic algorithms can be optimal in some certain environments. The problem of using vision input in the continuous action spaces still existing when all the state has high dimensions with a similar appearance. Thus, to overcome this issue they adopt a model which consists in compressing information while preserving useful functionalities. [3]

2.2.3 Results

The original configuration achieved 19 successful trials out of 20 trials while the second one has 17 successful trials out of 20 ones. The third and the fourth trials achieved 15 successful trials out of 20 and 11 successful trials out of 20 trials respectively. [3]

The experimental results are summarized in Table 2.2:

Table 2.2 The experimental results

Real environments	Trials	Complete/Non-complete
conf. 1	20	19/1
conf. 2	20	17/3
conf. 3	20	15/5
conf. 4	20	11/9

2.2.4 Discussion

In this type of drone, we conclude that the result obtained in the work of Khosrow-Pour et al.[3] has very good performance in both of the trained and untrained mazes with the present U-Net trained with the Actor-Critic network. The simulation platform AirSim which is used in this work, is a good simulation platform which allows to generate a realistic graphic content using the Unreal Engine.

2.3 Facial recognition drone

2.3.1 Overview

The design of facial recognition drone is practiced when identifying crime suspects. The work of N. F., Xavier & L. PAGLAN [4] utilize facial recognition algorithms. It is a method which can be executed by tools such as MATLAB to model the drone, and OPENCV for programming the facial recognition using biometric properties. Moreover, the LQR (Linear Quadratic Regulator) command allows to control the machine while moving on the concept of state variables. The raspberry control board (microcomputer) is necessary for face recognition to achieve the desired goal. It is programmed in python language [4]. In this work, the authors follow the following plan [15]:

- Define the architecture of a drone,
- Release all the blocks that constitute it,
- Dimensioned the constituents of each block,
- Program the raspberry card,
- Simulation process for the stability, controllability and observability of the drone.

2.3.2 Used methods

Face detection raised the question of locating faces presented as an input image. Ideally, the detection also provides their dimensions for possible further processing [4]. The main methods used in this work are:

➢ **Knowledge-based methods:**

These methods are based upon the different elements which constitute a face and the relations between these elements as a consequence, the face components such as mouth, eyes and nose are measured and then used for the 'face' 'non-face' classification. This method has some difficulty to recognize the face, if the definition has too much details. Faces will be lost, if the description is too general and the false positive rate will raise dramatically [4].

➢ **Feature invariant approaches:**

These approaches use invariant elements in illumination, orientation or expression such as the texture and the color signature of the skin for face detection [4].

➢ **Template matching methods:**

Model characteristics of an entire face or its components (mouth, eye, and nose) are created. Localization is done on the basis of the correlation of these models with the candidates [4].

➢ **Appearance-based methods:**

These methods use the same principle as presented in the previous point but are based on models learned from a training set. These advanced based methods have the advantage of being executed very quickly but require a long training time. The methods of this category have shown good results compared to the other three types of methods, among them we can cite the method based on the neural networks and the Schneiderman and Kanade method based on a naive Bayes classifier as well as the famous Viola and Jones algorithm which operating in real time. [4]

2.3.3 Tests and results

2.3.3.1 The stabilization tests

The first test after stabilization of this system gives a response time of 3 seconds at altitude (Z) and a follow-up of the reference for the other positions (response time 1s) and finally a response time of 2.5s for the angle Psi [4].

The Second test of the dynamic system gives a response time remains 3 seconds at the altitude (Z) but as the altitude is varied, the system has difficulty following the instruction and a monitoring of the reference for the other positions (response time 1s) and finally a response time of 1.5s for the angle Psi [4].

The Third test of the dynamic system was tested by changing the setpoint of the altitude and the angle Psi, observe a response time remains 3 seconds at the altitude (Z), when the setpoint is stabilized for more than 6 s (response time + time to return to the initial position) [4].

2.3.3.2 Facial detection and recognition test

This recognition script works well for people within 2 meters from the camera. [4].

2.3.4 Discussions

This work achieved its goal relatively by using both of stabilization and facial recognition methods. But, testing other methods is required especially in the facial detection for better results.

2.4 Navigating Assistance System

2.4.1 Overview

In this category, we can cite the work of Tung-Cheng wu [16]. They use a quadcopter agent for bypassing obstacles in 3D environment. This agent has two parts, one is quadcopter navigating function. This function will find the shortest straight path from quadcopter to target position [5]. The other part is collision avoidance function. This function is a deep Q-network, its role is planning the route to bypass obstacle [5].

This work use 3D environment which is based on unreal engine with AirSim. This simulation platform is presented by Microsoft. With this environment, we can easily simulate quadcopter flies in city.

2.4.2 Used method

2.4.2.1 Quadcopter navigation function

Quadcopter navigation function calculate the shortest path which is the straight line from quadcopter to the target position. To find this shortest path, we need to know the position of quadcopter and its target destination. Then, we need to calculate the turning degree for quadcopter to directly go to the target destination. In the simulating experiment, AirSim coordination is used to present positions. In the real world, GPS information may be used [5].

2.4.2.2 Collision avoidance function

On the way to target, quadcopter may face some obstacles on the way there. To avoid these obstacles, the authors use the deep Q network [5].

2.4.2.3 Action and Exploration Strategy

In this method, they output 13 different actions about quadcopter. These 13 actions guide quadcopter fly to different direction. Actions can be separated into two categories. One move attitude, agent will fly up, fly forward, fly down. The other is turn angle, agent will turn left 15°, turn left 5°, turn right 5°, and turn right 15°. Combining the above two categories, we can get 12 actions. The last action is going forward [5].

2.4.2.4 Decide final action with two functions

There are two functions in this model (quadcopter navigating function and collision avoidance function). Both models output how many degrees quadcopter must turn and add the output degrees directly. As a result, agent controls the turning degree by both quadcopter navigating function and collision avoidance function, and alter flying attitude/altitude by collision avoidance function [5].

2.4.2.5 Reward

After reaching new position, agent will have the new information from the environment. Using this new information, agent will assess the last action, and that is called feedback “reward”. [5]

2.4.2.6 Network Training

In this work, the authors introduce Q-network and a brief interpret on how Q-network update parameters in the neural network. Practically, there are two Q-network in training. One is

Qvalue-network that used to calculate Q-value, and the other is Qtarget-network. Every 50 steps, the authors enter the training state. They will randomly select a mini-batch dataset from experience. The parameters in Qvalue-network will be updated in every training state. However, the parameters in the Qtarget-network will update every 500 steps, and be updated by copy the parameters in Qvaluenetwork. [5]

2.4.3 Tests and results

In this work, a record of model weights each 100 flights until 500 flights has been made. Taking the training weight to fly in another place that drone does not fly before, in the same module.

Each weight was tested 1000 flights and recorded whether the drone collides with obstacles and how many rewards that agent will get in each step of flight. Figure 2.1 shows collision percentage of each weight in 1000 flights.

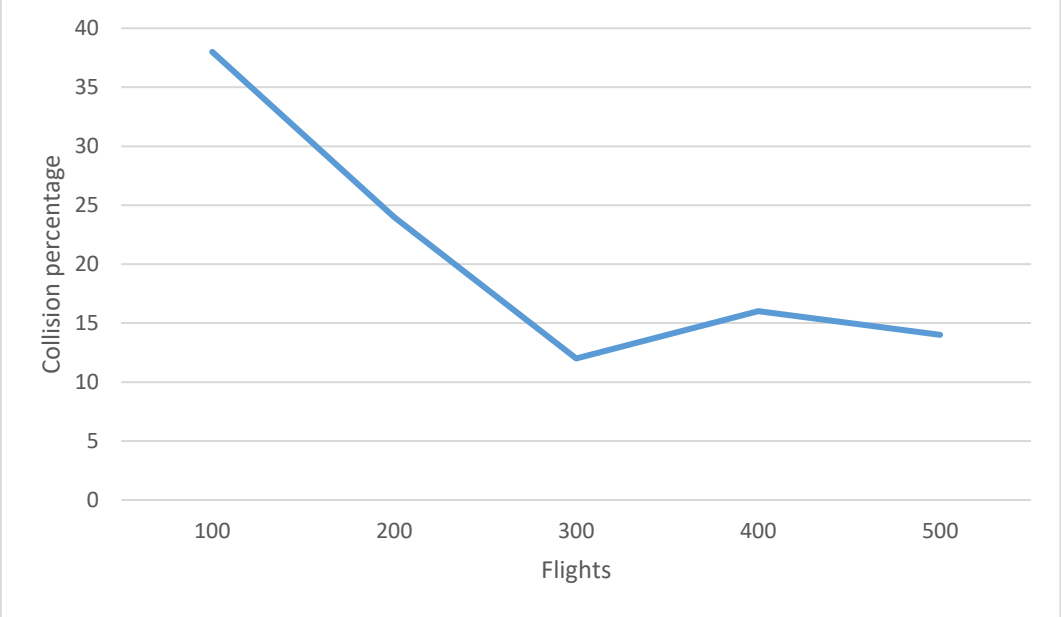


Figure 2.1 Plot of collision percentage of each weight.

2.4.4 Discussion

In this work, they achieve good results, in which it can avoid 86% object obstacle even in the strange path. But, it requires training in more complex environment in order to optimize this model.

2.5 Flying path optimization

2.5.1 Overview

The work of Y., Sang-Jo et al. [6] treat the flying path optimization problem. This work presents an optimal flying path for unmanned aerial vehicle-assisted internet of things sensor networks using genetic algorithm. It uses multi-layer location-information map that takes into account various utility features based on sensor density, flight time, energy consumption and level of flying risk. The genetic algorithm uses to maximize the overall weighted sum of multi-objective utility functions. [6]

2.5.2 Method used

2.5.2.1 Genetic algorithm

In genetic algorithm, population of candidate solutions is initialized in a form of chromosomes. Each chromosome consists of a combination of sensing cells and flying cells. Flying cells are selected in a way that they connect two corresponding SIG (cells with the shortest path, The fitness of each chromosome in the population is calculated using the total utility function using the selected flying and SIG (Sensor Information Gathering) cells in the chromosome, Some of the best chromosomes are selected as an intermediate population using a roulette wheel selection scheme, ‘Order 1 crossover’ is applied to ‘cell id’ with probability Pc for each consecutive pair. The ‘swap mutation’ operation is performed by flipping the SIG indication value of each offspring with probability Pm. New random solutions are a replacement for A fraction of the previous population with lower fitness values. The process is repeated until the stopping criterion is not contended. [6]

2.5.3 Tests and results

The evaluation of results will be calculate using fitness function of the genetic algorithm. When it satisfies the stop conditions, the route path which has the maximum fitness value will be selected. After a certain number of iterations, the total fitness increases with the number of iterations and saturates. Increasing the weight of the sensing utility increases the overall utility, but more SIG cells must be visited by the UAV, which takes more time and energy [6].

2.5.4 Discussion

In this work, each sensor field cell is defined by its environmental characteristics, such as the prohibited area, condition of the channel, statistics of sensor deployment, and level of risk. A multi-objective fitness function was formulated to derive the optimum path in terms of flying and SIG cells. Simulation results showed that the proposed method could derive the desired optimum path while satisfying the constraints.

2.6 Conclusion

The related works which was mentioned above gives us the opportunity to have an idea about the methods used for the conception of different types of drones. The results of these works are relatively good and achieved the goal, but some of them need more improvement in order to get better results.

In the next chapter, we will describe the modeling and design phase of our proposition based on these related works.

Chapter 3:

Optimization algorithms and Artificial intelligence techniques

3.1 Introduction

In this chapter, we will describe our proposition which used to modelling and design a Drone. Firstly, we will compare the different optimization algorithms for choosing the best one among them, to optimize the path followed by our Drone. Secondly, we explore the different artificial intelligence techniques and choosing the appropriate one for our work. Finally, we add a conclusion.

3.2 Comparison between optimization algorithms

In this work, we aim to design a quadrotor that will be able to pass a trajectory in the shortest path. Consequently, we have to compare between some optimization algorithms, after to this comparison we will choose the most appropriate one for our model.

3.2.1 Ant colony optimization

3.2.1.1 Definition

Ant colony optimization is a metaheuristic in which a colony of artificial ants cooperate into finding the good solutions for difficult discrete optimization problems. [7]

3.2.1.2 Principle

This algorithm is based on the behavior of ants searching for food. First of all, the ants walk randomly. Then, when it finds the food, the ants leave a marker (pheromones) which show the path in where the food is. After that, if the other ants find this path, they will follow this path with a certain probability and do the same thing (put the markers). The path gets stronger as more ants find it until there are streams of ants traveling to various food sources near the colony [8].

Once the food source is depleted, the route is no longer populated with pheromones and slowly disappear [8].

Figure 3.1 represents how this algorithm works

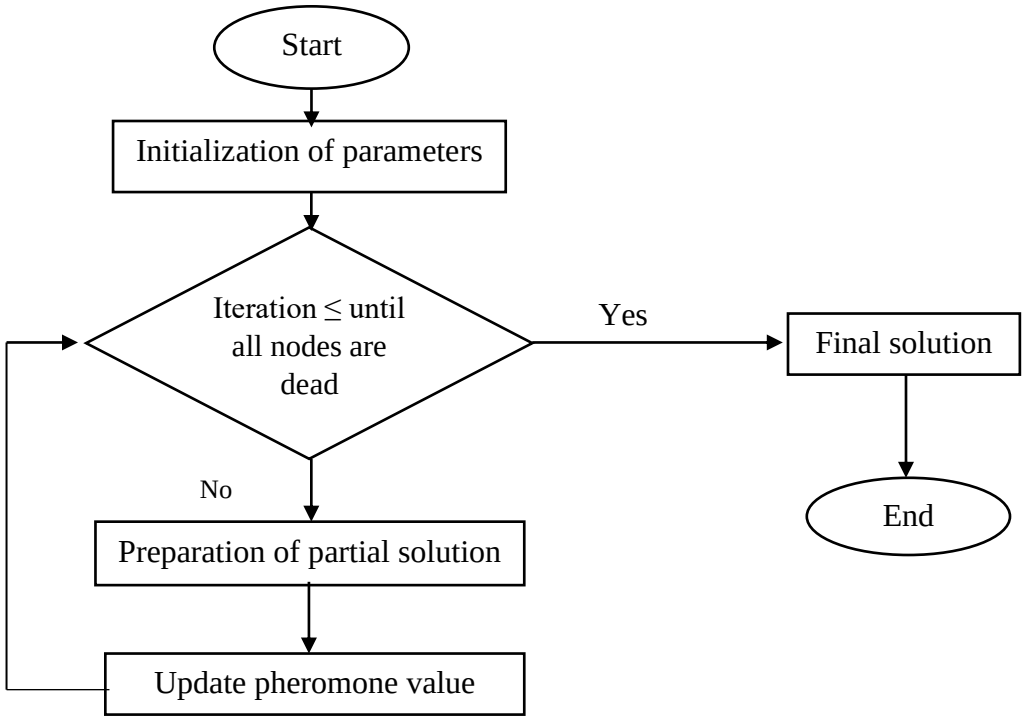


Figure 3.1 Flow chart of Ant Colony Optimization. [7]

3.2.1.3 Advantages

The advantages of this method are [7]:

- ✓ It is perfect for graph-based problems,
- ✓ It can adapt to changes such as a new distance,
- ✓ It can search among a population in parallel,
- ✓ It can give a rapid discovery of good solutions.

3.2.1.4 Drawbacks

The disadvantages of this method are [7]:

- × Long run time in the complicated cases,
- × It has a difficult theoretical analysis,
- × It cannot be applicate in all sort of problems,
- × A closed state may occur,
- × It has more experimental than theoretical research,
- × The probability of distribution can change for each iteration.

3.2.2 Particle Swarm optimization

3.2.3 Definition

PSO (Particle Swarm Optimization) is a heuristic global optimization method (also an optimization algorithm). it is based on simulating the movement behavior of the flock of birds. [9]

3.2.4 Principle

This optimization method is based on the collaboration of individuals among themselves. It also has similarities with ant colony algorithms, which are also based on the concept of self-organization. This idea is that a group of poorly intelligent individuals can have a complex global organization. [9]

Thus, due to very simple rules of displacement (in the space of solutions), the particles can gradually converge towards a global minimum. However, this metaheuristic seems to work better for spaces in continuous variables. [9]

Figure 3.2 shows the steps of the PSO algorithm.

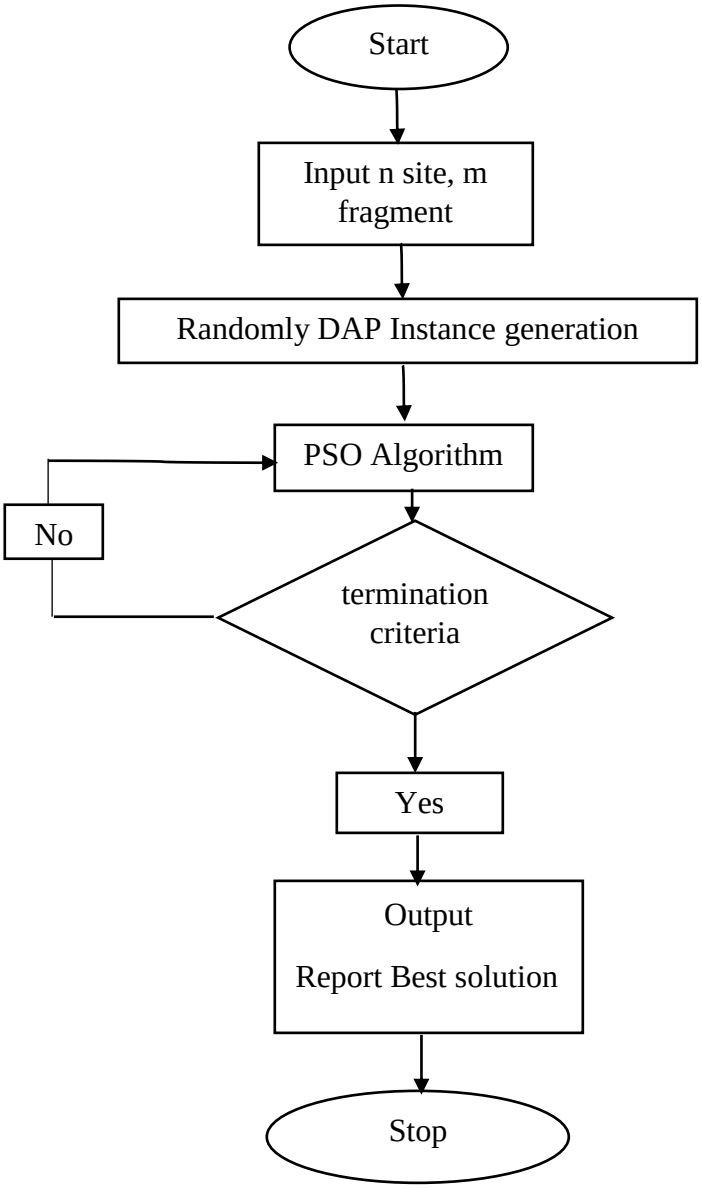


Figure 3.2 Flow chart of particle swarm optimization algorithm.

3.2.4.1 Advantages

The advantages of this method are [9]:

- ✓ It can be simple to implement,
- ✓ It has a few parameters to adjust,
- ✓ It can run parallel computation,
- ✓ It can be robust,
- ✓ It is efficient for solving problems presenting difficulty to find accurate mathematical model.

3.2.4.2 Drawbacks

The disadvantages of this method are [9]:

- It can be difficult to define initial design parameters,
- It cannot work out the problems of scattering,
- It can converge prematurely and be trapped into a local minimum especially with complex problems.

3.2.5 A star (A*)

3.2.5.1 Definition

The A* search algorithm is an algorithm using to find the shortest path in a graph between an initial node and an end node, both given. Due to its simplicity, it is often presented as a typical example of a planning algorithm in the field of artificial intelligence. [10]

3.2.5.2 Principle

A* computes the function f(n) = g(n) + h(n) where: g(n) is the actual cost and h(n) is the estimated cost.
such as: g(n) = “cost from the starting node to reach n” and h(n) = “estimate of the cost of the cheapest path to the goal node”. [10]

Figure 3.3 presents the algorithmic mechanism of the A*.

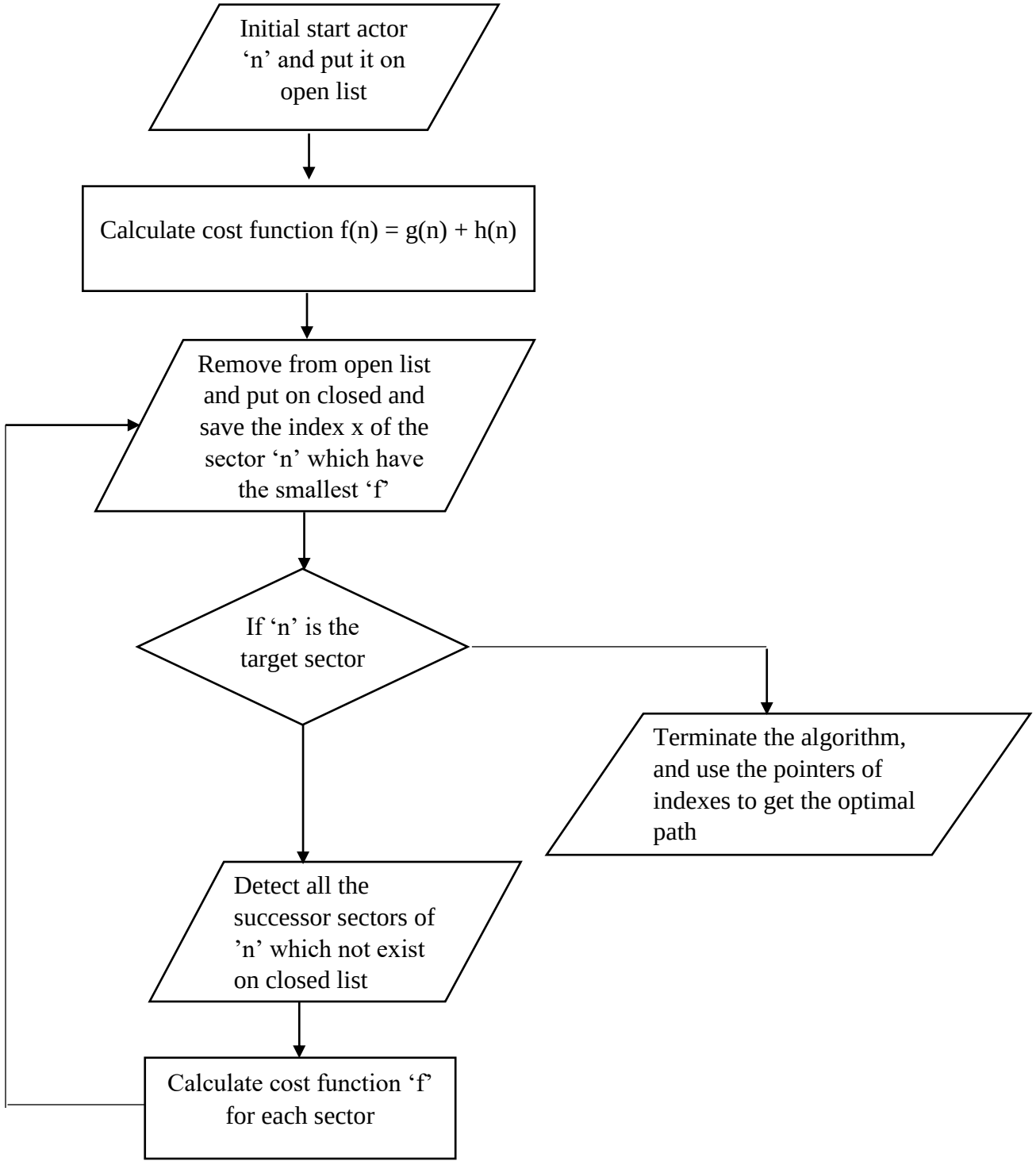


Figure 3.3 Flow chart of A star algorithm

3.2.5.3 Advantages

The advantages of this method are [10]:

- ✓ It is used to solve complex problems with path.
- ✓ It can be used in gaming and maps applications.
- ✓ It is nearly optimal compared to another optimization algorithm
- ✓ Less run times [10]

3.2.5.4 Drawbacks

The disadvantages of this method are [10]:

- × It can be optimal if every action has fixed cost
- × It may take more run time in very complex problems [10]

3.2.6 Discussions:

Previously, we can notice that the two first algorithms (Ant colony and Particle swarm) take more run time compared to the last one (A*). Also, the A* algorithm can handle with the obstacles better than the other two.

Based on this comparison, we conclude that the A* algorithm is the good choice for our model due to several reasons among them:

- Easy to handle with the obstacles (obstacles avoidance),
- Give results in appropriate execution time,
- It can simply integrate in our proposed model.

3.3 Artificial Intelligence techniques

In this section, we will explore the different algorithms of machine learning and deep learning.

3.3.1 Machine learning

3.3.1.1 Definition

Machine learning is a study field of artificial intelligence which is based on mathematical and statistical approaches to give computers the capacity to "learn" from data. The objectives of machine learning are to enable machines to make predictions, perform clustering, extract association rules, or make decisions from a given Zdataset.[11]

Machine learning algorithms can be classified according to the learning method that they use. Figure 3.4 summary all the techniques of machine learning:

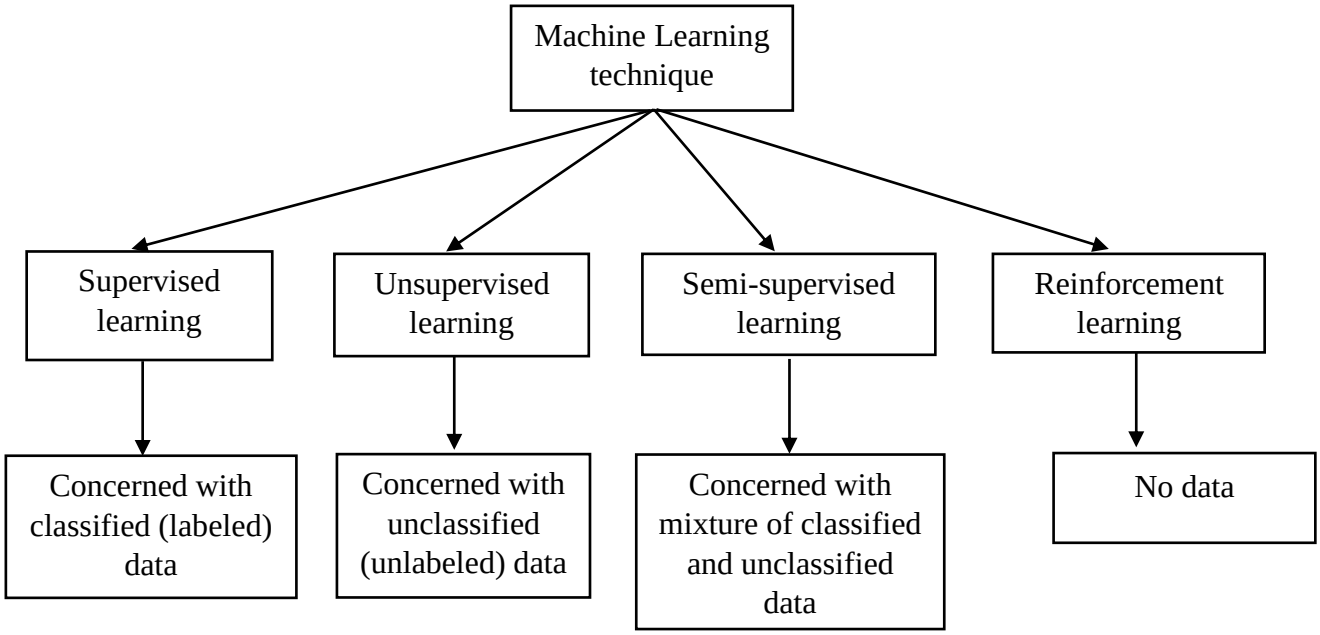


Figure 3.4 Different Machine learning techniques and their required data. [11]

We will describe the different types of machine learning.

3.3.1.2 Supervised learning

3.3.1.2.1 Definition

Supervised learning is a machine learning task when the data entering in the process is already classified and the algorithms must use it to predict an output in order to be able to do so later when the data is no longer classified. [11]

3.3.1.2.2 Algorithms

There are two categories of algorithms of supervised learning:

3.3.1.2.3 Regression:

In machine learning, regression algorithms attempt to estimate the mapping function (f) from the input variables (x) to numerical or continuous output variables (y). [11]

In this case, y is a real value, which can be an integer or a floating-point value. Therefore, regression prediction problems are usually quantities or sizes.

3.3.1.2.4 Classification:

Classification algorithms attempt to estimate the mapping function (f) from the input variables (x) to discrete or categorical output variables (y).

In this case, y is a category that the mapping function predicts. If provided with a single or several input variables, a classification model will attempt to predict the value of a single or several conclusions [11].

3.3.1.3 Unsupervised learning

3.3.1.3.1 Definition

In the unsupervised learning, we lack supervisors or training data. In other words, all what we have is unlabeled data. The idea is to find a hidden structure in this data. [11]

3.3.1.3.2 Algorithms

There are a lot of algorithms that can be used in unsupervised learning, below are the most important ones:

- ✓ **Dimensionality reduction**: structure discovery, meaningful compression, feature elicitation, Big Data visualization.
- ✓ **Clustering:** K-NN (K nearest neighbors), Hierarchical clustering, Singular Value Decomposition. [3]

3.3.1.4 Semi-supervised learning

3.3.1.4.1 Definition

Semi-supervised learning is an intermediate between supervised and unsupervised learning, the data are a mixture of classified and unclassified data. In this type, we use both of labeled and unlabeled data. The target of semi-supervised classification is to learn a model that will predict classes of future test data better than that from the model generated by using the labeled data alone. This way is pretty much similar to the process of child learning. The data in this category is divided into two types:

- **Unlabeled data** provided by the environment. In the beginning, child will see a lot of unlabeled data.
- **Labeled data** from the supervisor. For example, a mother teaches her son about his brothers and sisters names (labels) by uttering their names. [11]

3.3.1.5 Reinforcement learning

3.3.1.5.1 Definition

The reinforcement learning method aims to learn the actions to take from the interaction with the environment in order to maximize the notion of cumulative reward or minimize the risk. In order to produce intelligent agent, reinforcement learning goes through the following steps:

a. Input state is observed by the agent.
b. Decision making function is used to make the agent perform an action.
c. After the action is performed, the agent receives reward or reinforcement from the environment.
d. The state-action pair information about the reward is stored [11].

3.3.2 Deep learning

3.3.2.1 Definition

Deep learning is a type of artificial intelligence derived from machine learning where the machine is able to learn on its own, unlike programming where it is content to execute predetermined rules [12]. The neural network is the main component of deep learning,

3.3.2.2 Application

There are several fields that deep learning can be applied to, below are some examples:

- Self-car driving,
- Facial recognition,
- Visual art recognition,
- Natural language processing,
- Healthcare,
- Automatic game playing.

3.3.2.3 Main types of Deep Learning

Deep learning has a lot of types, each one can be used according to a specific problem. In our case, we will focus on the two main types of architecture for neural networks:

3.3.2.3.1 Artificial Neural Network

Artificial Neural Network (ANN) is a system whose design was originally schematically inspired by the functioning of biological neurons, and which subsequently approached statistical methods. It provides the connection between the input and output data by simulating the human brain working. [13]

Figure 3.5 presents the comparison between Natural neuron and artificial neuron.

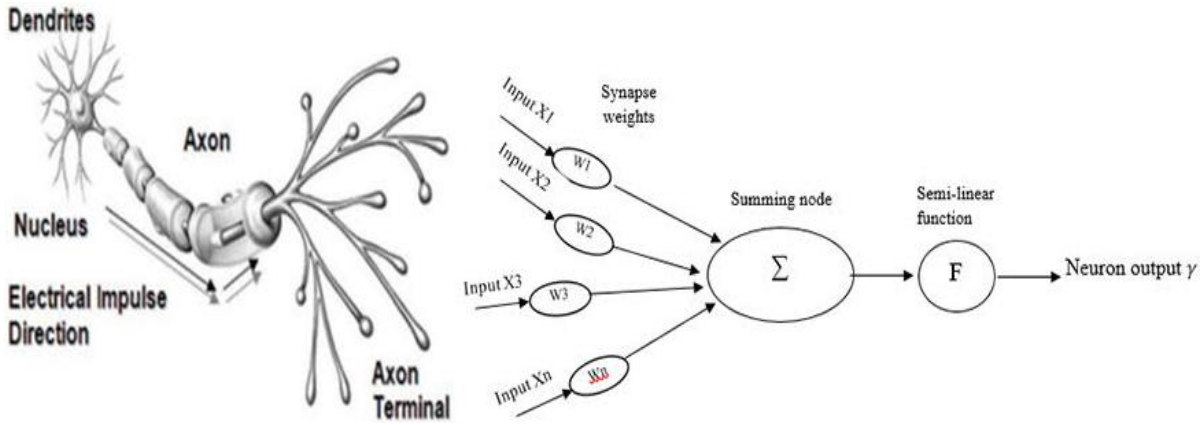


Figure 3.5 Comparison between Natural neuron and Artificial neuron [14]

A).Advantages

The advantages of this method are:

- Storing information on the entire network
- Ability to work with incomplete knowledge
- Having fault tolerance
- Ability to make machine learning [18]

B).Disadvantages

The disadvantages of this method are:

- Hardware dependency (High system requirements),
- Unexplained behavior of the network,
- Difficulty of showing the problem to the network,

- There is no specific rule for determining the structure of artificial neural networks. [18]

3.3.2.3.2 Conventional Neural Network

Conventional Neural Network (CNN) is a type of artificial neural network used in the recognition, processing of images and specially designed for pixel analysis.

In CNNs, "neurons" are arranged like those of the frontal lobe, the site of processing of visual stimuli in humans and other animals. The layers of neurons are organized so as to cover the entire visual field in order to avoid the problem of processing fragmented images of classical neural networks.

A CNN uses a system comparable to the multilayer perceptron, but designed to reduce the number of calculations. The structure of a CNN consists of a succession of layers: an input layer, an output layer and a hidden layer made up of numerous convolutional layers, grouping layers, fully connected layers and normalization layers.

The remove of restrictions and the improved efficiency of image processing have led to a much more efficient and easier to train system for automatically processing images and languages. [15]

Figure 3.6 shows the schematic diagram of a basic convolutional neural network (CNN) architecture.

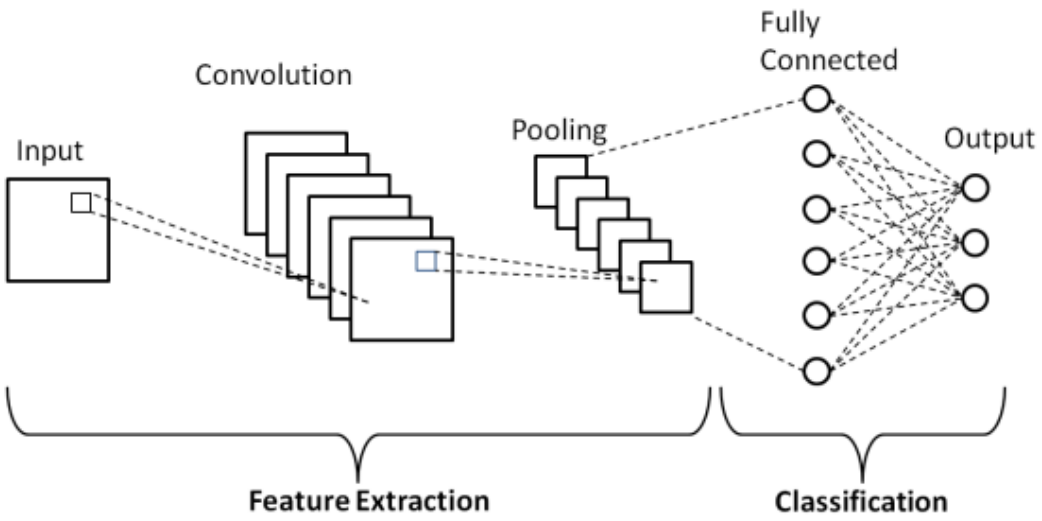


Figure 3.6 Schematic diagram of a basic convolutional neural network (CNN) architecture [16]

A).Advantages

- A Convolutional neural network is significantly slower due to an operation such as maxpool,
- If the CNN has several layers then the training process takes a lot of time if the computer doesn't consist of a good GPU,
- A Convolutional Network requires a large Dataset to process and train the neural network. [19]

B).Disadvantages

- Ability to represent any function, linear or not, simple or complex.
- Easy to use, much less personal work to do than in classic statistical analysis.
- For the beginner user, the idea of learning is easier to understand than the complexities of multivariate statistics. [19]

In our work, we choose to use the ANN because it has the ability to make machine learning, in which it can learn events and make decisions by commenting on similar events. Also, it is helpful to achieve our objective

3.4 Conclusion

In this chapter, we compare the different optimization algorithms which is useful in order to choose the best algorithm. The different features of each one (principle, advantages, drawbacks) was enough to choose the algorithm that fit with the desired work. In addition, we explain the basic concepts of Artificial intelligence (Machine learning and Deep Learning). In this stage, we choose to use ANN which is the right architecture for our work.

In the next chapter, we will present the implementation and testing phase for validating our model.

Chapter 4:

Modeling and Implementation

4.1 Introduction

Our model has many steps that we have to follow in order to achieve a good implementation. In this chapter, we are going to explain the programming language that we used to implement these steps, then we will explain each one. Also, we execute some test which are required in order to determine, evaluate and check the efficacy of our methods.

4.2 General architecture

In order to get a good architecture for this model, we have to follow the three steps below:

- Choose an appropriate environment and programming language which allows us to properly implement our model,
- Determine the dataset and its components.
- Propose the general architecture schema of this model.

4.2.1 Environment

4.2.1.1 Simulator

AirSim (Aerial Informatics and Robotics Simulation) is an open-source robotics simulation platform. It allows to:

- Build simulators for vehicles, robotics, drones and static IoT devices,
- Capture data for models without costly field operations.

It is a platform library built on Unreal Engine, and supports hardware-in-loop with flight controllers (PX4, ROS…etc). [17]

4.2.1.2 Overview

The AirSim simulator has a vehicle's model, the environment's model, physics engine to compute the resulting motions and photo-realistic. Also, it supports recording sensor observations which simulate the real word behavior (for example drifting, flying ...etc.) [17]

Figure 4.1 presents the components of this simulator architecture.

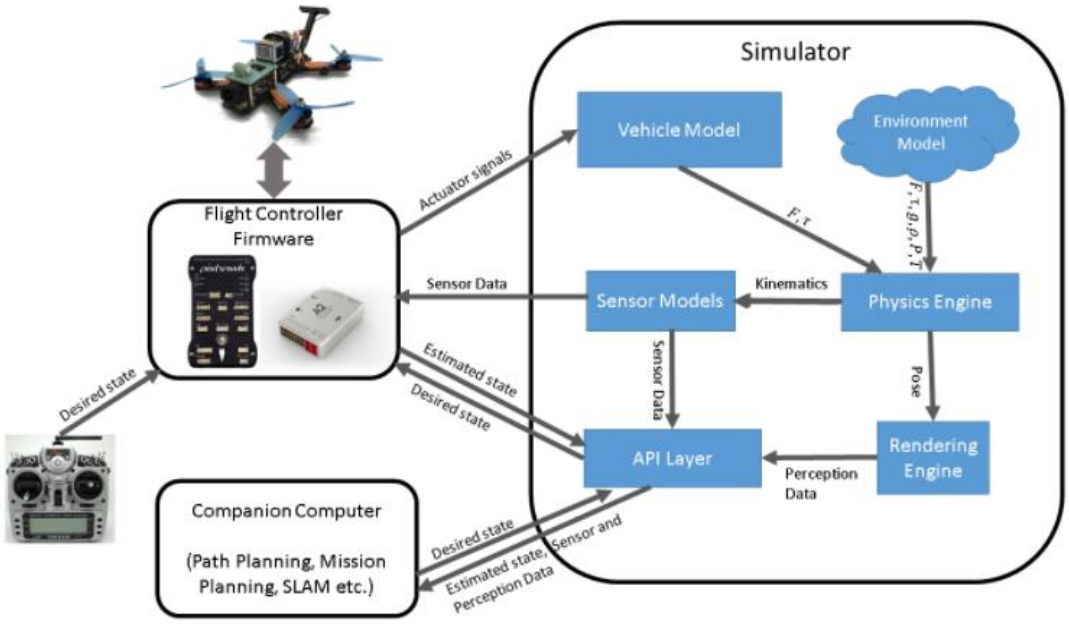


Figure 4.1. General architecture of the AirSim [17]

4.2.1.3 AirSim APIs and main concepts

AirSim provides many toolsets in order to create complex simulations. It allows us to control vehicles and get the different data and perspective on the environment by using the cameras of vehicles and taking pictures [17]. Here are some of the important toolset:

- **Common APIs:** It allows performing some common actions in the simulation, it does not matter which vehicle you are using.
- **Vehicles APIs:** It helps to get useful data on the using vehicle (car or multirotor), also it allows to control their movement by using different movement mechanisms.
- **Collision API:** If two different and separate objects touch each other there is a collision. So, Collision API allows us to detect this collision and get useful information (collision position, surface, depth ...etc.).
- **Unreal Engine (AirSim Server):** Unreal Engine is a game engine developed by Epic Games. It's used to create video games, an environment running on Unreal Engine (by adding the AirSim plugin)

- **Flight Controller:** It takes in the desired state as input, then estimates actual state by using sensors data. In the end, the flight controllers drive the actuators using this way in order to come close the actual state to the desired one. [17]

4.2.2 Python ecosystem

Usually, when it comes to choose a programming language for training a model or a system related to the artificial intelligence, Python would be the best choice for many reasons:

- **Free and open source:** Python is open to everyone online. It is available free of charge on their official website www.python.org. It has a wide group around the world working to create new modules and functions for python.
- **High level language:** When we write programs in python, we do not have to remember the system architecture or manage the memory.
- **Large standard library:** Python offer a wide standard library that includes a set of module and function i.e. writing this method manually is unnecessary.
- **Objected-Oriented language:** It supports all the OOP features and concepts (classes, objects, inheritance...etc.).

4.2.3 Proposed Model

In our model, we need three main functions, the first one based on calculating coordination point in order to find the straight path to goal, the second function has two parts which are the collision detection and avoidance while the last one is the A* implementation. In the end, the drone will combine these three functions.

In order to implement these functions, we follow these steps:

- **Data collecting,**
- **Drone training,**
- **Obstacles avoidance,**
- **Crossing the shortest path,**
- **Testing.**

Figure 4.2 shows the general architecture of the proposed model.

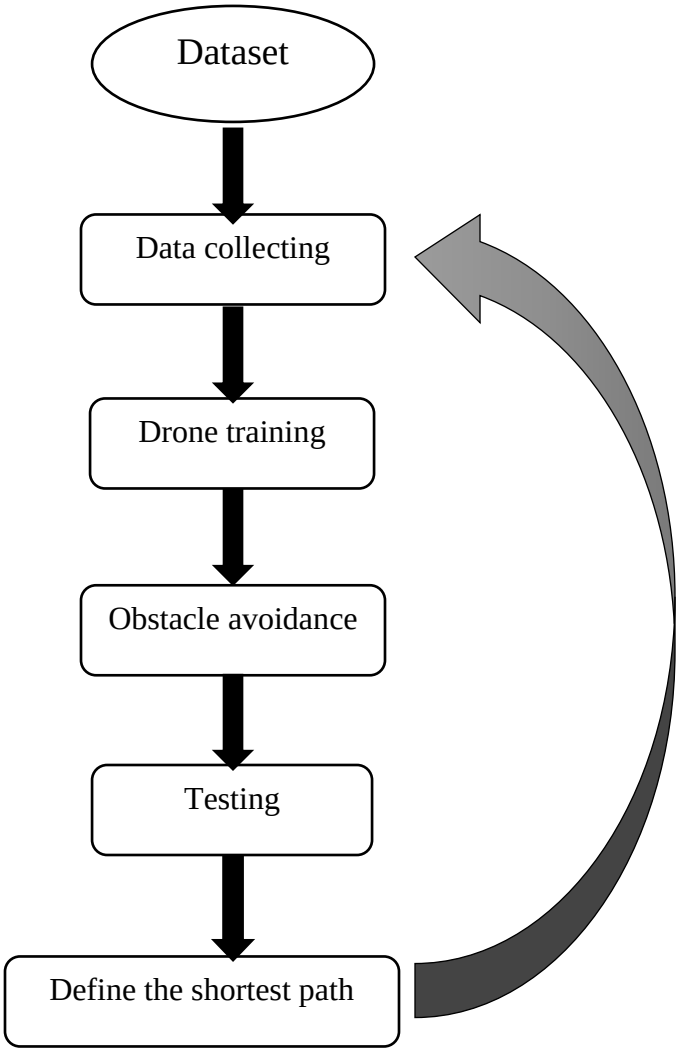


Figure 4. 2 General architecture of the proposed model

4.2.3.1 Data collecting

The drone starts taking pictures and collect it. Due to these pictures, drone will be able to get data from this environment. Figure 4.3 presents a sample of the collecting pictures

Figure 4.3 Set of pictures collected by the drone

4.2.3.2 Drone training

In this step, drone needs to learn about its environment, consequently we have to train it in a specific environment in order to get the maximum data which are a set of obstacles coordination points. Figure 4.4 shows the Drone training operation.

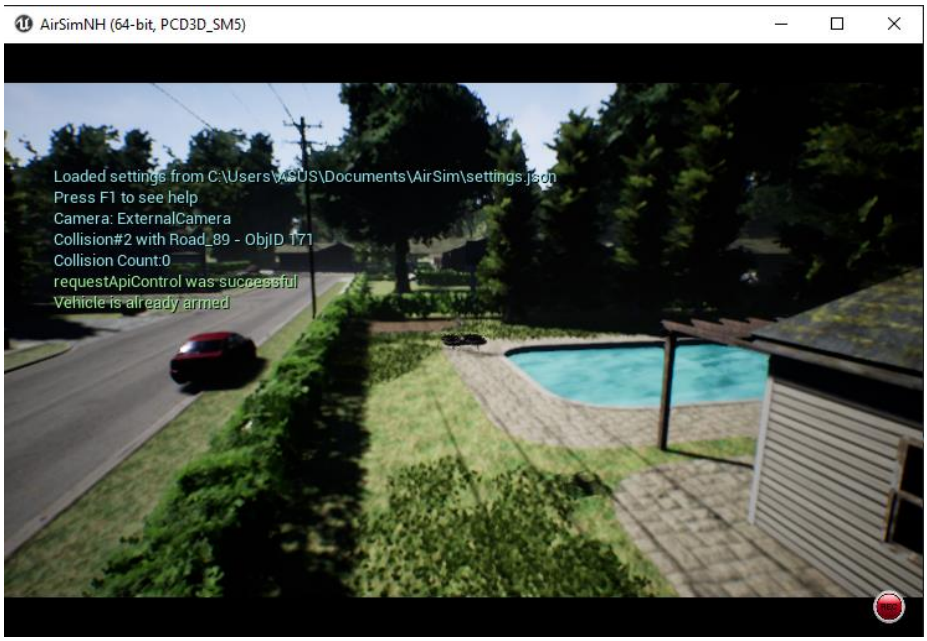


Figure 4.4 Drone training

The training drone and collected data allows the drone to learn how to avoid obstacle (collision) by using a simple variable “Safety”. If Safety take value great than 0.5 then the path is safe else the path is wrong.

4.2.3.3 Testing:

In this phase, we will give the drone the start points and end point, then the application can execute many tests using A* (considering the obstacles and the path taken).

4.2.3.4 Define the shortest path:

After the step of the test, our application can define the shortest path from the drone point to goal one. Then, the drone will follow this path which is the shortest one.

4.3 Implementation

In this section, we will explain the most important methods. In this stage, we cannot explain all the methods and classes. But, we give the main classes and methods for our proposed design:

4.3.1 Image collection:

This class has the role of collecting pictures, where it collects the pictures before the drone crushed into collision while the other pictures get ignored.

Figure 4.5 shows the method which used to calculate the distance between the start point and the next point (the goal)

```
def distance(point, goal):
    return mt.sqrt((point.getX() - goal.getX())**2 + (point.getY() - goal.getY())**2 + (point.getZ() - goal.getZ())**2)
```

Figure 4.5 Method of calculating the distance

Figure 4.6 shows the code which used to save the pictures before the crushing of drone. It can save these pictures in a folder called “carpix”.

```
# Where we'll store images
IMAGEDIR = './carpix'

# Create images directory if it doesn't exist
def loadgray(filename):
    '''
    Loads an RGBA image from FILENAME, converts it to grayscale, and returns
    '''

    image = plt.imread(filename)

    # RGB -> gray formula from https://www.johndcook.com/blog/2009/08/24/algo
    image = 0.21 * image[:,:,0] + 0.72 * image[:,:,1] + 0.07 * image[:,:,2]
    image = image[0::SCALEDOWN, 0::SCALEDOWN]
    image = image.flatten()

    return image
```

Figure 4.6 Saving the pictures before crushing

Figure 4.7 presents the code which used to get the collision information that is a set of coordinates points (X,Y, Z).

```
collision_info = client.simGetCollisionInfo()
xD = client.getMultirotorState().kinematics_estimated.position.x_val
yD = client.getMultirotorState().kinematics_estimated.position.y_val
zD = client.getMultirotorState().kinematics_estimated.position.z_val
maxLimit = 200
print("collision === ",collision_info.has_collided)

if collision_info.has_collided:
    print("Collision at pos %s, normal %s, impact pt %s, penetration %f, name %s, obj id %d" % (
        pprint.pformat(collision_info.position),
        pprint.pformat(collision_info.normal),
        pprint.pformat(collision_info.impact_point),
        collision_info.penetration_depth, collision_info.object_name, collision_info.object_id))
    break
```

Figure 4.7 Getting the collision coordinates points.

4.3.2 Collision training

This class uses the store pictures in the previous step. We use these pictures to train our model which use the artificial neural network to detect collisions. For detecting the collisions, we follow these steps:

- First of all, our application reads the saved pictures and classifies them one by one based on safety variable. Figure 4.8 presents the code which used to read the pictures and classify them.

```
def main():

    # This will get number of pixels in each image (they must
    imgsize = 0

    # Read in images from drone, convert to grayscale, scale d
    images = []
    for k in range(SAFESIZE):

        image = loadgray(IMAGEDIR + '/image%03d.png' % k)

        imgsize = np.prod(image.shape)
        images.append(image)

    # All but last image is safe (01 = no-crash; 10 = crash)
    targets = []
```

Figure 4.8 Reading pictures and classify them.

- When the drone goes to the wrong direction, the ratio of safety decreases till it crashes. Figure 4.9 presents the code used to manipulate the safety ratio.

```
def getSafety(responses):
    # Save it to a temporary file
    image = responses[0].image_data_uint8
    AirSimClientBase.write_file(os.path.normpath(TMPFILE), image)

        # Read-load the image as a grayscale array
    image = loadgray(TMPFILE)
    safety = sess.run(output, feed_dict={x:[image]})[0][1]
    return safety
```

Figure 4.9 The safety ratio

4.3.3 A* algorithm

In this section, we will explain the different steps used to applicating the A* algorithm. For this, we to follow the following points:

4.3.3.1 Calculating the distance

The main goal of this method is to calculate the distance between the current points and its neighbor. We use the radium of the ball coordinates in which the drone is contained for calculating these distances.

- Figure 4.10 shows the code used to get the neighbor coordinates of current point by using many parameters as current point, radium.

```
def neighbor_nodes(self, point, r, maxS):
    neib = []
    neib.append(Point(point.getX() + r, point.getY(), point.getZ()))
    neib.append(Point(point.getX() - r, point.getY(), point.getZ()))
    neib.append(Point(point.getX(), point.getY() + r, point.getZ()))
    neib.append(Point(point.getX(), point.getY() - r, point.getZ()))
    #neib.append(Point(point.getX(), point.getY(), point.getZ() + r))
    #neib.append(Point(point.getX(), point.getY(), point.getZ() - r))
```

Figure 4.10 The neighbor's points coordinates

- Figure 4.11 presents the method used to calculate the distance between current point and its neighbors using scalar function.

```
def distance_between_neighbors(self, point, goal):
    return mt.sqrt((point.getX() - goal.getX())**2 + (point.getY() - goal.getY())**2 + (point.getZ() - goal.getZ())**2)
```

Figure 4.11 Method of calculating the distance between neighbors

4.3.3.2 Finding the goal zone

This method finds out the zone of the goal by using the current point, the goal and the radium. When the distance between the coordinates of the current point and the goal less than the radium, therefore that's the zone of the goal. Figure 4.12 presents the code which used to find the goal zone.

```
def is_goal(self, point, goal, r):
    valx = np.abs(point.getX() - goal.getX())
    valy = np.abs(point.getY() - goal.getY())
    return valx <= r and valy <= r
    #return self.heuristic(point, goal) < r #point == goal
```

Figure 4.12 Finding the goal zone

4.3.3.3 Finding of the paths

This is the desired goal which we are looking for. This class allows to find the paths, based on the star and end point.

- First of all, it divides the path into three parts, all parts have the same distance. Figure 4.13 presents the code used to divide the finding path into three parts.

```
while True:

    next_f, tempPoint = open_queue.pop(0)
    vx = (1/3)*(tempPoint.getX() - point.getX())
    vy = (1/3)*(tempPoint.getY() - point.getY())
    vz = 0.0 #(1/3)*(tempPoint.getZ() - point.getZ())
```

Figure 4.13 Division of the distance between the point and the neighbor

- Then, from the starting point, the function tests if this path is safe or not. After that, our application measures the ratio of the safety, if it is more than 0.5 then drone will continue on the same path, else the drone comeback to the old position and remove the wrong path. Figure 4.14 shows the code used to test the safety of the path.

```
if safety > 0.5:

    quad_vel = client.getMultirotorState().kinematics_estimated.linear_velocity
    client.moveByVelocityAsync(vx, vy, vz, 2).join()
    xD = client.getMultirotorState().kinematics_estimated.position.x_val
    yD = client.getMultirotorState().kinematics_estimated.position.y_val
    zD = client.getMultirotorState().kinematics_estimated.position.z_val
    print("(",xD,",",yD,",",zD,")")
    #client.moveByVelocityAsync(quad_vel.x_val + vx, quad_vel.y_val + vy, quad_vel.z_val + vz, 0).join()
    time.sleep(0.5)
    point = tempPoint
    break
else:
    open_set.remove(tempPoint)
    vx = (-1/3)*(tempPoint.getX() - point.getX())
    vy = (-1/3)*(tempPoint.getY() - point.getY())
    vz = 0.0 #(-1/3)*(tempPoint.getZ() - point.getZ()) #+ point.getZ()
    quad_vel = client.getMultirotorState().kinematics_estimated.linear_velocity
    client.moveByVelocityAsync(vx, vy, vz, 1).join()
    #client.moveByVelocityAsync(quad_vel.x_val + vx, quad_vel.y_val + vy, quad_vel.z_val + vz, 0).join()
    time.sleep(0.5)
```

Figure 4.14 The safety testing

- Thus, our application saves the coordinates of the path in a specific file. Figure 4.15 presents the code used to save the coordinates of the safe path.

```
def storePathInFile(path):
    f = open("path.txt", "w")
    for p in path:
        s = str(p.getX())+","+str(p.getY())+","+str(p.getZ())+"\n"
        f.write(s)
        f.flush
    f.close()

def getFromFile():
    path = []
    f = open("path.txt", "r")
    for line in f:
        crd = line.split(",")
        path.append(Point(float(crd[0]), float(crd[1]), float(crd[2])))
    f.close()
    return path
```

Figure 4.15 Saving of the safe path coordinates

4.3.3.4 Implementation of A* algorithm

- In the previous step, our application saves the coordinate of the safe path in specific file. Thus, our algorithm A* can read this file using the code showed in Figure 4.16.

```
def getFromFile():
    path = []
    f = open("path.txt", "r")
    for line in f:
        crd = line.split(",")
        path.append(Point(float(crd[0]), float(crd[1]), float(crd[2])))
    f.close()
    return path
```

Figure 4.16 The safe paths.

- Then, our application chooses the shortest path followed by the drone using the A*. Figure 4.17 presents the code of A* in python.

```
def find_path(problem, start, goal, client):
    open_set = set()
    open_queue = list()
    closed_set = set()
    came_from = dict()
    g_score = dict()
    h_score = dict()

    def f_score(point):
        return g_score[point.getId()] + h_score[point.getId()]

    g_score[start.getId()] = 0
    h = problem.heuristic(start, goal)
    h_score[start.getId()] = h
    open_set.add(start)
    open_queue.append( (f_score(start), start) )
    tempPoint = start
    point = Point(start.getX(), start.getY()-1, start.getZ())
    problem.on_open(start, h, 0, h)

    while open_set:
        open_queue.sort(key=comp)
        # ------------ neural network code
        safety = 0
        # ------------ end neural network code
        while True:
```

Figure 4.17 Choose of the shortest path.

- Finally, our application sends this shortest path to drone for following it. Figure 4.18 present the code which is executed by Drone to follow the shortest path defined by our application.

```
    find_path(prob, start, goal, client)
    print("A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A-path found and stored-*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*A*")
else:
    #print("path found ************************************************************")
    for i in range(0, len(path) - 1):
        frm = path[i]
        to = path[i + 1]

        vx = to.getX() - frm.getX()
        vy = to.getY() - frm.getY()
        vz = to.getZ() - frm.getZ()
        quad_vel = client.getMultirotorState().kinematics_estimated.linear_velocity
        client.moveByVelocityAsync(vx, vy, vz, 1).join()
        #client.moveByVelocityAsync(quad_vel.x_val + vx, quad_vel.y_val + vy, quad_vel.z_val + vz, 0).join()
        time.sleep(0.5)
    client.moveToPositionAsync(goal.getX(), goal.getY(), 0, 1)#.join()
    #client.moveByVelocityAsync(0, 0, 20, o).join()
    while True:
        ci = client.simGetCollisionInfo()
        if ci.has_collided:
            print(ci.has_collided)
            break
        #client.enableApiControl(False)
    qs = client.getMultirotorState().kinematics_estimated.position
```

Figure 4.18 Following the shortest path

4.4 Testing and Results

Our work was tested in a PC with Intel i7 2.60 GHz CPU, NVidia GeForce GTX 950M for the GPU and 8 GB for the RAM. In order to evaluate the efficacy of our work, we need to check it based on two main criteria which are the obstacle avoidance and the shortest path.

4.4.1 Evaluation of efficacity according to the obstacle avoidance

For this evaluation, we use three Test.

4.4.1.1 Test 1

In the first test, we forced the drone to crushed many times in random collisions. Figure 4.19 presents drone in many collisions.

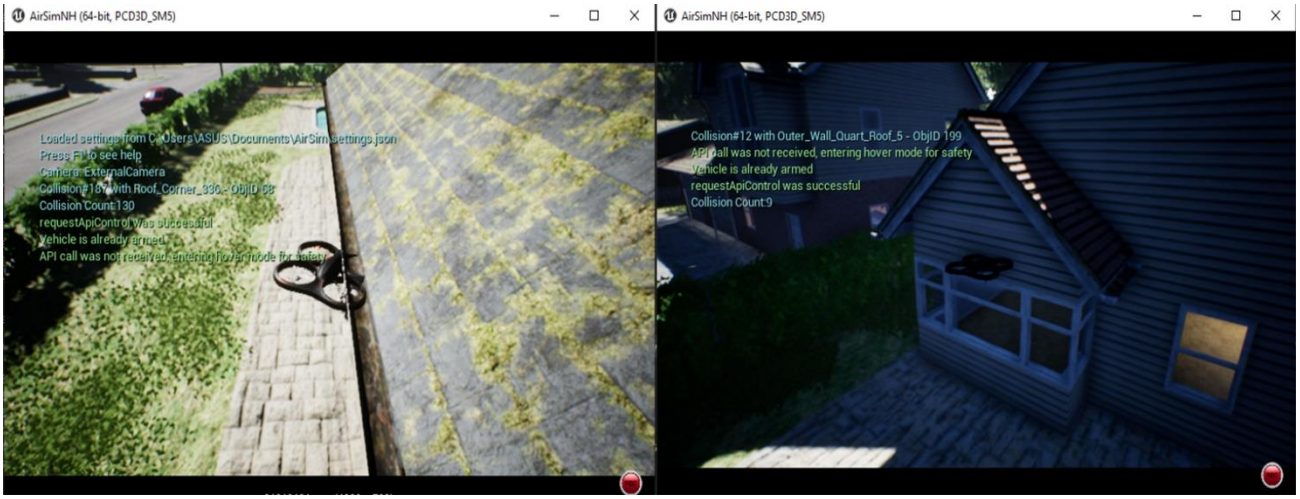


Figure 4.19 Drone collision.

4.4.1.2 Test 2

In this test, the number of collisions decreases compared to the test 1 because our drone was trained in the Test1. So, our Drone avoids more than three collisions, but it still get crushed in some other obstacles.

4.4.1.3 Test 3

After two tests, drone get more training which allow it to avoid the most obstacles.

4.4.1.4 Comparison of the results

The **Table 3.1** summary the results of the three tests:

Table 4.1 Summary of the obstacle avoidance results.

Tests	Number of obstacles avoided
Test 1	3/10
Test 2	5/10
Test 3	9/10

4.4.2 Evaluation of the shortest path

In this section, we evaluate the path found by our application over two tests.

4.4.2.1 Test 1

In this test, our application defines the optimal path but it is not the shortest one. Thus, it needs also some improvement to be the shortest path. Figure 4.20 present the coordinate points of the path defined by our application.

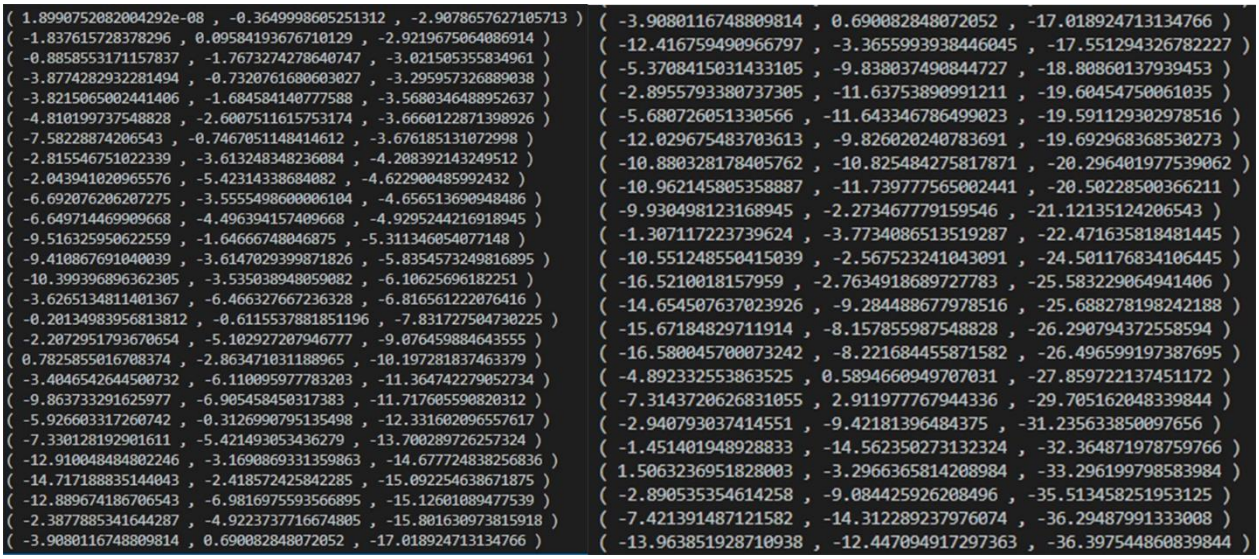


Figure 4.20 The coordinate point of the path defined by our application.

4.4.2.2 Test 2

In this test, our application detects path which is shorter than the path detected in the test1. Thus, this is the shortest path. Figure 4.21 present the coordinate points of the shortest path

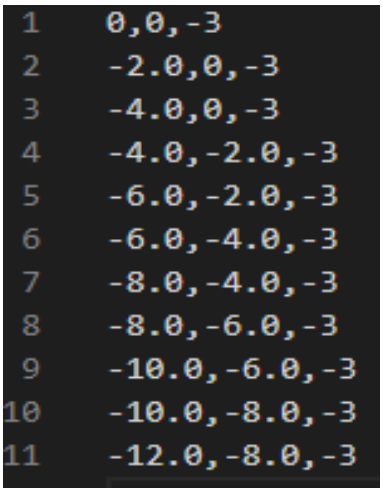


Figure 4.21 The coordinate points of the shortest path.

4.4.2.3 Comparison of the results

The difference between the results of the first and the second test means that the A* algorithm is working correctly, in which the defined path in the second test is shortest than the path defined in the second one.

4.5 Discussion

Based on the results of the first evaluation, the method that we have used for the obstacle avoidance was very effective. After the training of our drone many times, we have always obtained improved results compared to the previous results. This improvement, shows that our application is run correctly.

The obtained results in the second evaluation proves that we choose the most appropriate algorithms (A*) because we defined the shortest path in two steps.

4.6 Conclusion

In this chapter, we explain the implementation, in which all the main classes and methods of our work was presented. These methods were fair enough to implement our proposed model. Also, we describe the programing languages which is Python. The plenty of libraries of the choice programming language allowed us to perform many methods automatically. The results are very encouraging to continue in this project.

General conclusion

General conclusion

Computer vision is playing a key role in detecting the various types of objects while flying in midair. We use neural network to detect a lot of types of objects such as vehicles (car, drones ..etc), buildings, trees .. etc. These objects could be an obstacle for our Drone, which requires applying the obstacle detection and collision avoidance methods.

In our work, we propose a new architecture, which permits to drone to avoid collision and detect the shortest path rapidly without the human's intervention. For this, we use ANN algorithm for training our Drone for detecting the obstacle and we use A* algorithm for detecting the shortest path.

At the beginning, the objective was to construct a real drone, which can avoid the obstacles and define the shortest path but because of the COVID-19, we could not get the different parts of the Drone. For this reason, we use the simulation platform AirSim. Generally, the obtained results are encouraging, but we face some difficulties especially in the implementation phase, which are:

- Many required libraries.
- High system requirements for the simulator.
- The training phase takes a lot of time because of these requirements.

As a future work, we can cite many works, which can improve the performance of our system as:

- Train it in different environments.
- Implement it with other optimization algorithms (such as Genetic algorithm or Particle Swarm) and compare their results.
- Design real drone.

Bibliography

[1] Dormehl, Luke, (2018), “The History of Drones in 10 Milestones.”. https://www.digitaltrends.com/cool-tech/history-of-drones/.

[2] “Classification of the Unmanned Aerial Systems | GEOG 892: Unmanned Aerial Systems.” n.d. (Accessed September 2, 2020). https://www.e-education.psu.edu/geog892/node/5

[3] Shin, Sang-Yun, Yong-Won Kang, and Yong-Guk Kim, (2020), “Reward-Driven U-Net Training for Obstacle Avoidance Drone.” Expert Systems with Applications, 143, https://doi.org/10.1016/j.eswa.2019.113064.

[4] N. F., Xavier & L. PAGLAN, (2017), Conception d’un drone à reconnaissance faciale, 98, Université de Douala, Cameroun.

[5] Tung-Cheng Wu, Tung-Cheng, Shau-Yin Tseng, Chin-Feng Lai, Chia-Yu Ho, and Ying-Hsun Lai, (2018), “Navigating Assistance System for Quadcopter with Deep Reinforcement Learning.” International Cognitive Cities Conference (IC3), 16–19. Okinawa, Japan: IEEE. https://doi.org/10.1109/IC3.2018.00013

[6] Yoo, Sang-Jo, Jae-hyun Park, Su-hee Kim, and Anish Shrestha. (2016). “Flying Path Optimization in UAV-Assisted IoT Sensor Networks.” ICT Express 2 (3), 140–44. https://doi.org/10.1016/j.icte.2016.08.005

[7] Dorigo, Marco, and Thomas Stützle, (2003), “The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances.” In Handbook of Metaheuristics, edited by Fred Glover and Gary A. Kochenberger, 57:250–85. International Series in Operations Research & Management Science. Boston, MA: Springer US. https://doi.org/10.1007/0-306-48056-5_9.

[8] Weisstein, Eric W. n.d. “Ant Colony Algorithm.” Text. Wolfram Research, Inc. Accessed (September 22, 2020). https://mathworld.wolfram.com/AntColonyAlgorithm.html

[9] Qinghai, Bai, (2010), “Analysis of Particle Swarm Optimization Algorithm,” February.

[10] Chatterjee, Marina, (2020), “A* Search Algorithm in Artificial Intelligence | A* (Star) Algorithm in AI.”, GreatLearning. https://www.mygreatlearning.com/blog/a-search-algorithm-in-artificial-intelligence/.

[11] Mohammed, Mohssen, Muhammad Badruddin Khan, and Eihab Bashier Mohammed Bashier, (2016), Machine Learning: Algorithms and Applications. 0 ed. Boca Raton : CRC Press: CRC Press. https://doi.org/10.1201/9781315371658.

[12] Ian Goodfellow, Yoshua Bengio and Aaron Courville: http://www.deeplearningbook.org/

[13] Khosrow-Pour, D.B.A., Mehdi, ed, (2015), Encyclopedia of Information Science and Technology, Third Edition: IGI Global. https://doi.org/10.4018/978-1-4666-5888-2.

[14] Nor, Abu Hassan Shaari Md, Tamat Sarmidi, and Ehsan Hosseinidoust, (2014), "Forecasting of Palm Oil Price in Malaysia Using Linear and Nonlinear Methods." In , 138–52. Sarawak, Malaysia. https://doi.org/10.1063/1.4894340.

[15] "Que signifie Réseau neuronal convolutif?, Définition IT de Whatis.fr." n.d. LeMagIT. Accessed (September 11, 2020). https://www.lemagit.fr/definition/Reseau-neuronal-convolutif.

[16] Phung, and Rhee, (2019), "A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets." Applied Sciences 9 (21): 4500. https://doi.org/10.3390/app9214500.

[17]. ("Get to Know the Concepts." n.d). Accessed (August 26, 2020). https://aischool.microsoft.com/en-us/autonomous-systems/learning-paths/airsim-lab/airsim-introduction/airsim-introduction-get-to-know-the-concepts.

[18]. "Artificial Neural Networks Advantages and Disadvantages." n.d. Accessed (September 27, 2020). https://www.linkedin.com/pulse/artificial-neural-networks-advantages-disadvantages-maad-m-mijwel.

[19]. "Memoire Online - Identification et Commande Des Systèmes Non Linéaires - LEMMOU Amira- BELLAKHDAR Khaoukha- LEDJEDEL Adila." n.d. Accessed (September 27, 2020). https://www.memoireonline.com/04/12/5750/m_Identification-et-commande-des-systemes-non-lineaires21.html