UAV Path Planning and Obstacle Avoidance Based on Fuzzy Logic and Kinodynamic RRT Methods

Long Chen

A Thesis

In

The Department

of

Mechanical, Industrial & Aerospace Engineering

Present in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Mechanical Engineering) at

Concordia University

Montréal, Québec, Canada

March 2021

CONCORDIA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: **Long Chen**

Entitled: **UAV Path Planning and Obstacle Avoidance Based On Fuzzy Logic and Kinodynamic RRT Methods**

And submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Mechanical Engineering)

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Youmin Zhang

_____ External Examiner
Dr. Rastko Selmic

_____ Examiner
Dr. Youmin Zhang

_____ Thesis Supervisor
Dr. Wenfang Xie and Dr. Iraj Mantegh

Approved by _____
Dr. Siva Narayanswamy Graduate Program Director

_____ _____
Date of Defence Dr. Mourad Debbabi Interim Dean, Faculty of Engineering and Computer *Science*

Abstract

UAV Path Planning and Obstacle Avoidance Based On Fuzzy Logic and Kinodynamic RRT Methods

Long Chen, MASc.

Concordia University, 2021

Path Planning is one of the important problems to be explored in unmanned aerial vehicle (UAV) to find the optimal path between starting position and destination. The aim of path planning technique is not only to find the shortest path but also to provide the collision-free path for the UAV in unknown environment. Although there have been significant advances on the methods of path planning where the map of environment is known in advance, there are still some challenges to be addressed for dynamic autonomous navigation for the UAV in unknown environment.

This thesis research proposes a new path planning method named Fuzzy Kinodynamic RRT for unmanned aerial vehicle flying in the unknown environment. This method generates a global path based on RRT [1] (Rapidly-exploring random tree) and utilizes fuzzy logic system to avoid obstacles in real time. A set of heuristics fuzzy rules are designed to lead the UAV away from un-modeled obstacles and to guide the UAV towards the goal. The rules are also tested in different scenarios, and they are all working efficiently both in simple and complicated cases. The UAV starts to fly along the path generated by RRT, and the fuzzy logic system is then activated when it comes across the obstacle. When the sensor detects no collision within a specific distance, the fuzzy system is turned off and the UAV flies back to the previous path towards the final destination. The simulations of the developed algorithm have been carried out in various scenarios, with the sensor to detect the obstacles. The numerical simulations show the satisfactory results in various scenarios for path planning that considerably reduces the risk of colliding with other stationary and moving obstacles. A more robust and efficient fuzzy logic controller which embeds the path planning is finally proposed and the simulation shows the satisfactory results in complicated environments.

Acknowledgments

This thesis is submitted in partial fulfillment of the requirements for the degree of Master of Applied Science (MASc) at Concordia University (CU) under the financial support of National Research Council Canada's CivUAS program and IAM (Integrated Autonomous Mobility) initiative, and also the NSERC (Grant No. N00892). This thesis research has been conducted under the supervision of Prof. Wenfang Xie and Dr. Iraj Mantegh at the Department of Mechanical, Industrial & Aerospace Engineering, Concordia University and Canada National Research Council (C-NRC) respectively.

My deepest gratitude and appreciation go to my supervisor, Prof. Wenfang Xie and Dr. Iraj Mantegh, for offering me this valuable and unique opportunity to pursue my Master degree under their supervision. Their serious attitude towards research has taught me quite a lot. Without their consistent support and great help as well as valuable and insightful guidance, my research progress would probably be slowed and unsatisfactory. There are no words that can express my sincere gratitude for their outstanding supervision.

I would like to thank for joining my examination committee and providing brilliant feedback and insightful comments during my comprehensive exam and research proposal exam. I would also like to thank all my colleagues in my research group and collaborators who have helped me a lot during my two years and a half master study. It is my great pleasure to do research and share opinions with them and learn from them.

Contents

List of Figures

List of Tables

GLOSSARY

Explanation of Terms

UAV Unmanned aerial vehicle

ESC Electronic Speed Controllers

NRC National Research Council

IAM Integrated Autonomous Mobility

RRT Rapidly-Exploring Random Tree

PRM Probabilistic Roadmaps

A* A Star Algorithm

ROS Robotics operation system

IMU Inertial Measurement Unit

GPS Global positioning system

VTOL Vertical takeoff and landing

PID Proportional-integral-derivative

UN Unreal Engine

AirSim Open Source Platform in Microsoft

IoT Internet of things

Chapter 1 Introduction

1.1 Background

Unmanned Aerial Vehicle (UAV) is essentially a flying robot which can be remotely controlled or fly autonomously through software controlled flight plans in their embedded systems, working in conjunction with the sensors which are onboard instruments and GPS. UAVs contain a large number of technological components including electronic speed controllers (ESC), flight controller, GPS module, antenna, battery, receiver, cameras and sensors such as ultrasonic sensors and collision avoidance sensors, accelerometer and altimeter which are used to measure speed and altitude respectively. While UAVs serve a variety of purposes, such as recreational, photography, commercial and military, their two basic functions are flight and navigation.

UAV platforms are classified into two main types: rotor, including single-rotor and multi-rotor such as quadcopters, or fixed-wing which includes the VTOL (vertical takeoff and landing) drones that do not require runways. Single-rotor drones are the most basic types and they are ideal for longer flight times since it employs only a single rotor (besides the tail unit in some cases) and can often generate thrust more efficiently than their multi-rotor counterparts. Multi-rotor drones have several rotors positioned at strategic point on the drone and these extra rotors make it easier for the drone to maintain the balance while hovering. Fixed-wing style drones are more similar to controllable airplanes rather than the helicopter style of other drones. Unlike rotors, their wings provide vertical lift, which means they only need enough energy to keep moving forward, making them ideal long-range drones. Different drones have different kinds of advantages and disadvantages as follows:

1) Single rotor drones. As shown in Figure 1.1(a).
   - Able to hover vertically in the air
   - Have long-lasting flight time
   - Strong and durable
   - Harder to fly than multi-rotor drone types
   - Can be expensive and have higher complexity

2) Multi-rotor drones. As shown in Figure 1.1(b).
   - Easy control and maneuver
   - Very stable
   - Can take off and land vertically
   - Limited flying time (Usually 15-30 minutes)
   - Only have small payload capabilities

3) Fixed-wing drones. As shown in Figure 1.1(c).
   - Flight time is long
   - Can fly at a high altitude
   - Have the ability to carry more weight
   - More difficult to land than two other categories of drones
   - Can only move forward instead of hovering in the air

(a) (b) (c)

Figure 1.1 Single rotor drone (a), multi-rotor drone (b) and fixed-wing drone (c)

Based on the applications, UAVs can also be categorized as military, commercial and personal applications.

1) In the recent past, UAVs were most often associated with the military, where they were used initially for anti-aircraft target practice, intelligence gathering and then, more controversially, as weapons platforms. Drones are now also used in a wide range of civilian roles ranging from search and rescue, surveillance, traffic monitoring, weather monitoring and firefighting, to personal drones and business drone-based photography, as well as videography, agriculture and even delivery services.
2) The integration of drones and internet of things (IoT) technology has created numerous enterprise use cases. Drones working with on-ground IoT sensor networks can help agricultural companies monitor land and crops; energy companies survey power lines and operational equipment; and insurance companies monitor properties for claims and policies.
3) Many personal drones are now available for consumer use, offering high definition (HD) video or still camera capabilities, or to simply fly around. These drones often weigh anywhere from less than a pound to 10 pounds.

UAVs can be equipped with a number of sensors, including distance sensors (ultrasonic, laser, lidar), time-of-flight sensors, chemical sensors, and stabilization and orientation sensors, among others. Visual sensors offer still or video data, with RGB sensors which is the metering sensor that helps the camera analyze the scene and collect standard visual red, green and blue wavelengths, and multispectral sensors collecting visible and non-visible wavelengths, such as infrared and ultraviolet. Accelerometers, gyroscopes, magnetometers, barometers and GPS are also common drone features.

For example, thermal sensors can be integral in surveillance or security applications, such as livestock monitoring or heat-signature detection. Hyperspectral sensors can help

identify minerals and vegetation, and are ideal for use in crop health, water quality and surface composition. Some drones employ obstacle detection and collision avoidance sensors. Initially, the sensors were designed to detect objects in front of the drone. Some drones now provide obstacle detection in all six directions: front, back, below, above and side to side. For the purpose of landing, the drones employ visual positioning systems with downward facing cameras and ultrasonic sensors. The ultrasonic sensors determine how close the drone is to the ground.

In Canada, Transport Canada has launched an RTM (RPAS Traffic Management) initiative and organized an action team (RTMAT) comprised of the regulator, Nav Canada, NRC, and various industry representatives. In addition, the National Research Council (NRC) is collaborating with Transport Canada (TC) to develop a 5-year R&D plan to support regulatory development for visual line-of-sight (VLOS)/beyond visual-line-of-sight (BVLOS) remotely piloted aircraft systems (RPAS) operations and to identify technology advancements, testing and certification that will enable safe operation of RPAS (also known as drone, UAS or UAV) in Canada. The objectives are to develop and oversee the Government of Canada's transportation policies and programs so that Canadians can have access to a transportation system that is safe and secure; green and innovative; and efficient. The Consortium for Aerospace Research and Innovation in Canada (CARIC) and the Consortium for Research and Innovation in Quebec (CRIAQ) will support Transport Canada and NRC to deploy a RPAS R&D Program.

Over the years, autonomous vehicles have been used to perform missions which are dangerous and dirty such as military operations and wild-fire surveillance. It becomes crucial for the UAV to react dynamically towards changing environments and missions especially in these tasks. For example, forest fires can change rapidly because the environment conditions are changing and hard to predict with time going on. Therefore, real time decision making becomes essential when dealing with wildfires since conditions such are wind are changing dynamically.

Because most effective path planning algorithms such as RRT, PRM, A* and Artificial potential fields [2] rely on complete prior knowledge of the whole environment, it severely limits the implementations and capabilities of UAVs. Additionally, theses algorithms cannot obtain solutions for dealing with complex environments. Rapidly-exploring random tree algorithm and its variants, for instance, cannot guarantee a complete and accurate path in any environments. Furthermore, the UAV always requires new paths re-generated by these algorithms when environments changing dynamically in real time. Therefore, a more sophisticated method to two-dimensional path planning is needed for usage of UAVs in dynamically changing, unknown environments. In order to do this, UAVs have to do path planning and obstacle avoidance in dynamic sense. Thus, the objective of this research is to propose new methods of path planning and obstacle avoidance for an UAV that can allow it to operate in real time and unknown environments.

1.2 Problem Definition

The research will focus on the path planning and real time obstacle avoidance of the UAV in unknown environments in order to meet the challenges that most effective path planning algorithms are not applicable in these cases where they severely rely on complete prior knowledge of the whole environment. With the methods proposed in our research, only a limited amount of information (GPS coordinates of the starting point and target) is required for autonomous navigation of the UAV.

In this work, a sensor system is used to provide feedbacks about the obstacles in the UAVs local environment within sensing range. The information is then processed at each sampling time and used as one of the factors to determine the motion of the UAV for path planning and obstacle avoidance in order to reach the target.

Path planning and obstacle avoidance play a big role in the application of UAVs where UAVs fly from the starting point to destination and avoid the obstacles dynamically and efficiently. Currently, there are currently a lot of classic and variant methods that have been implemented on UAVs, and each method has its own pros and cons according to the specific application situation. Thus, it is very essential to utilize an appropriate algorithm or methodology in order to improve the efficiency in each specific application. In this research, the issues on global path planning will be addressed by RRT algorithm which can give an overall planned path for the UAV to follow. Then a fuzzy logic inferencing system is designed to drive the UAV to avoid the obstacle in real time, and a novel algorithm named Fuzzy Kinodynamic RRT method is developed to guide the UAV towards the goal. Finally, a sophisticated and robust fuzzy controller is designed especially for the path planning and obstacle avoidance for UAVs in more complicated and unknown environments. The research modeled the dynamic constrains of the UAV, developed methods for path planning and obstacle avoidance with only sensor information about the local environments and validated the proposed methods by simulation in MATLAB and AirSim in Unreal Engine.

1.3 Motivation and Contribution of this Thesis

In this research, tremendous efforts have been dedicated to the path planning and obstacle avoidance of UAVs and to improve the performance of the autonomous navigation for the UAVs. The objectives of this research are to implement an efficient UAV path planning and obstacle avoidance method in unknown environments and real-time collision avoidance using distance sensor. Even though extensive algorithms have been developed for mobile robots [3], the applications to UAVs in dynamically changing environment conditions are limited. The main contributions of this work can be summarized as follows:

1) A fuzzy inferencing system is developed for supporting the UAV to avoid obstacle dynamically in unknown environment. This fuzzy system consists of two inputs and one output. The rules for different inputs and output for the fuzzy logic inference are set up in the form of "IF-THEN" statements, and are based on heuristics and human experience with navigating through an environment, which is similar to driving a car.

2) Fuzzy-Kinodynamic RRT is a novel method which uses RRT algorithm to do global path planning and utilizes fuzzy logic system to avoid obstacles. The UAV starts to follow the path generated by global path planning algorithm and the fuzzy logic system is activated when it comes across new obstacles. The UAV can avoid obstacles dynamically according to the rules designed in this research work and then fly back to the previous path.
3) A more sophisticated and robust fuzzy logic controller with four inputs, two outputs and totally 40 fuzzy logic rules is designed for dynamically path planning and obstacle avoidance in unknown environments without the support of global path planning as implemented in Fuzzy Kinodynamic RRT method.
4) This dissertation proposes an algorithm on the combination of UAV global path planning and sensor-based real-time obstacle avoidance, and validates the effectiveness of this method through simulation mainly on Unreal Engine AirSim platform.

The main advantages over other methods of this work can be summarized as follows:

(i) This system does not need a priori environment information, and it works well with very limited information.

(ii) It can continue planning the path towards the target while avoiding obstacles efficiently.

(iii) The target and obstacles can either be stationary or moving.

(iv) The methods are fast and applicable with only onboard sensors, and no camera is required.

1.4 Organization of this Thesis

This thesis is structured in the following manner:

- Chapter 2 mainly introduces the literature survey about global path planning methods including sampling-based and search-based algorithms for quadrotors. Besides, real time obstacle avoidance methods are also presented in this chapter.

- Chapter 3 describes the hardware components and software used in this thesis, such as Unreal engine, AirSim and MATLAB. The selection of quadrotors with distance sensors or camera is also included in this chapter.

- Chapter 4 illustrates the fuzzy logic inferencing design part including control system, problem modeling, membership functions and rules design. Moreover, the combination of global path planning and obstacle avoidance method fuzzy kinodynamic RRT and its flowchart are also presented. Finally, a more sophisticated and robust fuzzy logic controller is designed for efficient obstacle avoidance in more complicated environments without global path planning as a prior trajectory for the

UAV.

- Chapter 5 presents the test and simulation results on each algorithm in different scenarios, including the initial fuzzy logic controller, Fuzzy Kinodynamic RRT and the improved fuzzy logic controller.

- Chapter 6 presents the conclusions of the presented research works and summarizes several predominant ideas for the future development of the dissertation's outcomes.

Chapter 2 Literature Survey

This chapter will give a comprehensive literature survey on global path planning and local path planning methods for the quadrotor in real time obstacle avoidance.

2.1 Global Path Planning

Global path planning [4] is a type of path planning methods to design an offline path from the current position to a target position while avoiding obstacles given a known map. Local path planning [5], on the other hand is referred to as the methods that take in information from the surroundings in order to generate a simulated field where a path can be found. This allows a path to be found in the real-time as well as adapting to dynamic obstacles.

Path planning has been widely studied in many fields, such as unmanned aerial vehicle, mobile robots and autonomous road vehicles. Many motion planning approaches have been presented over the past decades. There are some well-known path planning algorithms such as RRT, Potential field method, A* and D* to realize obstacle avoidance.

One of the popular path planning methods in the presence of stationary and moving obstacles is the artificial potential field [6] [7]. Charles W. Warren [8] developed an artificial potential field technique. However, it can only give one route solution in a static environment which may not be the best path with the shortest distance to the goal. Jianli Yu [9] illustrated a method of potential field based on 3D path planning where the shape and positions of collisions are known. However, this method cannot be implemented on path planning in unknown environment and the path needs to be designed beforehand. Another method is position estimation method which can generate a path between current location and the desired location. The drawback is that this method will accumulate estimation error [10] [11] during the whole process. Genetic algorithm [12] [13] is also

applied to solve path planning problem which is a search technique analogous to natural evolution [14]. The path planning process is repeated over and over again and the population is evolved generation by generation. Toogood et al [15] developed a path planning method where obstacle avoidance is achieved by genetic algorithm for 3D robot manipulator. Even though genetic algorithm has a strong environmental adaptability and has been applied in path planning, it needs to be trained for a long time in order to have enough generations to find the optimized solution.

2.1.1 Rapidly-Exploring Random Tree

Rapidly-exploring Random Tree (RRT) is a sampling-based [16] algorithm that is designed for efficiently searching non-convex high-dimensional spaces efficiently. RRTs are constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree. RRTs are particularly suited for path planning problems that involve obstacles and differential constraints. It can be considered as a technique for generating open-loop trajectories for nonlinear systems with state constraints.

RRT is a fast algorithm but cannot guarantee asymptotic optimality [17,18]. To improve the efficiency of the RRT algorithm, various methods have been proposed including Potential Field Planner [19, 20]. RRT relies on the randomly-selected branches and collision-checked method to obtain a path in the mission environment. The path is planned by building a random tree which starts from the initial point and ends at the desired position. When a point is randomly sampled in the space, it then checks whether this point collides with the obstacles in the space. If there is no collision, it checks whether the straight line starting from this sampling point to the nearest point in the tree has no collisions. If there is no collision, this sampling point is then added into the tree and the nearest point is the current node. However, this sampling point is ignored and seen as invalid one if it has any collisions. The RRT algorithm continuously searches the map and the trees with points and the nodes that are formatted during the whole process. Even though RRT can achieve path planning in most common cases, the process of RRT is very time-consuming and it cannot

even find a final path in some cases where the space is too narrow for the tree to stretch out. Thus, an RRT itself may not be sufficient to solve path planning problems. However, RRT can be regarded as a component algorithm which can be incorporated into the development of many other path planning algorithms.

As a result, some variations of RRT are designed to improve efficiency. There are many RRT-variants such as RRT* [21], DT-RRT [22] and Fast RRT [23] that are widely used for optimization. RRT* is one of the recent sampling-based algorithms proposed as an extension of RRT [24, 25]. It iteratively generates and optimizes the path as the number of sampling times increases. RRT-connect [26] is a bi-directional version method which synchronously searches two trees from start point to goal point. The RRT-connect planner [27] is designed specifically for path planning that involves no differential constraints [28, 29]. In this case, the need for incremental motions is less important. The connect heuristic is a greedy function that can be considered as an alternative to the extension function. DT-RRT is a dual tree RRT algorithm with workspace tree and state tree. The workspace tree contains the position, type and connections of nodes. The state tree contains the trajectories with control inputs. All the RRT-variants [30-34] can optimize the performance of RRT by changing the rules or combining it with other methods in different cases. Thus, RRT algorithm can be considered as a base method which can be combined with other methods to perform better in some cases.

2.1.2 Probabilistic Roadmap

Another popular sampling-based path planning algorithm is Probabilistic Roadmap method (PRM). Instead of generating the graph in each desired path, PRM aims to build a single roadmap by generating a limited number of random points in a given area. Kavraki [35] proposed an analysis of probabilistic roadmaps for path planning. The method has proven to be successful in practice, but the performance is still limited theoretically. Path planning method based on PRM algorithm [36, 37] is then implemented on car-like mobile robot in unknown environments, where the algorithm performs well in the complex environment. In addition, a 3D PRM based real-time path

planning method [38, 39] is proposed for UAV in complex environment, which can significantly reduce the computational time than traditional PRM method. However, similar to RRT, it is still not efficient enough and may not find a final solution in complicated environments where narrow nodes can be ignored.

2.1.3 A* Graph Traversal Path Search

A* is a search based [40] modification algorithm of Dijkstra's Algorithm that is optimized for a single destination. Dijkstra's Algorithm can find paths to all locations. A* finds some paths to one location, or the closest of several locations. It prioritizes the paths that seem to be leading closer to a goal. Starting from the initial vertex where the path should start, the algorithm marks all direct neighbors of the initial vertex with the cost to get there. It then proceeds from the vertex with the lowest cost to all of its adjacent vertices and marks them with the cost to get to them if this cost is lower. Once all neighbors of a vertex have been checked, the algorithm proceeds to the vertex with the next lowest cost. Once the algorithm reaches the goal vertex, it terminates and the robot can follow the edges pointing towards the location with the lowest cost.

While Dijkstra's Algorithm works well to find the shortest path, it wastes the time exploring in the directions that are not promising. That is why A* comes out and is widely used in some cases. Depending on the environment, A* might accomplish search much faster than Dijkstra's algorithm does. Greedy Best First Search explores in the promising directions but it may not find the shortest path. The A* algorithm uses both the actual distance from the start and the estimated distance to the goal. Compared with other artificial intelligence algorithms, A* has many advantages such as high efficiency, easy implementation and shorter running time. Thus, it has been widely implemented in various fields.

2.2 Local Path Planning

2.2.1 Artificial Potential Field(APF)

The potential field method was first proposed by Khatib [41]. This algorithm is based on the principle of potential field force of attraction or repulsion in which robot and obstacle act as a positive charge where the goal acts as a negative charge. Thus, obstacles repel from the robot by generating repulsive force and the goal attracts the robot due to opposite charge results in attractive force. Final force on robot is the vector sum of all repulsive and attractive force.

The idea of a potential field is taken from nature. For instance, a charged particle navigating a magnetic field, or a small ball rolling in a hill. The idea is that depending on the strength of the field, or the slope of the hill, the particle, or the ball can arrive at the source of the field, the magnet, or the valley in this example. In robotics, we can simulate the same effect, by creating an artificial potential field that attracts the robot to the goal. By designing adequate potential field, we can make the robot exhibit simple behaviors. For instance, if there is no obstacle in the environment, the robot should seek this goal. To do that in conventional planning, one should calculate the relative position of the robot to the goal, and then apply the suitable forces that drive the robot to the goal. In the potential field approach, we simple create an attractive field going inside the goal. The potential field is defined across the entire free space, and in each time step, we calculate the potential filed at the robot position, and then calculate the induced force by this field. The robot then should move according to this force. However, APF has its own pros and cons:

The advantages of APF is that it is applicable for online or real-time environment as well with the added obstacle avoidance component.

The disadvantages are that potential fields method suffers from local minima [42] problem and high complexity due to its bi-operational path model. Koren and Borenstein [43] identified some problems that are inherent to potential fields, i.e. they exist in all implementations of the method:

The first problem is the trap situations due to local minima. The local minima problem may occur when all the vector field from obstacles and the goal point cancel each other so

the path never reaches the goal. Furthermore, there is no passage between closely spaced obstacles. If two obstacles are placed close to each other like a doorframe, the repulsive forces from each obstacle is combined into a single repulsive force that points away from the opening between the obstacles. The robot will then turn away from the opening even if the goal is on the other side.

2.2.2 Fuzzy Logic Method

Fuzzy logic method [44] is more robust and performs better in local minima problem compared with artificial potential field. The advantages in fuzzy logic is that it produces better result than a human can produce in a short period of time. It is well suited for implementing a solution in the complex autonomous mobile system, but it is difficult for simple control system.

Fuzzy logic is a method of reasoning that resembles human reasoning. The approach of fuzzy logic imitates the way of decision making in humans that involves all intermediate possibilities between digital values YES and NO. The conventional logic block that a computer can understand takes precise input and produces a definite output as TRUE or FALSE, which is equivalent to human's YES or NO. The inventor of fuzzy logic, Lotfi Zadeh, observed that unlike computers, the human decision making includes a range of possibilities between YES and NO, such as definitely yes, possibly yes, and cannot say etc. The fuzzy logic works on the levels of possibilities of input to achieve the definite output.

The reason why researchers choose Fuzzy Logic on path planning is that it has a perfect implementation when it comes to real time obstacle avoidance. The methods discussed above assume that the map of the environment is known in advance. Based on that knowledge, we can easily find the path that the UAV should follow on according to many path planning algorithms. However, if the environment is known, path planning becomes very tough for many methods such as RRT and PRM. Therefore, fuzzy logic is useful to help deal with the uncertainty.

Fuzzy logic [45-47] control and inferencing systems have found applications in several path planning methods (c.f., [48] and [49]). Fuzzy logic is a soft computing method and has the ability to make use of knowledge expressed in the form of linguistic rules [50, 51]. By establishing the Fuzzy logic rules it can implement expert human knowledge and experience and perform well for obstacle avoidance [52, 53]. Fuzzy logic can generally work with imprecise state of variables and uncertainties. Due to the ability to handle unknown conditions and react dynamically [54, 55], fuzzy logic is an ideal tool to address the obstacle avoidance problem. Thus, the fuzzy logic-based path planning can lead to an efficient approach for UAV to move and avoid obstacle in real time. By far, most of the fuzzy logic systems have a large amount of inputs and outputs information to be processed and thus do not work efficiently. When no obstacle is detected, the UAV will waste time on finding the direction towards the target. In this work, a combination of RRT and fuzzy logic controller is implemented for improving the efficiency. This method makes it more efficient for the UAV to reach the target since fuzzy logic controller stops searching when no obstacles are detected in sensing radius.

2.3 Summary

In this chapter, a literature survey on global path planning method and local path planning and obstacle avoidance method have been carried out. Different methods have their own pros and cons according to different implementation cases. For example, RRT, PRM and A* are very useful and efficient global path planning algorithms. However, RRT is very time-consuming when it comes to narrow corner or 3D environment because the nodes are stretching out randomly. Both sampling-based and search-based algorithm has to know the map of the environment in advance. Fuzzy logic can then overcome that constrains and help avoid obstacle more efficiently in real time without knowing the environment.

Chapter 3 Integrated System Description

This chapter presents an overview of the drone selection, software simulation environment including AirSim, ROS Gazebo with Simulink and fuzzy logic inference system in this research.

3.1 UAV Model

3.1.1 Structure

In this thesis, a 3D quadrotor Iris Quadrotor model simulation model is chosen. It has many virtual parameters, fixed-pith propellers, 850kv brush-less motors, Electronic Speed Controllers (ESC), aluminum arms, a power distribution board and GPS. The quadrotor is powered by a Hyperion 3s 4000mAh 25C battery. The curved body comes with 4 hands, and two of them are black while the other two are green, allowing better orientation. Every hand extends from a corner of the body, and a motor with a propeller. The quadrotor is shown in Figure 3.1.

Figure 3.1 IRIS quadrotor model

The quadrotor is chosen to fly in Quadrotor X- configuration. These configurations are shown in Figure 3.2, where the blue and green arrows indicate rotor configuration. Green indicate clockwise direction of the rotors, while blue indicate counter-clockwise direction.

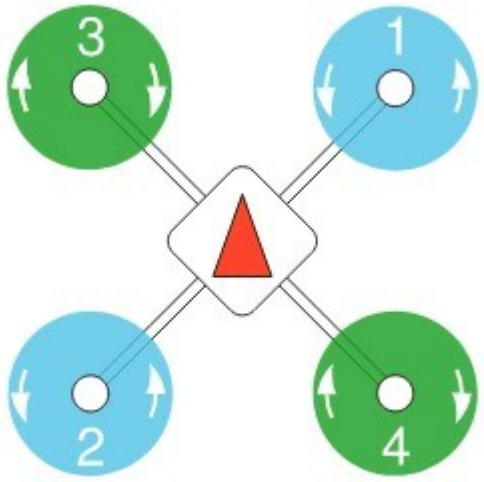

Figure 3.2 Quadrotor X-configuration

3.1.2 Movement principle

The X-shape quadrotor mathematical modeling of dynamical systems can be presented as below [56]:

$$\begin{cases} \ddot{x}=(\cos\phi\sin\theta\cos\varphi+\sin\phi\sin\varphi)\frac{1}{m}U_1-\frac{K_1}{m}\dot{x} \\ \ddot{y}=(\cos\phi\sin\theta\sin\varphi-\sin\phi\cos\varphi)\frac{1}{m}U_1-\frac{K_2}{m}\dot{y} \\ \ddot{z}=(\cos\phi\cos\theta)\frac{1}{m}U_1-g-\frac{K_3}{m}\dot{z} \\ \ddot{\phi}=\dot{\theta}\dot{\varphi}\left(\frac{I_y-I_z}{I_x}\right)-\frac{Jr}{I_x}\dot{\theta}\Omega+\frac{l}{I_x}U_2-\frac{lK_4}{m}\dot{\phi} \\ \ddot{\theta}=\dot{\phi}\dot{\varphi}\left(\frac{I_z-I_x}{I_y}\right)-\frac{Jr}{I_y}\dot{\phi}\Omega+\frac{l}{I_y}U_3-\frac{lK_5}{m}\dot{\theta} \\ \ddot{\varphi}=\dot{\phi}\dot{\theta}\left(\frac{I_x-I_y}{I_z}\right)+\frac{l}{I_z}U_4-\frac{lK_6}{m}\dot{\varphi} \end{cases} \quad (3\text{-}1)$$

where ϕ, θ, φ are the rotation angle (counterclockwise) of the fuselage around the Y-axis, X-axis and Z-axis, I_x, I_y, I_z are the moment of inertia of the fuselage in three directions, J_r is the moment of inertia, $K_1, \ldots K_6$ are the air resistance coefficient, l is the arm length from the motor to the center of mass, m is the mass of the body and g is the gravitational acceleration, and U is the propellers' speed input.

Then the X-shape control input can be defined as:

$$\begin{cases} U_1 = b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 = b(\Omega_1^2 - \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ U_3 = b(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_4 = d(-\Omega_1^2 + \Omega_2^2 + \Omega_3^2 - \Omega_4^2) \end{cases} \tag{3-2}$$

where $\boldsymbol{\Omega_1, \Omega_2, \Omega_3, \Omega_4}$ are the speeds of the four motors respectively, b, d are the force to torque scaling factors respectively.

Hovering

As each motor rotates with its propeller, it generates an upward lift force and a counter-torque force in the opposite direction. When the counter-torque force generated by the two diagonal shafts (motor 1+2 VS motor 3+4) is equal to each other, the system stability is guaranteed. At the same time, the combined lift from the four motors is just enough to cancel out the plane's own gravity, and the plane hovers.

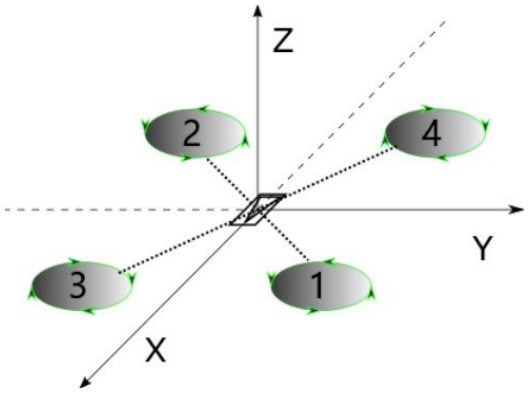

Figure 3.3 The rotation state of the four motors when hovering

Vertical motion

It is ensured that the reversing torques cancel each other and the total lifting force is

increased so that it is greater than gravity, and the body can rise vertically. If the total lifting force is decreased to the level less than gravity, the body can fall vertically.

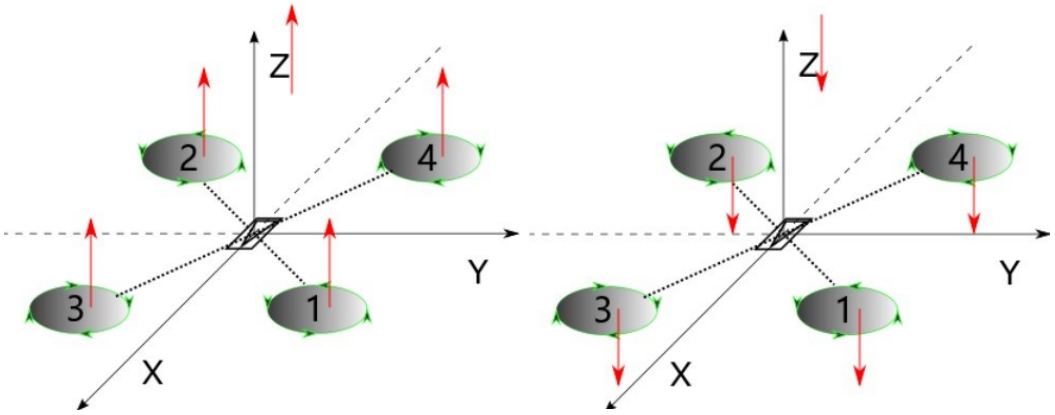

Figure 3.4 The rotation state of the four motors when doing Vertical movement

Pitch motion (forward and backward motion)

At the same time, the speeds of motors 1 and 3 are reduced and the speeds of motors 2 and 4 are increased, so the aircraft will be bent forward. The total lift in the forward position is not vertical, but forward with the plane. This will produce a component going forward in the horizontal direction. In this position, the plane moves forward with this horizontal force. Similarly, if we increase the speed of motor 2 and 4 and decrease the speeds of motors 1 and 3, the plane will lean back. The total lifting force in the case of rearward, creates a component of the horizontal rearward force. In this position, the plane will move backwards with this horizontal force.

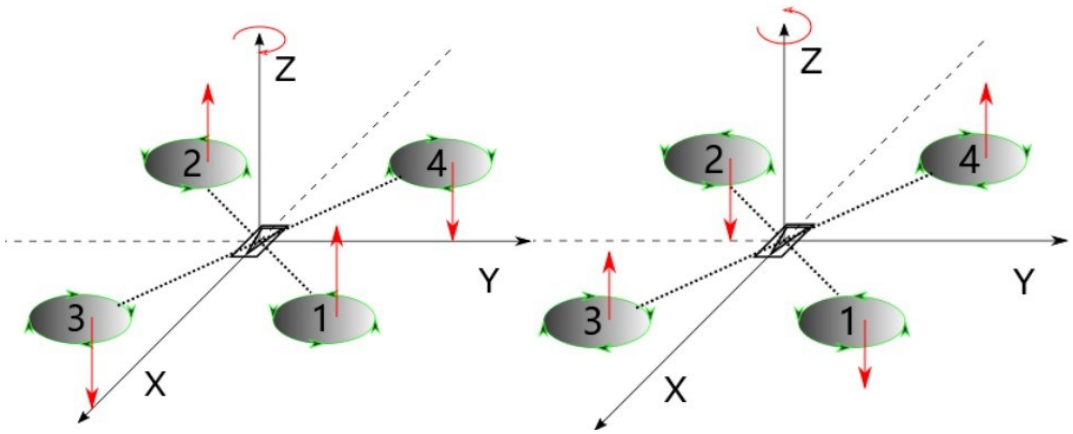

Figure 3.5 The rotation state of the four motors when doing Pitch motion

Rolling motion (side motion)

The principle is similar to pitching motion. If one increases the speeds of motors 1 and 4, and decreases the speeds of motors 2 and 3, and the plane will roll to the right. If one leans to the right, the plane will move to the right. If one increases the speeds of motors 2 and 3, and decreases the speed of motors 1 and 4, and the plane will roll to the left. If one leans to the left, the plane will move to the left.

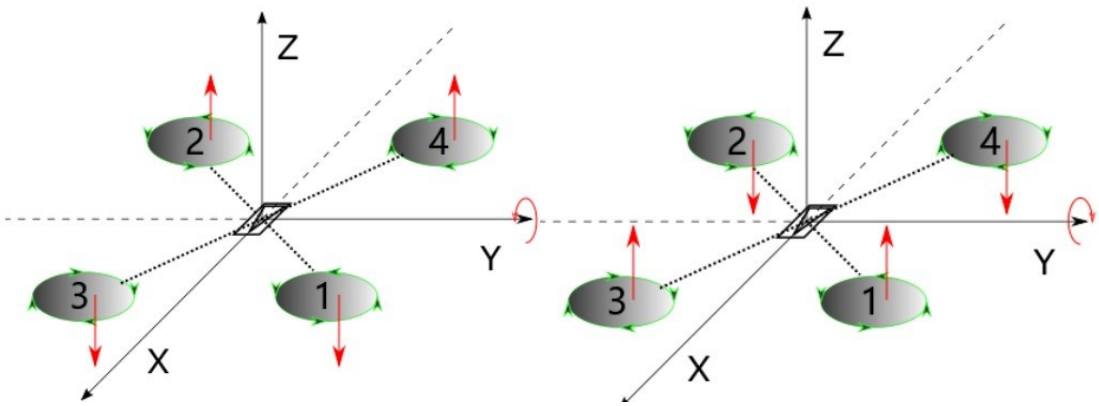

Figure 3.6 The rotation state of the four motors when doing Rolling motion

3.1.2 Flight controller

Pixhawk is a popular general flight controller launched by manufacturer 3DR. It is an independent open-hardware project providing readily-available, low-cost, and high-end, autopilot hardware designs to the academic, hobby and industrial communities, and is the reference hardware platform for PX4.

The primary job of flight controller is to take the desired state as input, estimate actual state using sensors data and then drive the actuators in such a way that actual state comes as close to the desired state. For quadrotors, the desired state can be specified as roll, pitch and yaw, for example. It then estimates the actual roll, pitch and yaw using gyroscope and accelerometer. Then it generates appropriate motor signals so that the actual state becomes the desired state.

The Pixhawk has a 32-bit ARM Cortex M4 core with FPU processor, MPU6000 and ST Micro 16-bit gyroscope sensors, 7V servo rail high power and many interfaces for ports, signals input and output to use. It also includes an Inertial Measurement Unit (IMU)

with InvenSense MPU-6000, 3-axis Gyro/3-axis Accelerometer, Honeywell HMC5883L-TR 3-axis Digital Compass and Measurement Specialties MS5611-01BA03 Barometric Pressure Sensor.

Figure 3.7 Pixhawk hardware

In addition, the Pixhawk has a very rich expansion. It has Scalable 1 set of electronic compass, 2 sets of NMEA or UBX standard GPS, CAN bus device (ESC), 2 I2C devices (smart battery, status light, optical flow smart camera, laser sensor, ultrasonic sensor, etc.) This drone control system runs PX4 and ArduPilot environment. It has many powerful features with a very high stability.

3.2 Simulation Environment

Software simulation is essential for quadrotor testing since it has very low cost and there are a lot of open source platforms. In this research, AirSim is mainly used for simulation, MATLAB is used to design fuzzy logic inference and ROS GAZEBO with Simulink is utilized for testing.

3.2.1 AirSim and Unreal Engine

AirSim is a simulator for drones, cars and more, built on Unreal Engine and Unity platform. It is open-source, cross platform, and supports software-in-the-loop simulation with popular flight controllers such as PX4 & ArduPilot and hardware-in-loop with PX4 for physically and visually realistic simulations. It is developed as an Unreal plugin that can simply be dropped into any Unreal environment.

AirSim is cross-platform simulator for drone simulation. It can be used in Windows, Linux and macOS. It also provides support for PX4 and ArduPilot.

Since setting up PX4 is not a trivial task in most cases, AirSim also provides a built-in flight controller called simple_flight. Normally the flight controllers are designed to run on actual hardware on vehicles and their support for running in simulator varies widely. They are often fairly difficult to configure for non-expert users and lacking cross platform support. All these problems have played significant part in design of simple_flight.

The built-in simple_flight flight controller can control vehicle by taking the desired input as angle rate, angle level, velocity or position. Each axis of control can be specified with one of these modes. Internally simple_flight uses the cascade of PID controllers to finally generate actuator signals.

3.2.2 Computer vision

AirSim has a “Computer Vision” mode. In this mode, we can use the keyboard to move around the scene, or use APIs to position available cameras in any arbitrary pose, and collect images such as depth, disparity, surface normal or object segmentation.

3.2.3 ROS GAZEBO with Simulink

The Robot Operating System (ROS) is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating

complex and robust robot behavior across a wide variety of robotic platforms.

In this research, there is a simulation based on ROS being implemented. GAZEBO is a 3D environment simulation software, which can load the flight model. In addition, Simulink is also used to calculate and generate the path, and publish the commands for quadrotor in ROS GAZEBO to use. The PID controller is simulated in SIMULIINK and also the trajectory generation is implemented in MATLAB.

3.3 Inference System

3.3.1 Fuzzy Logic MATLAB

Fuzzy Inference System (FIS) is the key unit of a fuzzy logic system whose primary work is making the decision. It uses the "IF…THEN" rules along with connectors "OR" or "AND" for drawing essential decision rules.

The following are some characteristics of FIS:

- The output from FIS is a fuzzy set irrespective of its input which can be fuzzy or crisp.
- It is necessary to have fuzzy output when it is used as a controller.
- A defuzzification unit would be there with FIS to convert fuzzy variables into crisp variables.

The fuzzy inference system has five functional blocks. The first block is the rule base which contains fuzzy IF-THEN rules. The second one is database, which defines the membership functions of fuzzy sets in fuzzy rules. Then decision making unit is used to perform operation on rules. Fuzzification inference unit converts the crisp quantities into fuzzy quantities. Finally, Defuzzification Interface Unit converts the fuzzy quantities into crisp quantities. The following is a block diagram of fuzzy interference system.

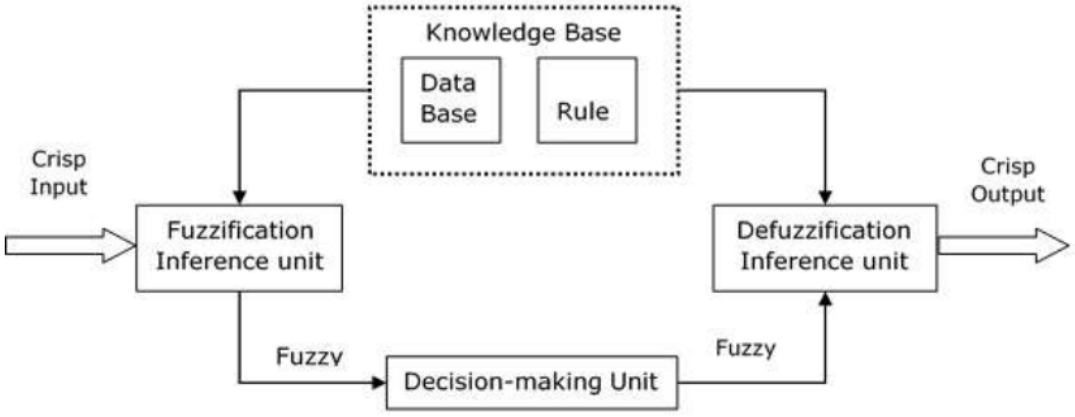

Figure 3.8 FIS block diagram

Mamdani Fuzzy Inference System [57] was proposed in 1975 by Ebhasim Mamdani. Basically, it was anticipated to control a steam engine and boiler combination by synthesizing a set of fuzzy rules obtained from people working on the system.

In this research, Mamdani Fuzzy Inference System is designed in MATLAB. This fuzzy logic inference is implemented to avoid the obstacles efficiently in unknown environments in real time.

3. 4 Summary

This chapter mainly describes the quadrotor selection with movement principles, flight controller, simulation experimental environment used in this research. The simulation environment includes the AirSim open source platform, ROS Gazebo with Simulink, and finally the Fuzzy Logic Inference System designed in MATLAB.

Chapter 4 Research Methodologies

4. 1 Fuzzy Logic Overview

In this chapter, the developed methodologies will be presented on Fuzzy Kinodynamic RRT, which is a combination of rapidly-exploring random tree algorithm and fuzzy logic inference for two-dimensional navigation, and the improved fuzzy logic controller without global path planning for drone autonomous navigation. First, the fuzzy logic inferences involved in the developed methodologies are introduced.

The core part of fuzzy logic controller is the fuzzy logic inference shown in Figure 4.1.

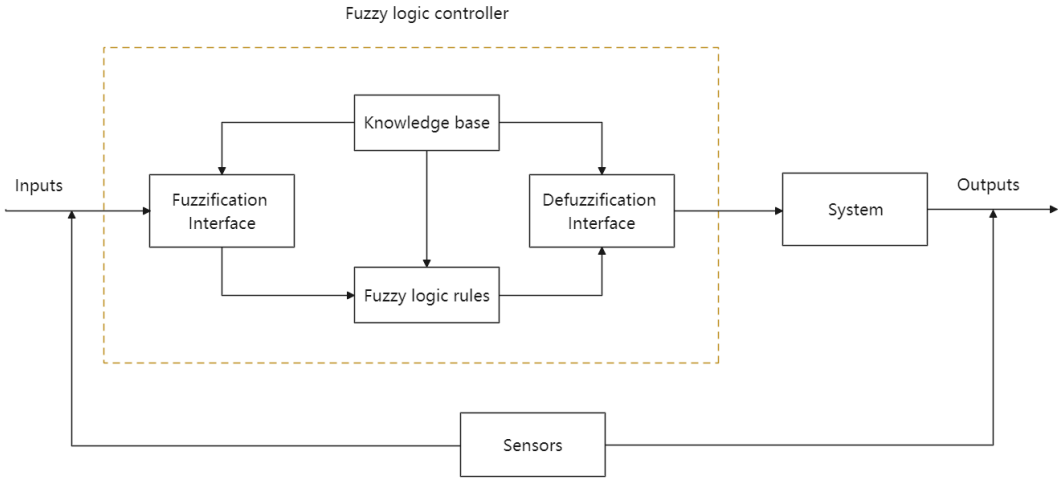

Figure 4.1 Fuzzy logic controller overview

The whole obstacle avoidance for this fuzzy logic inference is very similar to driving a car. The agent can be equipped with three sensors, which are on the forward, 45 degrees to the left and 45 degrees to the right respectively. Then we can simply set two inputs, which are the distance to the front and the distance subtraction between left distance and right distance. Based on the designed fuzzy logic rules, we can get the steering angle as the output from the system. With the help of this fuzzy logic controller, the agent can avoid obstacles effectively in real time.

4.2 Path planning and collision avoidance

A new method is developed for path planning and obstacle avoidance in unknown environments. Fuzzy Kinodynamic RRT uses rapidly-exploring random tree (RRT) algorithm to find the global trajectory, and the fuzzy logic inference is activated when the sensors detect any obstacles to help the drone to avoid collisions dynamically. With the help of RRT and fuzzy logic inference, global path can be effectively generated and any incoming obstacles can be avoided in real time. However, Fuzzy Kinodynamic RRT algorithm is subjected to some constraints because it is not fast enough and the drone needs to wait for some time for the system to generate and regenerate the global trajectory. In addition, mapping the whole environment is another issue. Thus, the fuzzy logic controller for more complicated environments is developed for autonomous drone navigation without knowing the map and global path planning in advance. We have conducted simulations of the proposed algorithm using MATLAB fuzzy logic inference, Visual Studio Code and Microsoft open source platform AirSim in Unreal Engine.

The following sub-sections explain the individual parts of the presented methods for drone path planning and obstacle avoidance.

4.2.1 Rapidly-exploring random tree

Traditional path planning algorithms include artificial potential field method, fuzzy rule method, genetic algorithm, neural network, simulated annealing algorithm, and ant colony optimization algorithm. However, all these methods need to model obstacles in a certain space. The computational complexity grows exponentially as the robot's degree of freedom increases, which is not suitable for path planning of multi-degree-of-freedom robots in complex environments. The path planning algorithm based on RRT (rapidly-exploring random tree), through the collision detection of sampling points in the state space, avoids the modeling of the space, and can effectively solve the path planning in high-dimensional space and complex constraints planning issues. The feature of this method is that it can search high-dimensional space quickly and effectively. Through random sampling points

in the state space, the search is directed to blank areas, so as to find a planned path from the starting point to the target point. It is suitable for solving the complex problems of multi-degree-of-freedom robots. Similar to PRM, this method is probabilistically complete and not optimal.

The RRT is an efficient planning method in a multi-dimensional space. It uses an initial point as the root node, and generates a random extended tree by randomly sampling and adding leaf nodes. When the leaf node in the random tree contains the target point or enters the target area, one can find a random tree and the path from the initial point to the target point. The basic RRT algorithm is shown in the following pseudo code:

```
function BuildRRT(qinit, K, Δq)
    T.init(qinit)
    for k = 1 to K
        qrand = Sample()
        qnearest = Nearest(T, qrand) -- selects the node in the RRT tree that is closest to qrand
        if  Distance(qnearest, qgoal) < Threshold then
            return true
        qnew = Extend(qnearest, qrand, Δq)  -- moving from qnearest an incremental distance in qrand direction
        if qnew != NULL then
            T.AddNode(qnew)
    return false

function Sample() -- chooses a random configuration
    p = Random(0, 1.0)
    if 0 < p < Prob then
        return qgoal
    elseif Prob < p < 1.0 then
        return RandomNode()
```

Figure 4.2 pseudo code RRT

The random tree T contains only one node during initialization: the root node ***qinit***. First, the Sample function randomly selects a sampling point ***qrand*** (4 lines) from the state space; then the Nearest function selects a node ***qnearest*** (5 lines) closest to ***qrand*** from the random tree. Finally, the Extend function extends a distance from ***qnearest*** to ***qrand***, Get a new node ***qnew*** (line 8). If ***qnew*** collides with an obstacle, the Extend function returns empty, giving up this growth, otherwise adding ***qnew*** to the random tree. Repeat the above steps

until the distance between ***qnearest*** and the target point ***qgoal*** is less than a threshold, which means that the random tree has reached the target point, and the algorithm returns ***success*** (lines 6-7). In order to make the algorithm controllable, one can set the upper limit of the running time or the upper limit of the number of searches (3 lines). If the target point cannot be reached within the limited number of times, the algorithm returns failure.

The RRT algorithm is used in our Fuzzy Kinodynamic RRT method to make global path planning for the drone to follow. The drone will strictly follow the generated path until it encounters any obstacles and then utilize the proposed fuzzy logic controller to avoid them.

4.2.2 Fuzzy logic inference design

A simple Fuzzy logic inference system is implemented in this work for UAV navigation and obstacle avoidance, as shown in Fig. 4.3.

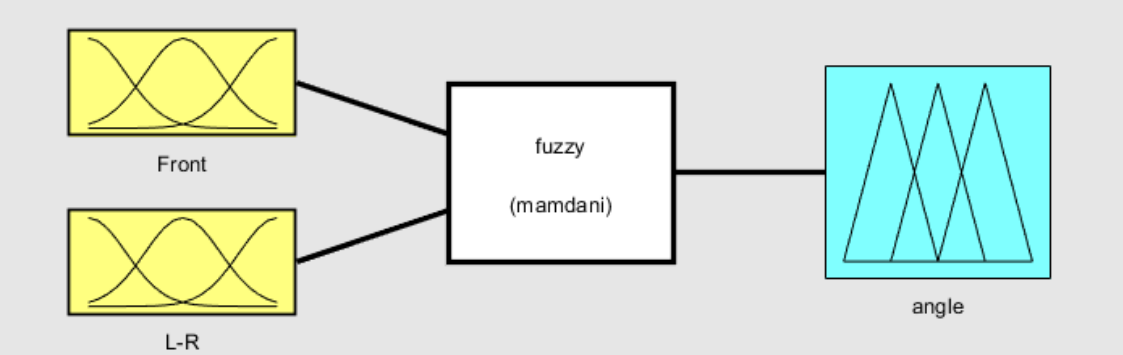

Figure 4.3. Fuzzy inference inputs and output

The system is designed with 2 inputs and 1 output, as defined below:

TABLE 4.1. Input and output

Input 1	Input 2	Output
Front distance to obstacle *I*	(Left-Right) distance to obstacle *II*	Steering Angle *III*

I: Front distance to obstacle means the distance from the front of UAV to the obstacle.

II: (Left-Right) distance to obstacle is the result of from left 45 degrees distance of the UAV to the obstacle minus right 45 degrees distance of the UAV to the obstacle.

III: Steering Angle is the steering reaction of UAV based on Fuzzy logic rules.

There are 9 Fuzzy logic rules in total. The fuzzy logic system is activated when the sensor detects any obstacles on the way. De-fuzzification is done by using center of gravity (COG) method, which produce a quantifiable result in Crisp logic, given Fuzzy sets and corresponding membership degrees. Finally, the output steering angle variable and control surface are calculated based on input variables and logic rules.

In this fuzzy inference system, Gaussian membership function is implemented which is given by equation (4.1).

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}} \tag{4.1}$$

where σ represents standard deviation and c is the mean value for Gaussian function. The membership functions for two inputs variables and one output variable with a range of {‘small’, ‘medium’, ‘large’} are shown in Figs. 4.4, 4.5 and 4.6 respectively.

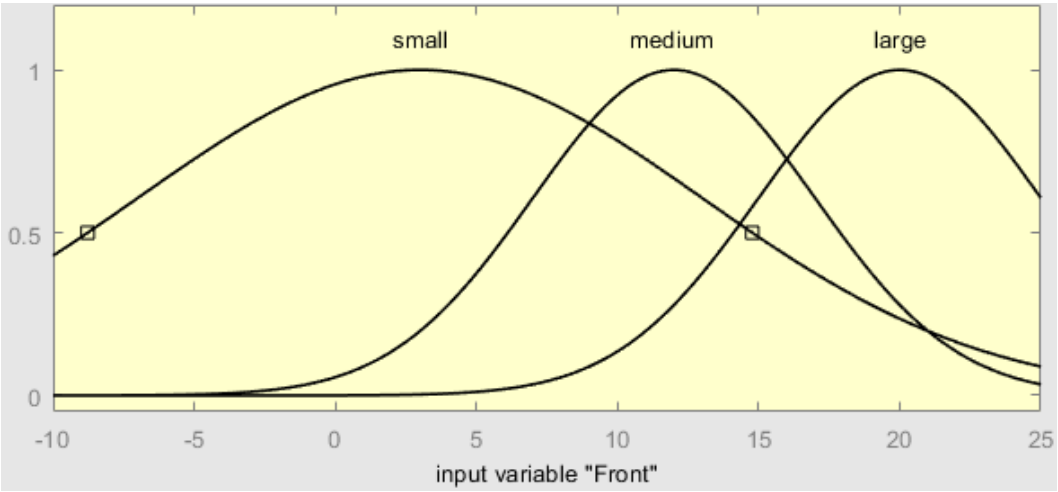

Figure 4.4. Fuzzy membership function input 1

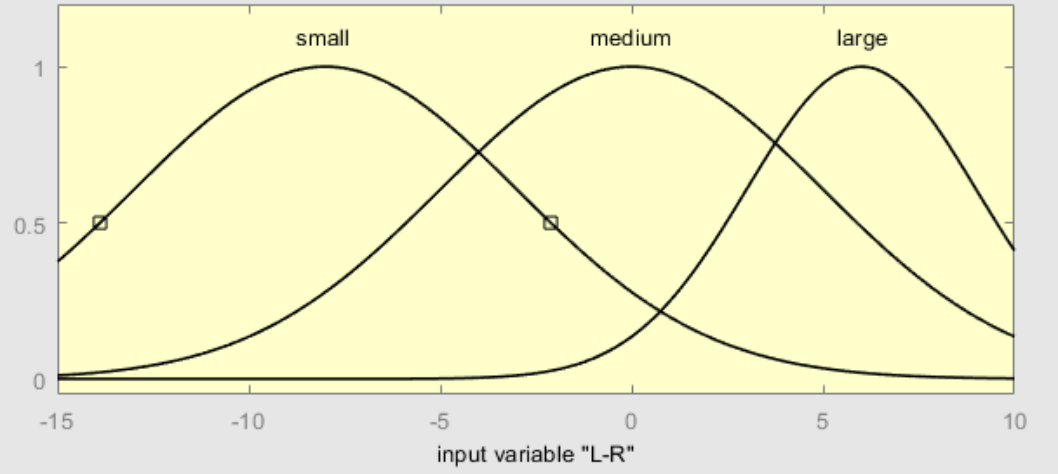

Figure 4.5. Fuzzy membership function input 2

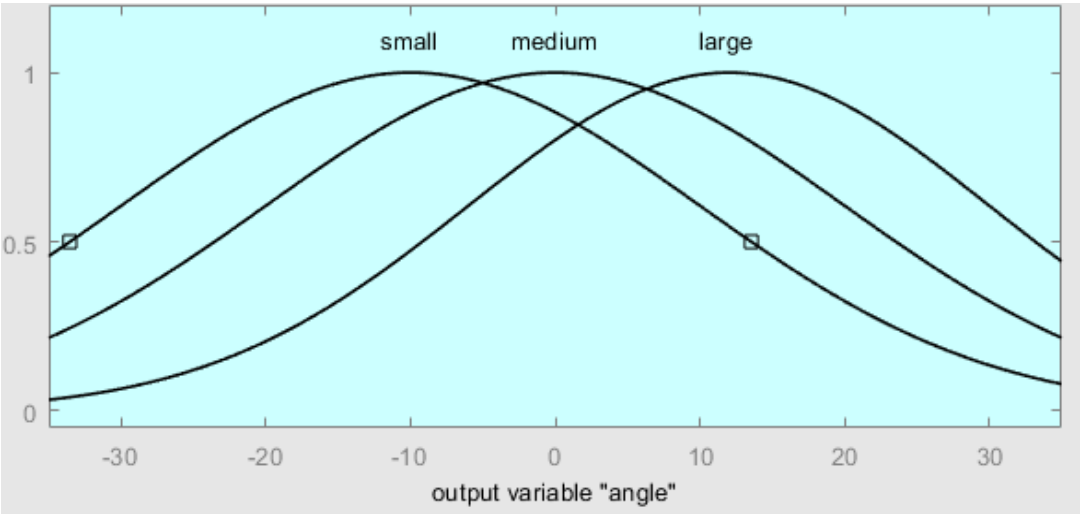

Figure 4.6. Fuzzy membership function output

The small, medium and large are based on normal Gaussian function and the parameters (mean and standard deviation) for each fuzzy variable are designed. The small, medium and large values for input 1 represent the distance to front obstacle. Thus, the value is set ranging from -10 to 25. It should be noted that left minus right distance and the output angle are possible to be both positive and negative, so the small of these two variables is defined as negative, medium as approaching zero and large as positive. The exact values are set according to experience at the beginning, and further adjustments are made according to the performance in simulation.

Each input and output membership functions are divided into three values which can be chosen from a range of {'small', 'medium', 'large'}. The physical meaning for three sets of input and output variables is shown in TABLE 4.2.

TABLE 4.2. Physical meaning of rules

	Input 1	Input 2	Output
Small	Very close to obstacle	Left distance less than right distance	Turn left
Medium	Near obstacle	Left distance almost equal to right distance	Straight ahead
Large	Far away from obstacle	Left distance larger than right distance	Turn right

With all the real meanings discussed above, "IF-AND, THEN" fuzzy logic is used to determine the rules of output based on two different inputs. Based on the real physical logics discussed above, one needs to consider all the cases where the input and output change from "small", "medium" to "large". All nine cases are discussed as shown below and the verification is implemented on simulation part.

Rule 1: If the front distance is small and left minus right is small, that means the drone

is near to the obstacles and the distance to left is less than right, then it should turn right to avoid obstacles. Thus, the output is large.

Rule 2: If the front distance is small and left minus right is medium, that means it is near obstacle and going straight ahead, then it can go either left or right to avoid obstacles. Hence the output is set as large.

Rule 3: If the front distance is small and left minus right is large, that means it is near obstacle and left is larger than right, then it should go left to avoid obstacles. Hence the output is set to small.

Rule 4: If the front distance is medium and (left - right) distance is small, that means the front distance is average and left distance is less than right distance, then it should turn right to avoid obstacles. Thus, we set the output to large.

Rule 5: If the front distance is medium and (left - right) distance is medium, that means the front distance is average and left distance is similar to right distance, then it does not need to turn left or right. Thus, we set the output to medium.

Rule 6: If the front distance is medium and (left - right) distance is large, that means the front distance is average and left distance is larger than right distance, then it should turn left to avoid obstacles. Thus, we set the output to small.

Rule 7: If the front distance is large and (left - right) distance is small, that means it has sufficient space to go straight ahead and left distance is less than right distance, then it is better to turn right in advance. Thus, we set the output to large.

Rule 8: If the front distance is large and (left - right) distance is medium, that means it has sufficient space to go straight ahead and left distance is similar to right distance, then it does not need to do anything. Thus, we set the output to medium.

Rule 9: If the front distance is large and (left - right) distance is large, that means it has sufficient space to go straight ahead and left distance is larger than right distance, then it is better to turn left in advance. Thus, we set the output to small.

As discussed above, each of two inputs and the output have three variables as {‘small’, ‘medium’, ‘large’}. So there are nine logics in total needed to be considered in real world. The Fuzzy rules are generated heuristically and are designed to cover most cases where the UAV will come across and get the proper output to avoid obstacles in unknown

environments. The Fuzzy logic rules and results are summarized in TABLE 4.3 below.

TABLE 4.3. Summary of Rules and results

	If (Input 1)	And (Input 2)	Then (Output)
Rule 1	**Small**	**Small**	**Large**
Rule 2	**Small**	**Medium**	**Large**
Rule 3	**Small**	**Large**	**Small**
Rule 4	**Medium**	**Small**	**Large**
Rule 5	**Medium**	**Medium**	**Medium**
Rule 6	**Medium**	**Large**	**Small**
Rule 7	**Large**	**Small**	**Large**
Rule 8	**Large**	**Medium**	**Medium**
Rule 9	**Large**	**Large**	**Small**

Based on the rules we can calculate the output variables and the steering angle control surface is shown in Fig. 4.7.

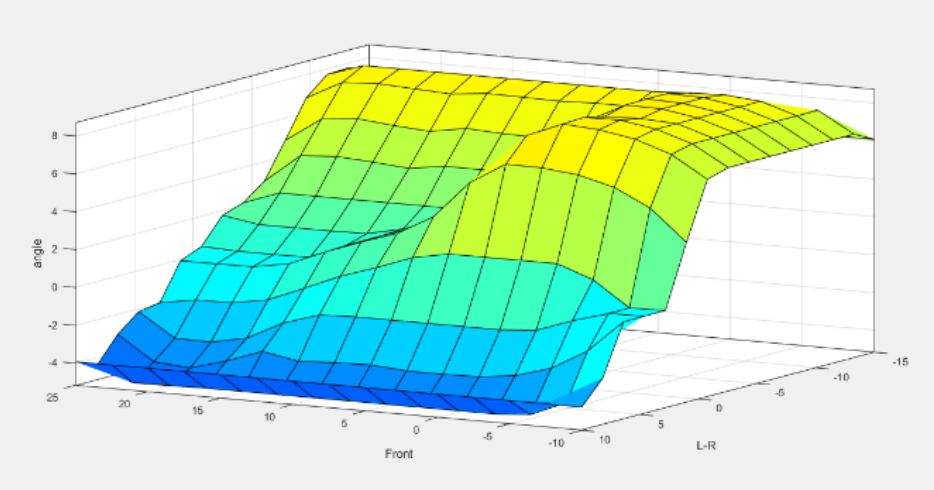

Figure 4.7. Control surface

4.2.3 Fuzzy kinodynamic RRT

The new Fuzzy-Kinodynamic RRT method presented in this work is based on RRT for global path planning and fuzzy rules for obstacle avoidance. RRT is a sampling-based method that explores the entire environment to find a connected path from the start to goal positions. However, in a scene with many obstacles the chance of finding a point that is not lying in an obstacle and able to connect to the tree is very small. The final path generated by RRT algorithm is not always the optimum path with the shortest distance from the starting to the goal point because the random trees may go far away to avoid obstacles.

The proposed Fuzzy-Kinodynamic RRT method implements both RRT algorithm and fuzzy logic and utilizes RRT algorithm to perform global path planning at the first level, and then uses fuzzy logic to achieve obstacle avoidance, and finally generates an optimized path and a set of path points that the UAV can follow by with good performance.

First, the global path planning is carried out by RRT and then fuzzy logic inferencing is activated when the obstacles are detected during the path. L is the distance from the left or right to obstacle and this parameter can be set and adjusted in program. After avoiding obstacles, the UAV will move back to and follow the previous path done by RRT, and a set of path points will be generated in the final step. The entire algorithm is illustrated in the diagram of Fig 4.8.

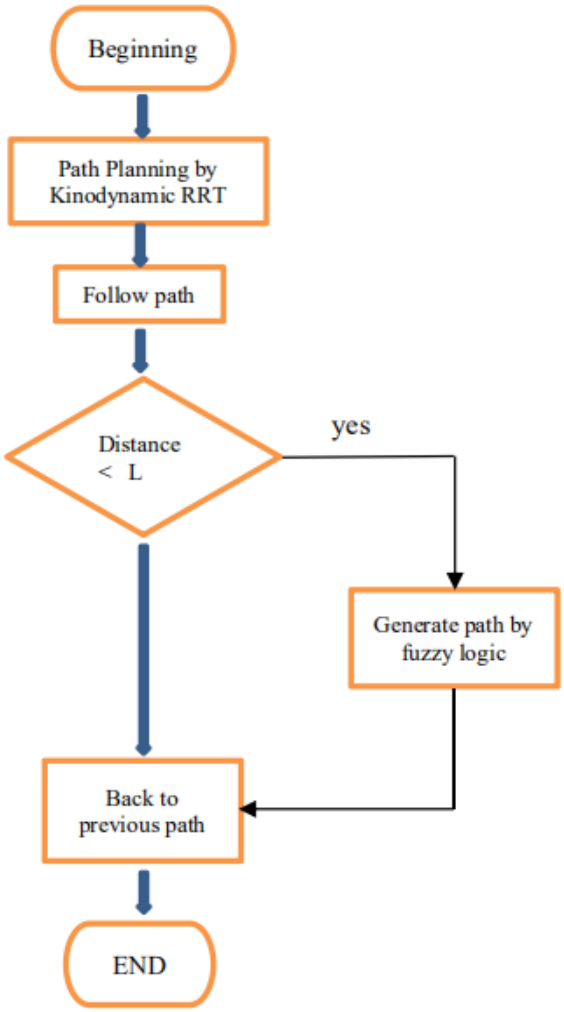

Figure 4.8. Fuzzy-Kinodynamic RRT method flowchart

The pseudocode of the algorithm is presented below:

Table 4.4 Run RRT to make global path planning

1: function BuildRRT()
2: Initialize search tree T with x start
3: while T is less than the maximum tree size do
4: xsamp sample from X
5: xnearest nearest node in T to xsamp
6: employ a local planner to find a motion from xnearest to xnew in xsamp direction

```
7:      if the motion if collision-free then
8:              add xnew to T with edge from xnearest to xnew
9:              if xnew is in Xgoal then
10:             return SUCCESS and the motion to xnew
11:             end if
12:     end if
13: end while
14: return FAILURE
```

Table 4.5 Run Fuzzy Logic Controller to avoid obstacles

```
1: function Fuzzy
2: while obstacles detected by sensor with distance L do
3:      Initialize Dfront and D(l-r)
4:      Import fuzzy rules
5:      Analyze control output based on fuzzy rules
6:      employ heading angle to agent
7: end while
8: if xcurrent in path
9: return None
10: else
11: set xcurrent to xstart
12: do function BuildRRT()
```

4.2.4 Improved Fuzzy Logic Controller

In order to make our fuzzy logic method more robust and efficient in more complicated environments without global path planning, we need to design more complex rules. In our research, the AirSim built-in distance sensor is used for distance detection. The problem formulation is shown as follows:

The path planning problem is formulated by generating a path from the initial state to final state, which are $O(x_o, y_o)$ and $T(x_{target}, y_{target})$ respectively. Thus, the kinematic equations for a UAV can be formed as a function of the inertial position (x, y), the cruise velocity (v), and the heading angle (Θ). The UAV in inertial reference frame is shown in Figure 4.9.

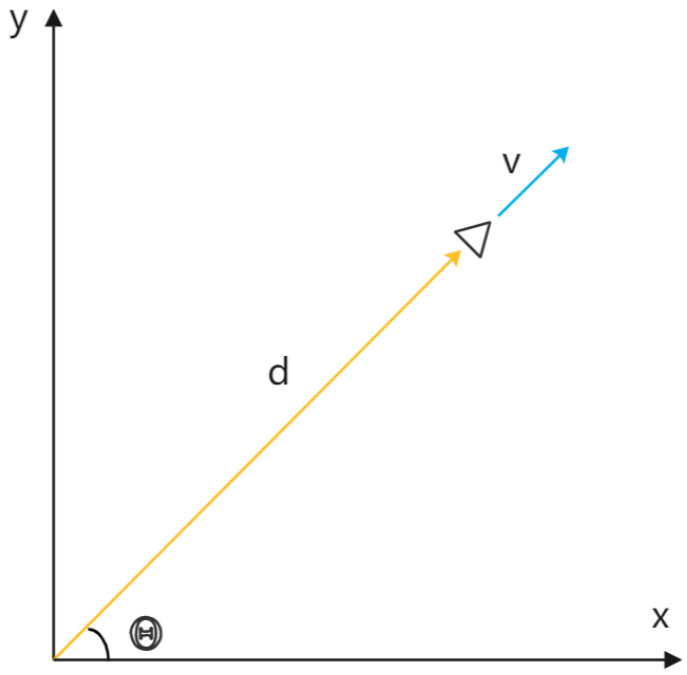

Figure 4.9: UAV in inertial reference frame

The second-order differential equations which describes the aircraft autopilot system were developed by Buzogany et al [58]. The assumption is that the altitude is constant, so we only consider all motions in two dimensions and the altitude is removed as a degree of freedom. Therefore, we are only interested in controlling the velocity and heading angle of the UAV, which is v_c and Θ_c respectively. The motion of the UAV can be described as follows in equation (4.3), where τ_v is the time delays when controlling the velocity, while τ_θ is the time delays of controlling heading angle

$$\dot{v} = \frac{1}{\tau_v}(v_c - v) \quad , \quad \dot{\theta} = \frac{1}{\tau_\theta}(\theta_c - \theta) \tag{4.2}$$

Further development by Dong et al. [59] illustrates the UAV in the inertial reference frame (Figure 4.9).The position of the UAV can be defined using the heading angle and distance from the origin which is represented as d as follows:

$$X = \begin{pmatrix} \mathrm{x} \\ y \end{pmatrix} = \begin{pmatrix} d\cos\theta \\ d\sin\theta \end{pmatrix} \tag{4.3}$$

It follows that

$$\dot{X} = \begin{pmatrix} \dot{\mathrm{x}} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \end{pmatrix} \tag{4.4}$$

And

$$\ddot{X} = \begin{pmatrix} \ddot{\mathrm{x}} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{v}\cos\theta - v\dot{\theta}\sin\theta \\ \dot{v}\sin\theta + v\dot{\theta}\cos\theta \end{pmatrix} \tag{4.5}$$

Combing equations (4.2) and (4.5), the kinematic equations can be written in the following form:

$$\ddot{X} = \begin{pmatrix} \ddot{\mathrm{x}} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \frac{1}{\tau_v}(v_c - v)\cos\theta - v\frac{1}{\tau_\theta}(\theta_c - \theta)\sin\theta \\ \frac{1}{\tau_v}(v_c - v)\sin\theta + v\frac{1}{\tau_\theta}(\theta_c - \theta)\cos\theta \end{pmatrix} \tag{4.6}$$

where the control inputs are the velocity (v_c) and heading angle (Θ_c). Both the velocity and heading angle are constrained as follows in (4.7) and (4.8) respectively. In addition, the acceleration and heading angle rate are bounded to prevent instantaneous changes as shown in equations (4.9) and (4.10) respectively,

$$V_{\min} <= V <= V_{\max} \quad (4.7)$$

$$-\theta_{\max} <= \theta <= \theta_{\max} \quad (4.8)$$

$$-a_{\max} <= \dot{v} <= a_{\max} \quad (4.9)$$

$$-\omega_{\max} <= \dot{\theta} <= \omega_{\max} \quad (4.10)$$

In this problem setup, the drone is equipped with the AirSim built-in distance sensor, which has a sensing range within ±90◦ with a certain sensing radius as shown in Figure 4.10.

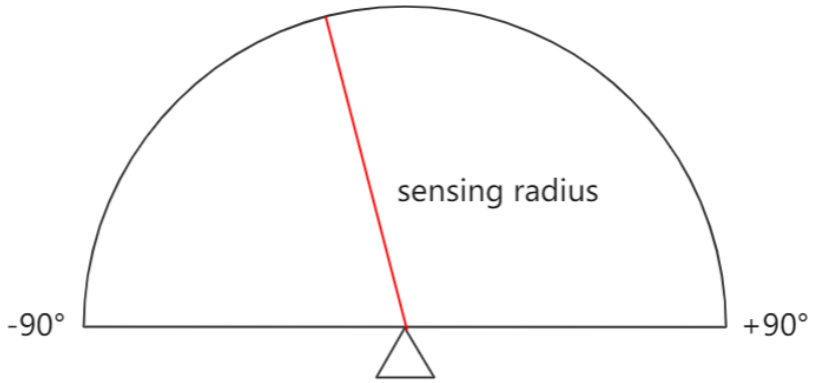

Figure 4.10. Sensing radius

Assumptions

For the purpose of this investigation, several simplifying assumptions are made without taking away from the applicability of the developed approach. While the control methodologies developed in this research can be extended three dimensions, it is assumed that all motions are in two dimensions. The altitude is a constant. Additionally, the aircraft is regarded as a point mass, and therefore, no moment effects are considered. Furthermore, the dynamic constraints of the aircraft are assumed to be known.

To isolate the performance of the control methodology, several assumptions are made on the capabilities of the sensor system. The UAV is assumed to be able to detect the obstacles within its sensing range in real time.

The position of the UAV and the target location are assumed to be known by a GPS (global positioning system) with an accurate coordinate. In addition, this information is updated at a reasonable sampling time. Furthermore, the start location of the UAV and the target positions are given with the GPS locations, and there are no obstacles at these

locations. Finally, in this research, it is assumed that a feasible solution and clear path exist and the UAV can navigate around the obstacles safely to the target location. This includes the assumption that the target location is a "safe" distance away from any obstacle.

Inputs and outputs

For this problem setup, four inputs are used for the fuzzification and two outputs are given after the defuzzification. The inputs into the system are as follows: the distance from the UAV to the obstacle, angle between the UAV and the obstacle, the distance to the target, and the error between the current heading angle of the UAV and the angle of the target in relation to the inertial reference frame. The target inputs are chosen based on the assumption that there is only minimal and limited amount of information about the target: GPS coordinates. Based on this limited information, no other mapping information needs to be known in advance. The outputs for the system is used as the control inputs to a UAV system.

The distance to the obstacle (Figure 4.11) is described by four membership functions: Close, Medium Distance, Far, and Very Far which is out of sensing radius. The angle between the obstacle and the UAV (Figure 4.12) is described by six membership functions: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Positive Small (PS), Positive Medium (PM), and Positive Big (PB). Similar to the obstacle distance, the distance to the target (Figure 4.13) is described by three membership functions: Close, Medium Distance, and Far. Finally, the error between the heading angle and the target angle (Figure 4.14) is described by seven membership functions (similar to the obstacle to the UAV): Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero, Positive Small (PS), Positive Medium (PM), and Positive Big (PB).

After defining the above inputs, the control input (FIS output also) for the system is obtained after decision making rules. Therefore, the outputs of the fuzzy inference system are the percent of the maximum velocity and the heading angle change, which are used as the control input to the system.

Thus, the outputs of the FIS are the percent of maximum velocity and the heading angle

change. The Mamdani method is used in the FIS. The output velocity (Figure 4.15) is represented by four membership functions: Very Slow, Slow, Fast, and Very Fast. Furthermore, the output angle change (Figure 4.16) is parallel to the target angle. Therefore, the output of heading angle change is broken into seven membership function: Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero, Positive Small (PS), Positive Medium (PM), and Positive Big (PB).

TABLE 4.6. Input and output

Input 1	Input 2	Input 3	Input 4	Output 1	Output 2
D_o	D_t	Θ_t	Θ_o	V	Θ_c

where:

D_o : Distance from the UAV to the obstacle

D_t : Distance from the UAV to the target

Θ_o : Angle between the UAV and the obstacle

Θ_t : The error between the current heading angle of the UAV and the angle of the target in relation to the inertial

V : The percent of the maximum velocity

Θ_c : The heading angle change

Fuzzy logic rules

The fuzzy logic controller rules are using the "IF-THEN" statements and the design is based on human experience and heuristics for obstacle avoidance and path planning. There are totally 40 rules in this fuzzy logic system. Technically the rules are based on two main cases: obstacle within sensing radius, obstacle without sensing range.

Firstly, we consider the case where there is no obstacle within the sensing radius (D_o = Very Far). In this circumstance, the main objective of the fuzzy logic controller is to drive the UAV to fly to the destination directly. Thus, path planning is done by changing

the heading angle of the UAV to match the angle of the target in the inertial reference frame, and the heading angle error between these two angles becomes zero. Compared with the Fuzzy Kinodynamic RRT method illustrated above, this fuzzy controller does not require the mapping information of the whole environment. Therefore, autonomous navigation for the UAV in more complicated environments can be done by this fuzzy logic inference without global path planning.

Since there are no obstacles in sensing range in this case, the UAV tends towards its maximum operating speed. When the UAV reaches the target location, it changes the velocity to slow down and driving the angle error to zero, i.e. the mission is accomplished. The rules between 1 and 10 are illustrated in details as shown below:

Rule 1: If D_o = Very Far and D_t = Far, then V = Very Fast.

Rule 2: If D_o = Very Far and D_t = Medium Distance, then V = Slow.

Rule 3: If D_o = Very Far and D_t = Close, then V = Very Slow.

	If (D_o)	And (D_t)	Then (V)
Rule 1	Very Far	Far	Very Fast
Rule 2	Very Far	Medium Distance	Slow
Rule 3	Very Far	Close	Very Slow

When there is error between the heading angle of the UAV and target, the fuzzy logic controller tends to change the heading angle Θ_c to Θ_t for driving the UAV towards the target.

Rule 4: If D_o = Very Far and Θ_t = Negative Small, then Θ_c = Negative Small.

Rule 5: If D_o = Very Far and Θ_t = Negative Medium, then Θ_c = Negative Medium.

Rule 6: If D_o = Very Far and Θ_t = Negative Big, then Θ_c = Negative Big.

Rule 7: If D_o = Very Far and Θ_t = Zero, then Θ_c = Zero.

Rule 8: If D_o = Very Far and Θ_t = Positive Small, then Θ_c = Positive Small.

Rule 9: If D_o = Very Far and Θ_t = Positive Medium, then Θ_c = Positive Medium.

Rule 10: If D_o = Very Far and Θ_t = Positive Big, then Θ_c = Positive Big.

	If (D_o)	And (Θ_t)	Then (Θ_c)
Rule 4	Very Far	NS	NS
Rule 5	Very Far	NM	NM
Rule 6	Very Far	NB	NB
Rule 7	Very Far	Zero	Zero
Rule 8	Very Far	PS	PS
Rule 9	Very Far	PM	PM
Rule 10	Very Far	PB	PB

Secondly, when the obstacles are detected within the sensing radius (D_o ={Far, Medium, Close}), the UAV changes the velocity and heading angle to avoid the obstacle and then comes back to normal fly. Meanwhile, the UAV has to slow down to ensure sufficient response time to avoid crash. After it avoids the obstacle, it continues the path toward the target. The regarding fuzzy rules are described in details shown below:

Rule 11: If D_o = Far and Θ_o = Negative Small, then V = Fast and Θ_c = Positive Small.

Rule 12: If D_o = Far and Θ_o = Negative Medium, then V = Very Fast and Θ_c = Zero.

Rule 13: If D_o = Far and Θ_o = Positive Medium, then V = Very Fast and Θ_c = Zero.

Rule 14: If D_o = Far and Θ_o = Positive Small, then V = Fast and Θ_c = Negative Small.

Rule 15: If D_o = Medium and Θ_o = Negative Big, then V = Fast and Θ_c = Zero.

Rule 16: If D_o = Medium and Θ_o = Negative Medium, then V = Slow and Θ_c = Positive Small.

Rule 17: If D_o = Medium and Θ_o = Negative Small, then V = Slow and Θ_c = Positive Medium.

Rule 18: If D_o = Medium and Θ_o = Positive Small, then V = Slow and Θ_c = Negative Medium.

Rule 19: If D_o = Medium and Θ_o = Positive Medium, then V = Slow and Θ_c = Negative Small.

Rule 20: If D_o = Medium and Θ_o = Positive Big, then V = Fast and Θ_c = Zero.

Rule 21: If D_o = Close and Θ_o = Negative Big, then V = Slow and Θ_c = Positive Small.

Rule 22: If D_o = Close and Θ_o = Negative Medium, then V = Very Slow and Θ_c = Positive Medium.

Rule 23: If D_o = Close and Θ_o = Negative Small, then V = Very Slow and Θ_c = Positive Big.

Rule 24: If D_o = Close and Θ_o = Positive Small, then V = Very Slow and Θ_c = Negative Big.

Rule 25: If D_o = Close and Θ_o = Positive Medium, then V = Very Slow and Θ_c = Negative Medium.

Rule 26: If D_o = Close and Θ_o = Positive Big, then V = Slow and Θ_c = Negative Small.

	If (D_o)	And (Θ_o)	Then (V)	Then (Θ_c)
Rule 11	Far	NS	Fast	PS
Rule 12	Far	NM	Very Fast	Zero
Rule 13	Far	PM	Very Fast	Zero
Rule 14	Far	PS	Fast	NS
Rule 15	Medium	NB	Fast	Zero
Rule 16	Medium	NM	Slow	PS
Rule 17	Medium	NS	Slow	PM
Rule 18	Medium	PS	Slow	NM
Rule 19	Medium	PM	Slow	NS
Rule 20	Medium	PB	Fast	Zero
Rule 21	Close	NB	Slow	PS

Rule 22	Close	NM	Very Slow	PM
Rule 23	Close	NS	Very Slow	PB
Rule 24	Close	PS	Very Slow	NB
Rule 25	Close	PM	Very Slow	NM
Rule 26	Close	PB	Slow	NS

Finally, the last set of fuzzy rules is for the cases where the obstacles are far away and the heading angle is very big (D_o = Far and Θ_o = Big). In this case, there is not any threat of collision. Thus, the main objective for the drone is to move towards the target with the same heading angle and fast speed. The fuzzy rules in details are shown below:

Rule 27: If D_o = Far and Θ_o = Negative Big and Θ_t = Negative Big, then V = Fast and Θ_c = Negative Big.

Rule 28: If D_o = Far and Θ_o = Negative Big and Θ_t = Negative Med, then V = Fast and Θ_c = Negative Med.

Rule 29: If D_o = Far and Θ_o = Negative Big and Θ_t = Negative Small, then V = Fast and Θ_c = Negative Small.

Rule 30: If D_o = Far and Θ_o = Negative Big and Θ_t = Zero, then V = Fast and Θ_c = Zero.

Rule 31: If D_o = Far and Θ_o = Negative Big and Θ_t = Positive Small, then V = Fast and Θ_c = Positive Small.

Rule 32: If D_o = Far and Θ_o = Negative Big and Θ_t = Positive Med, then V = Fast and Θ_c = Positive Med.

Rule 33: If D_o = Far and Θ_o = Negative Big and Θ_t = Positive Big, then V = Fast and Θ_c = Positive Big.

Rule 34: If D_o = Far and Θ_o = Positive Big and Θ_t = Negative Big, then V = Fast and Θ_c = Negative Big.

Rule 35: If D_o = Far and Θ_o = Positive Big and Θ_t = Negative Med, then V = Fast and Θ_c = Negative Med.

Rule 36: If D_o = Far and Θ_o = Positive Big and Θ_t = Negative Small, then V = Fast and Θ_c = Negative Small.

Rule 37: If D_o = Far and Θ_o = Positive Big and Θ_t = Zero, then V = Fast and Θ_c = Zero.

Rule 38: If D_o = Far and Θ_o = Positive Big and Θ_t = Positive Small, then V = Fast and Θ_c = Positive Small.

Rule 39: If D_o = Far and Θ_o = Positive Big and Θ_t = Positive Med, then V = Fast and Θ_c = Positive Med.

Rule 40: If D_o = Far and Θ_o = Positive Big and Θ_t = Positive Big, then V = Fast and Θ_c = Positive Big.

	If (D_o)	And (Θ_o)	And (Θ_t)	Then (V)	Then (Θ_c)
Rule 27	Far	NB	NB	Fast	NB
Rule 28	Far	NB	NM	Fast	NM
Rule 29	Far	NB	NS	Fast	NS
Rule 30	Far	NB	Zero	Fast	Zero
Rule 31	Far	NB	PS	Fast	PS
Rule 32	Far	NB	PM	Fast	PM
Rule 33	Far	NB	PB	Fast	PB
Rule 34	Far	PB	NB	Fast	NB
Rule 35	Far	PB	NM	Fast	NM
Rule 36	Far	PB	NS	Fast	NS
Rule 37	Far	PB	Zero	Fast	Zero
Rule 38	Far	PB	PS	Fast	PS
Rule 39	Far	PB	PM	Fast	PM
Rule 40	Far	PB	PB	Fast	PB

The membership function designed in MATLAB of four inputs and two outputs are shown in the following figures:

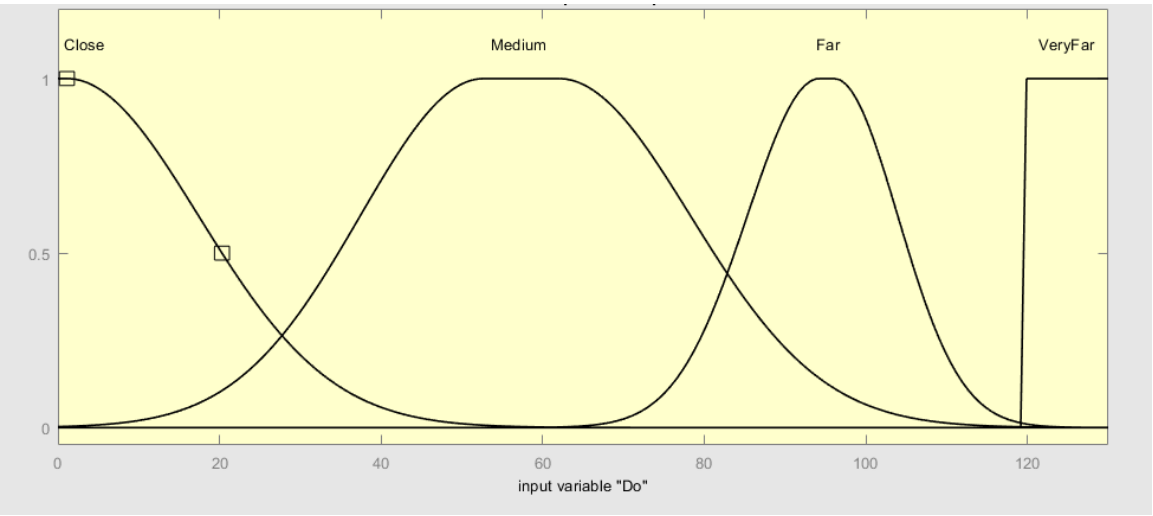

Figure 4.11: Input one: distance from obstacle to UAV

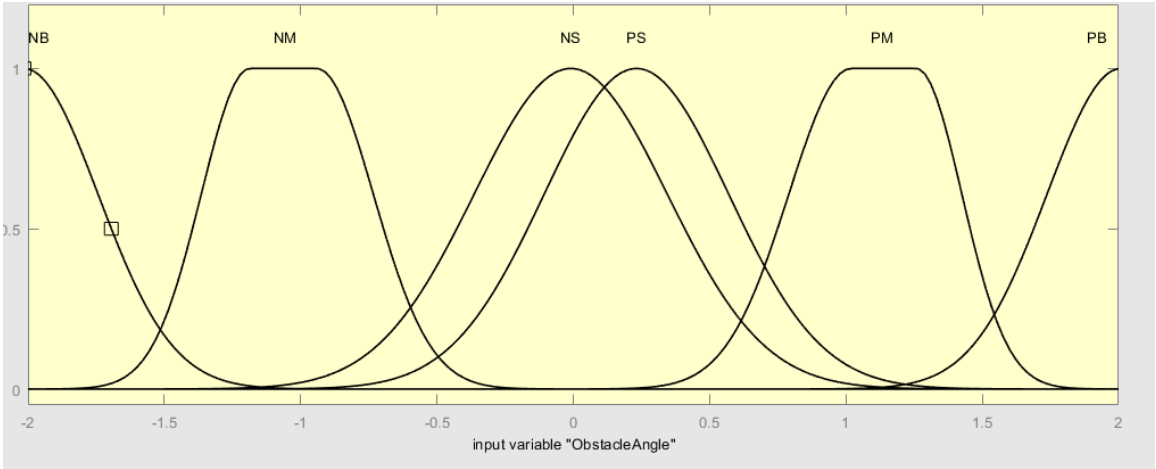

Figure 4.12: Input two: heading angle between obstacle and UAV

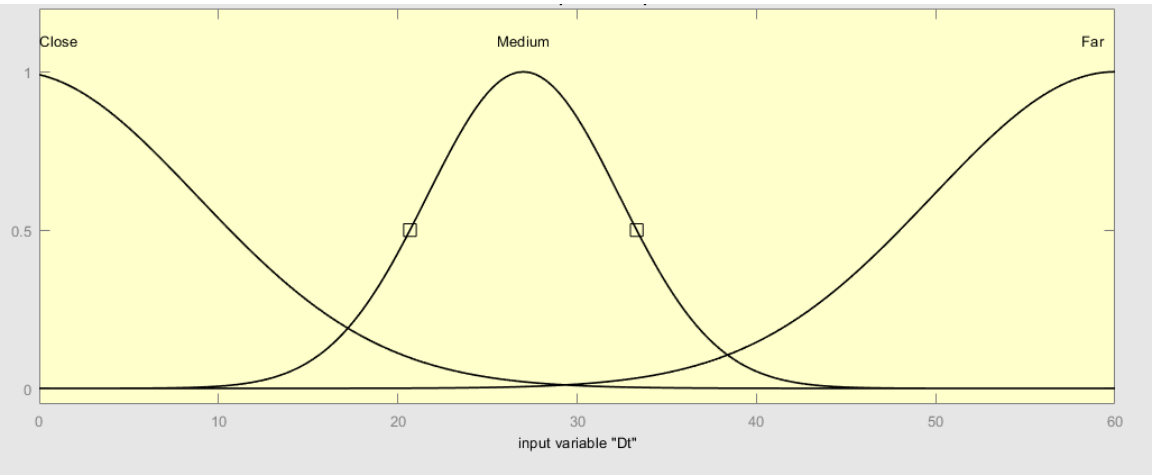

Figure 4.13: Input three: distance from UAV to target

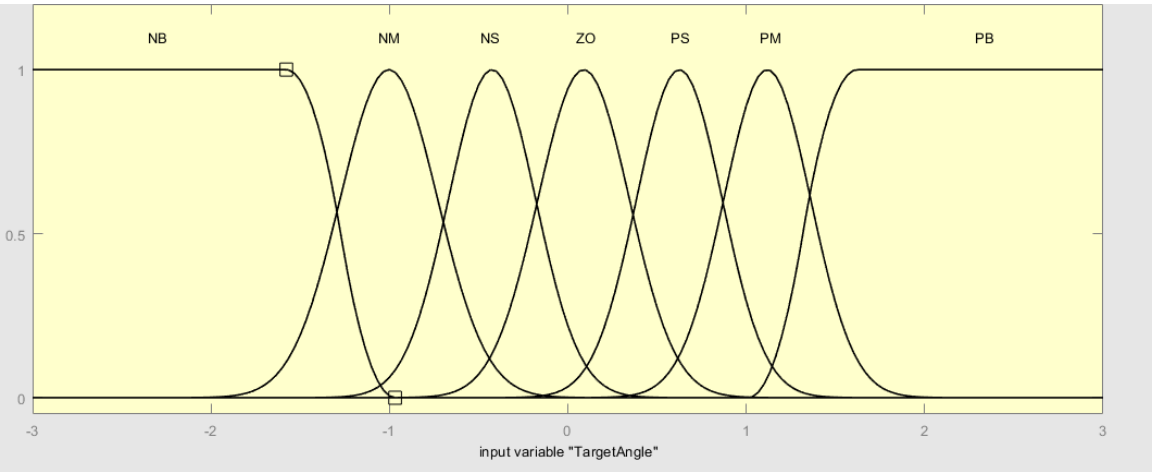

Figure 4.14: heading angle between UAV and target

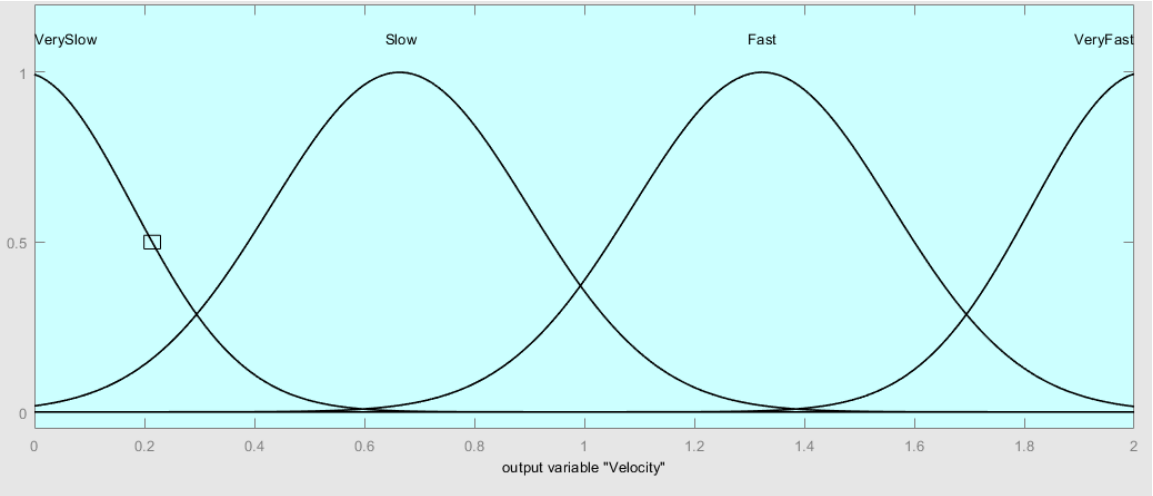

Figure 4.15: Output one: percentage of maximum velocity

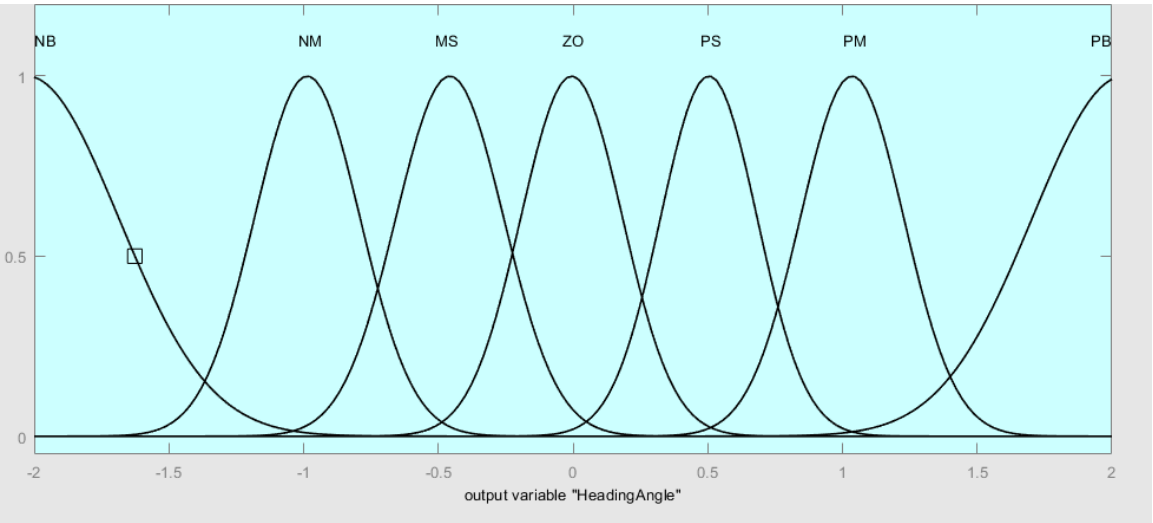

Figure 4.16: Output two: heading angle deviation

The defuzzifiacation takes the output from the designed “IF-THEN” rules and converts it to a crisp number. The centroid method is used for defuzzification in this fuzzy logic controller. Based on this designed fuzzy logic inference, the UAV can achieve path planning and obstacle avoidance in real time efficiently without mapping the environment and global path planning in advance.

4.3 Summary

This chapter illustrates the developed path planning and obstacle avoidance method including RRT, a fuzzy logic inference implemented in two-dimensional environments, Fuzzy-Kinodynamic RRT, and finally the improved fuzzy logic controller implemented in a more complicated environment without global path planning. Each section shows the design procedure of the proposed method in details.

Chapter 5 Simulations and Results

The proposed path planning and obstacle avoidance algorithms are illustrated and evaluated through a series of test cases by simulations. First, three basic cases and three sharp and complicated cases are used to test the performance of proposed fuzzy logic controller in 2D environment. In addition, the Fuzzy Kinodynamic RRT method which is a combination of rapidly-exploring random tree and fuzzy logic controller is tested by MATLAB simulation. Finally, the improved fuzzy logic inference simulation in a more complicated environment is done on Unreal Engine platform.

5.1 Fuzzy Logic Scenario Test 2D

5.1.1 Basic Cases

The simulation is performed using Python in Visual Studio Code. The Fuzzy logic method proposed in this thesis is implemented on 2D scenarios in order to test the performance. The obstacles are corridors when the agent is moving forward. First, the basic environment maps are used for testing. As shown in Figure 5.1, the starting point is on corner bottom right, while the desired zone is the red square on top left.

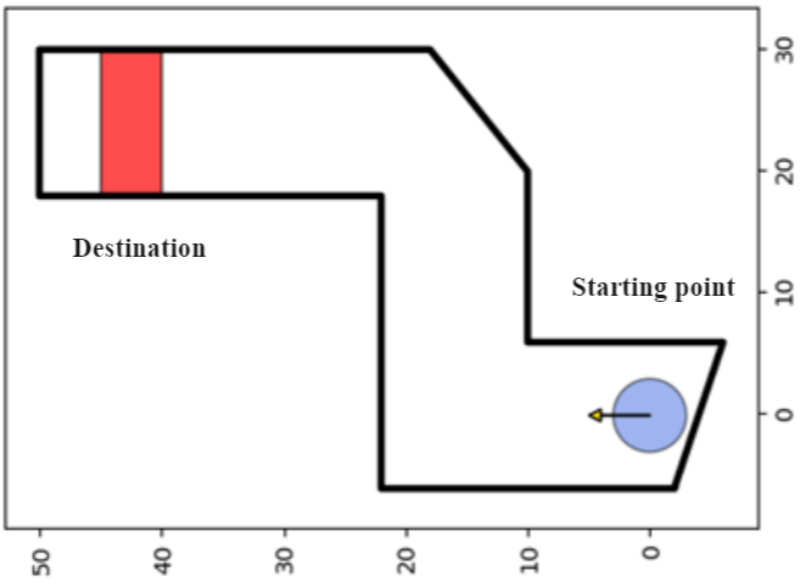

Fig. 5.1 Staring point

Firstly, the fuzzy logic rule 8 is fired to drive the agent move forward since at this time, front distance is medium, and the distance of left minus right is medium. Thus, the output of the fuzzy logic controller is medium with no steering angle change.

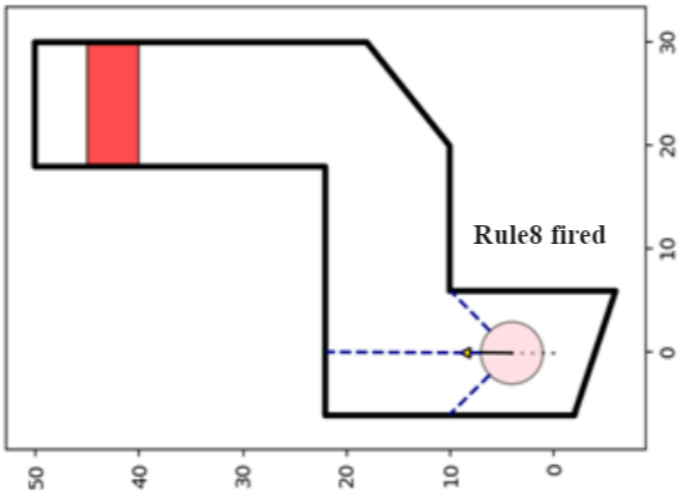

Fig. 5.2 Rule8 fired

The agent moves straight ahead until it comes to the first corner. The front distance to the wall is medium and the (left-right) distance is small. It means the front distance is average, but left distance is less than right distance. Thus Rule 4 is fired for the first corner, and output is set to large, to drive the agent turning right for obstacle avoidance.

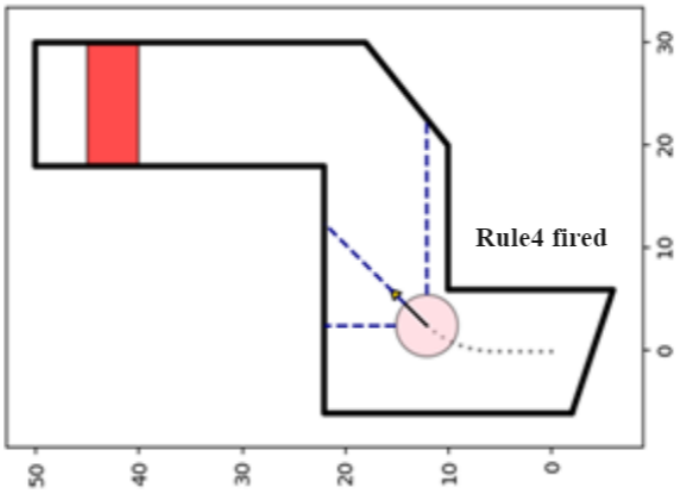

Fig. 5.3 Rule4 fired

After that, the agent safely passed the first corner, and the Rule 8 is fired in quite a short time until it comes to the second corner. In this circumstance, the front distance is medium, and the distance between left and right is large. Thus, the output of the fuzzy logic controller is small to drive the agent to turn left and avoid the obstacle.

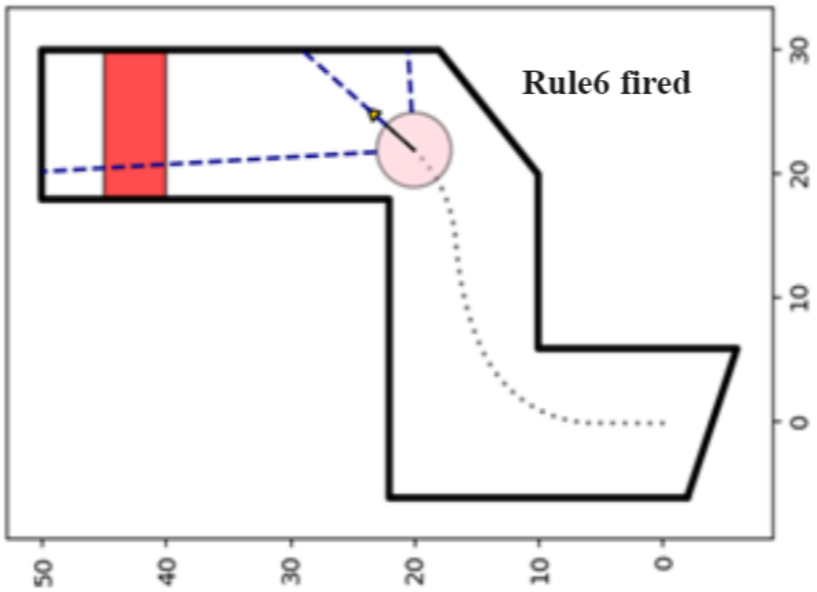

Fig. 5.4 Rule6 fired

Finally, all corners are passed and Rule 8 is fired to drive the agent moving straight ahead. The front distance is medium or large while the distance between left and right is medium, so the output of the fuzzy logic controller is medium with no steering angle.

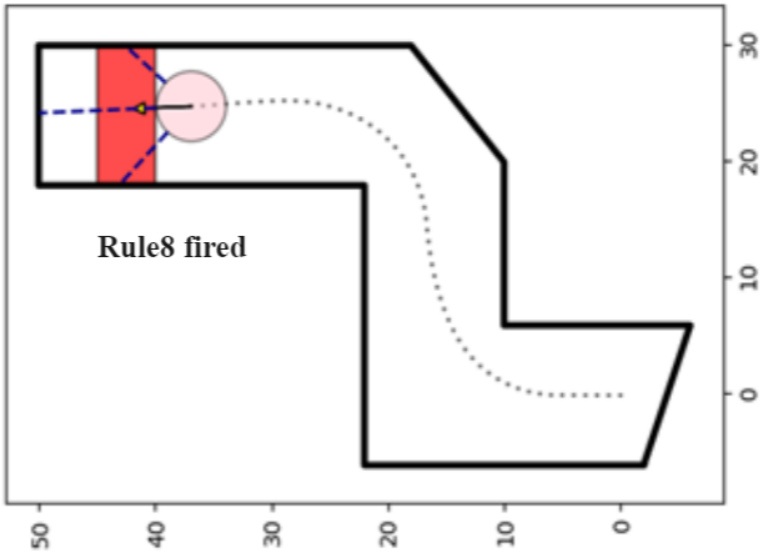

Fig. 5.5 Rule8 fired

The whole trajectory is shown in Figure 5.6 below. The agent can avoid the obstacle and arrive at the destination successfully and efficiently in this basic environment.

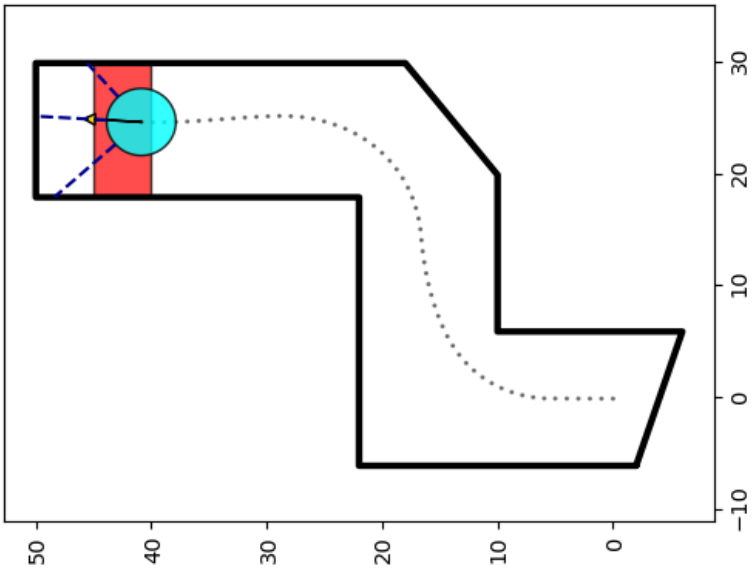

Fig. 5.6. Basic case 1

The fuzzy logic controller is also tested on another two basic environments. Different fuzzy logic rules are implemented in real time to help the agent to avoid obstacles dynamically and finally arrive at the desired position. The simulation result with the whole trajectory are shown in Figure 5.7 and Figure 5.8.

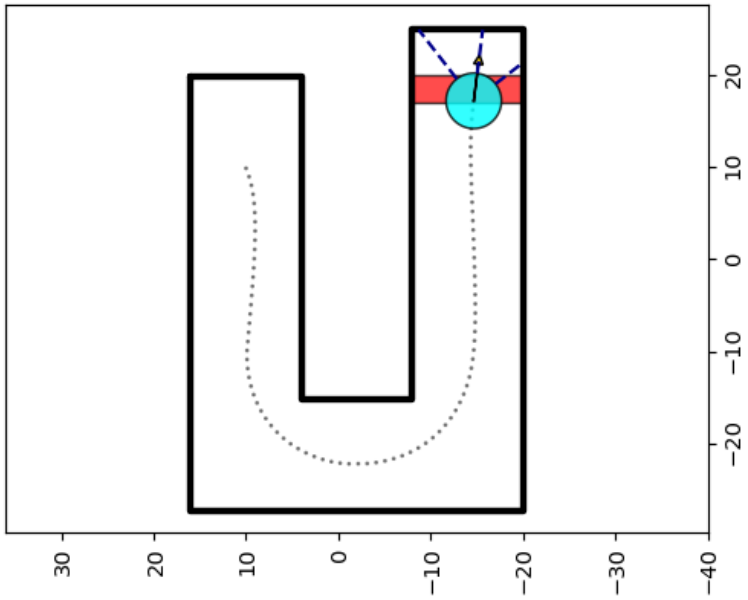

Fig. 5.7. Basic case 2

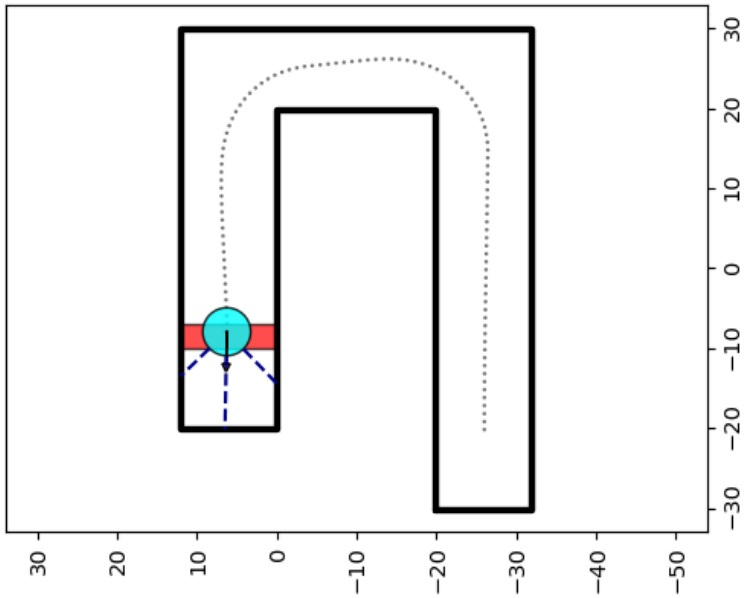

Fig. 5.8. Basic case 3

5.1.2 Complicated Cases

First, a complicated runway map is shown in Figure 5.9. Similarly, the starting point is on corner bottom right, while the desired zone is the red square on top left.

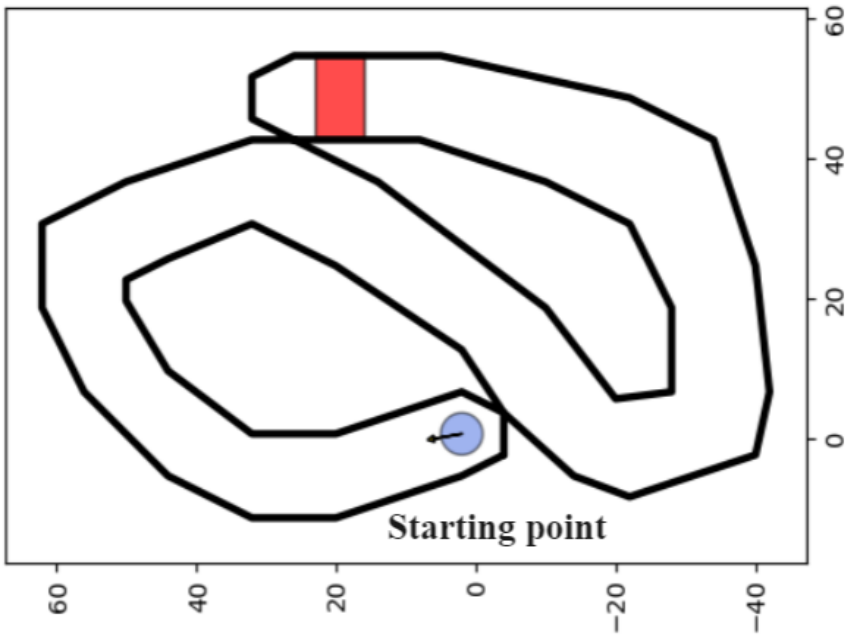

Fig. 5.9 Staring point

The fuzzy logic rule 8 is fired to drive the agent move forward at the beginning. It continues to go straight ahead until the first corner.

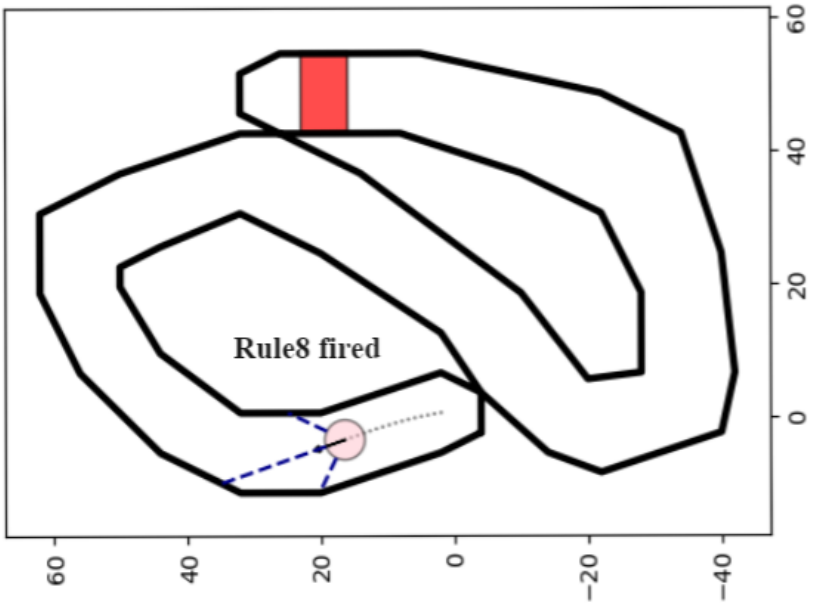

Fig. 5.10 Rule8 fired

Then the agent comes across the first corner. The front distance to the obstacle is medium and the (left-right) distance is small. Rule 4 is fired for the first corner, and output is set to large, to drive the agent turning right for obstacle avoidance.

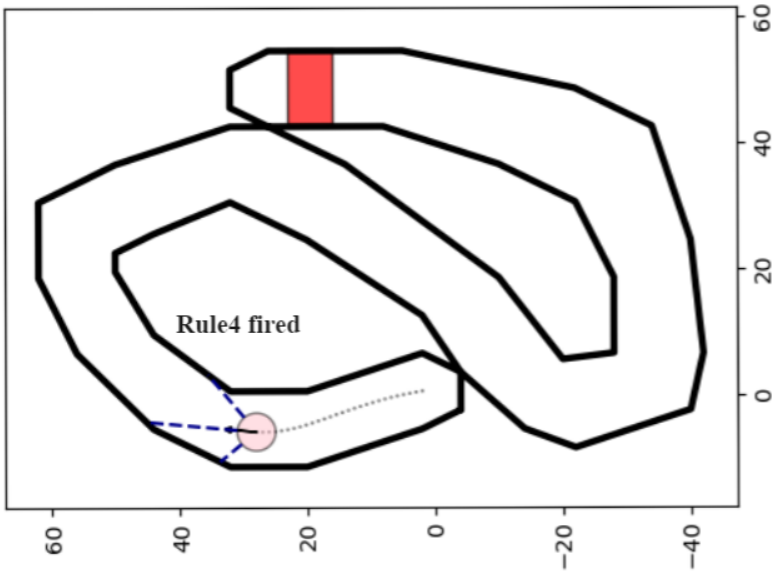

Fig. 5.11 Rule4 fired

In the second corner, the agent also implements Rule4 to turn right to avoid the obstacle while moving forward in the map.

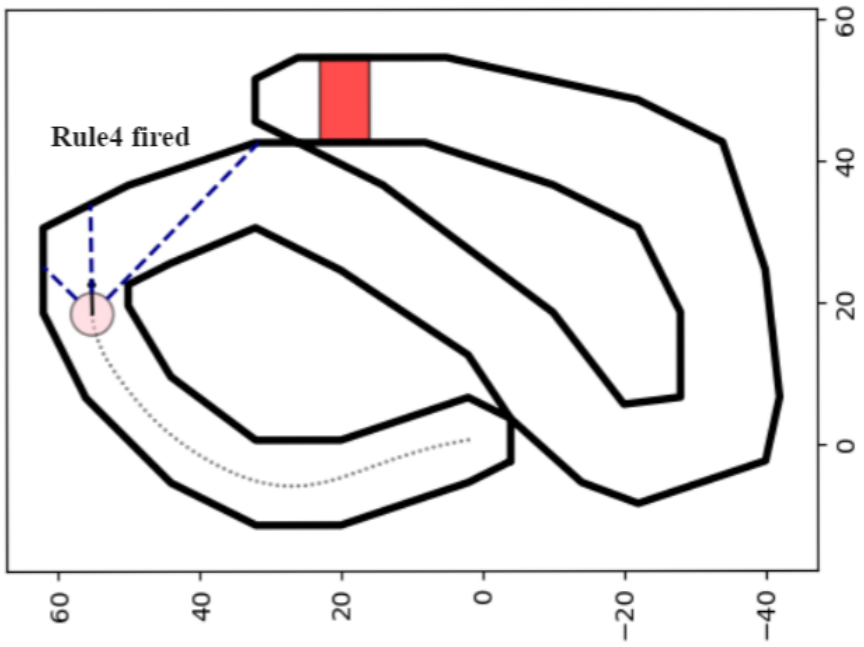

Fig. 5.12 Rule4 fired

After that, the front distance is large and the distance to left and right is almost the same. Thus, Rule8 is fired to help the agent move forward when there is no obstacle and threat at all.

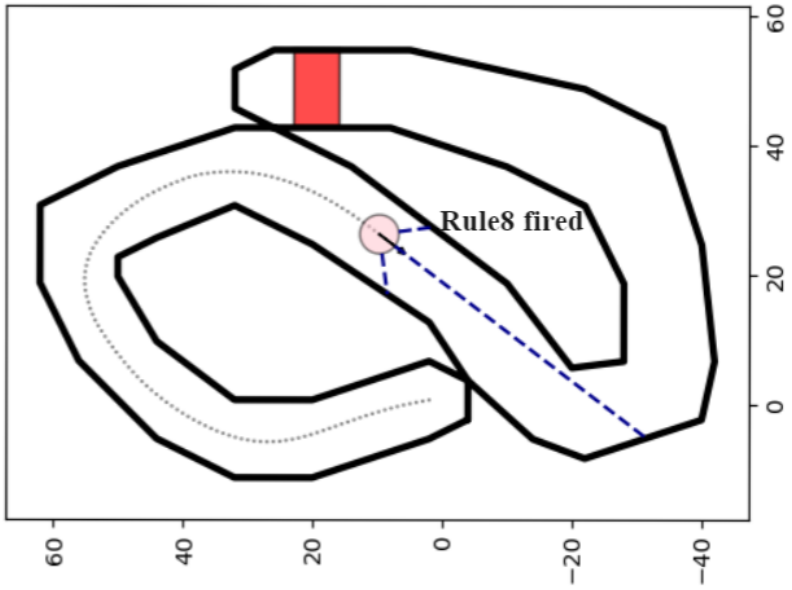

Fig. 5.13 Rule8 fired

Furthermore, the agent comes to the third corner. In this case, front distance is medium while left minus right distance is large. Thus, the output is small and Rule6 is implemented to drive the agent turning left.

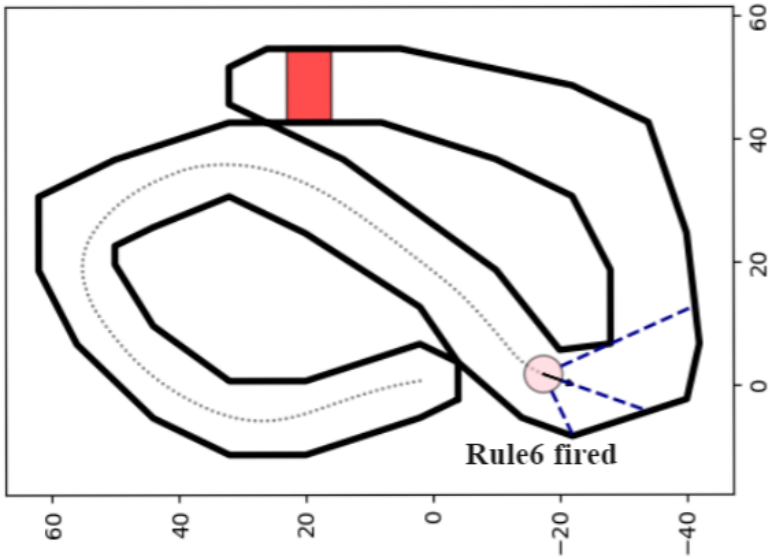

Fig. 5.14 Rule6 fired

Finally, all sharp corners are passed and Rule9 is fired to drive the agent turning left with a small angle. The agent can then move towards the destination until it arrives and stop.

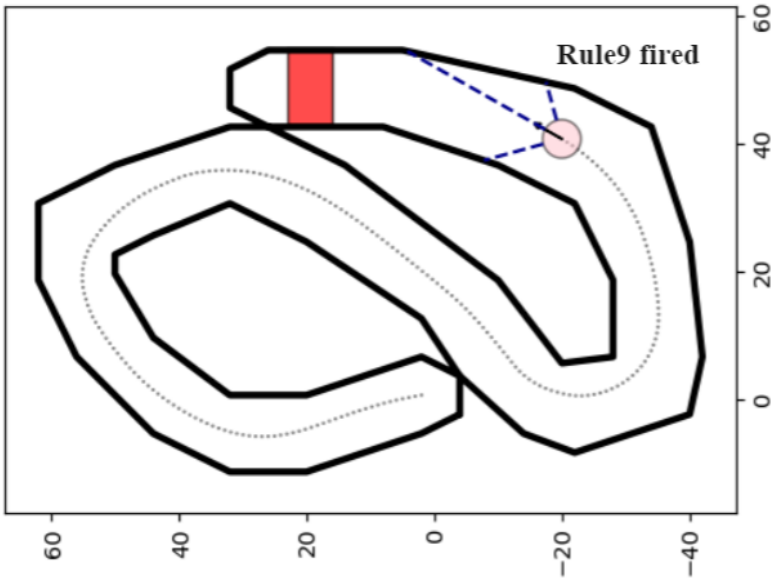

Fig. 5.15 Rule9 fired

In this complicated runway map, the majority of designed fuzzy logic rules are implemented in the obstacle avoidance process. Some rules are fired for a very short time so they are not described in detail. Overall, the fuzzy logic controller can work precisely for the complex environment, and the whole trajectory is shown in Figure 5.16.

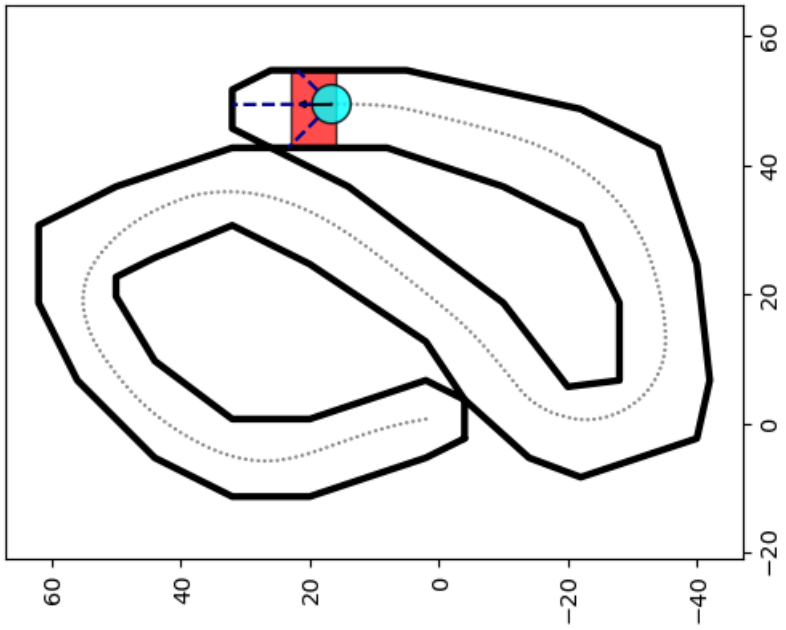

Fig. 5.16. Complicated runway

The fuzzy logic controller is also tested on another two complicated environments, which are maze map and multiple choice map respectively. The simulation results are shown in Figure 5.17 and Figure 5.18.

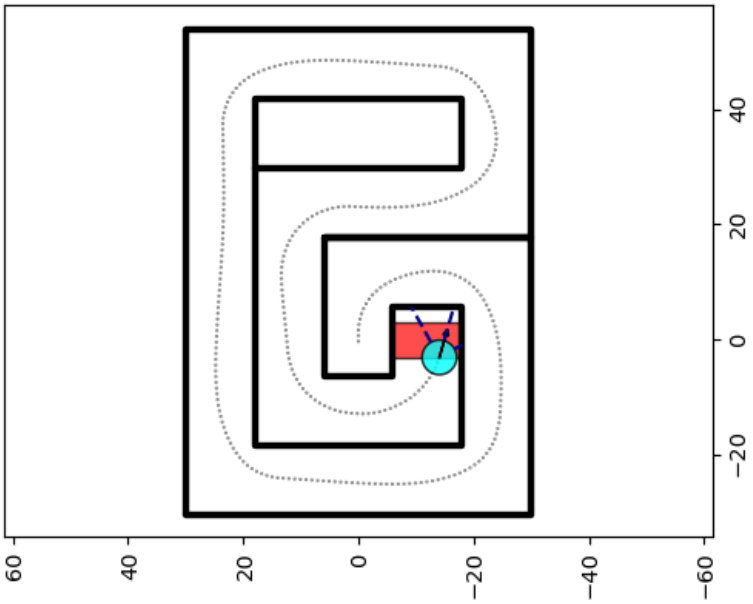

Fig. 5.17 maze map

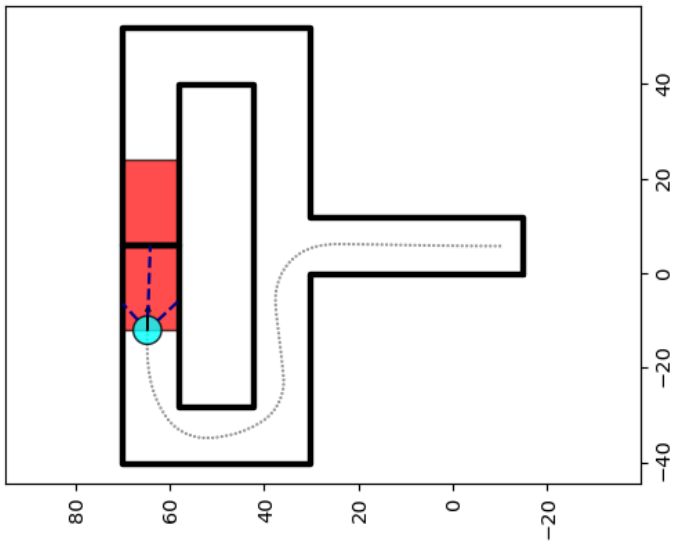

Fig. 5.18 multiple choice map

Even though there are multiple turns in these cases, they can also refer to the rule changes to make decisions and finally arrive at the destination successfully and precisely. The Fuzzy logic method proposed in this thesis can be implemented for each scene and has good performance on achieving obstacle avoidance.

5.2 Fuzzy Kinodynamic RRT method

Based on the Fuzzy logic method proposed above, we also test the performance of Fuzzy-Kinodynamic RRT method. The simulation result is shown in Fig. 5.19. In this scenario, path planning is done from the black circle (starting point) to blue circle (goal point), and optimized path using Fuzzy- Kinodynamic RRT method is expected where Fuzzy logic method is utilized to do obstacle avoidance during the path following.

The red dash line is the path generated by RRT. The green path is the optimized path done by Fuzzy-Kinodynamic RRT, and the blue trajectories show a set of path points where UAV can follow by. It is obvious that the optimized path done by Fuzzy-Kinodynamic RRT has shorter distance from starting point to desired point, and has a better performance to avoid collisions compared with the traditional RRT.

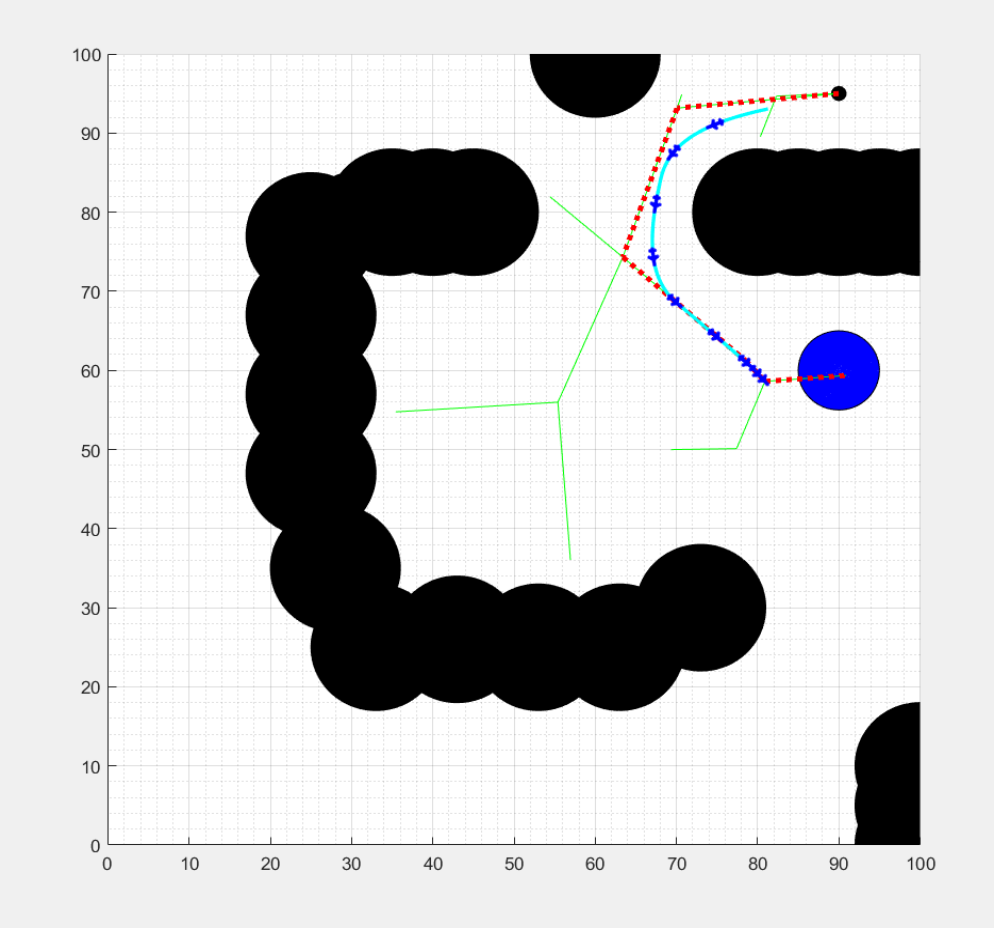

Figure 5.19 Fuzzy Kinodynamic RRT

The whole path planning and obstacle avoidance workflow is illustrated as follows:

First, the global trajectory is generated by RRT which is the red dash line in the simulation. The agent follows the trajectory until there is any obstacle detected within sensing radius L. Then, the fuzzy logic inference is activated to do local obstacle avoidance and corresponding fuzzy rules are implemented in real time. After collisions are avoided or there is no obstacle within sensing range, fuzzy logic inference is disabled. The agent finally moves along the previous trajectory until arriving at the destination. The pseudo code is shown in Table 5.1.

Table 5.1 Fuzzy Kinodynamic RRT method

```
1: do function BuildRRT()
2: Trajectory generated by RRT
3: Distance sensor is on with sensing radius L
4: Import fuzzy logic rules
5: Agent follows the generated trajectory
6: while obstacles detected within sensing range L
7:         Fuzzy Logic Inference Activated
8:         Analyze control output based on fuzzy rules
9:         Agent avoids obstacle based fuzzy logic controller
10: end while
11: return previous trajectory
12: return True
```

The whole path planning and obstacle avoidance using Fuzzy Kinodynamic RRT method is illustrated in details in Figure 5.20.

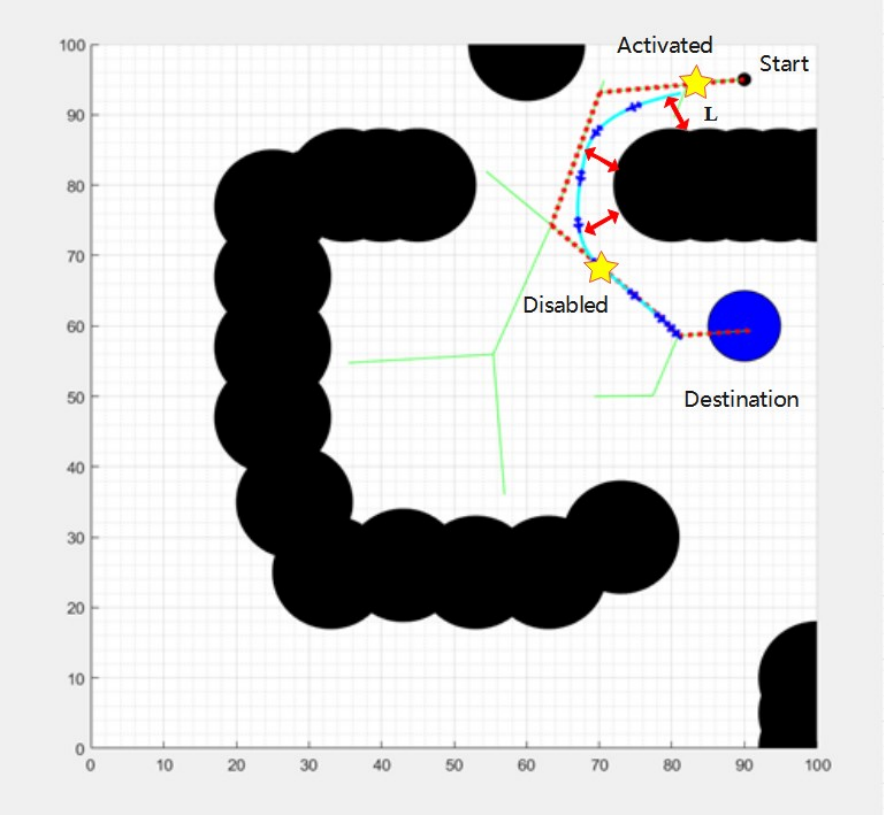

Figure 5.20 Fuzzy Kinodynamic RRT in details

* Starting point: black circle
* Destination: blue circle
* Obstacles: black zone
* Trajectory by RRT: red dash line
* Trajectory by Fuzzy Kinodynamic RRT: green line

In addition, the red arrow in the map is the sensing radius L, while the yellow stars are the points when fuzzy logic controller is activated and disabled.

At the beginning, RRT is implemented to generate the global trajectory. The agent is then set to follows the red dash trajectory until it comes to the first yellow star. In this circumstance, the obstacle is within sensing range L which is set to 5 in this map. So, the fuzzy logic inference is activated at this time and Rule 9 is fired to drive the agent turning left for obstacle avoidance.

When it comes to the second yellow star, the obstacle is no longer within sensing range. Thus, the fuzzy logic controller is disabled and the agent goes back to its previous path generated by RRT until it arrives at the destination. If there is any obstacle detected again, the system will recall the fuzzy logic inference to do obstacle avoidance, and finally the agent can reach the desired zone successfully.

The advantage of the proposed Fuzzy Kinodynamic RRT is to improve efficiency and optimize the trajectory with a shorter distance. The reason is that RRT is a sampling-based algorithm and tries to find the path with nodes stretching out randomly. In this case, the trajectory is not always the shortest and optimized one, since the nodes can be generated far away from the destination. With the help of Fuzzy Kinodynamic RRT method, the trajectory is optimized and the agent can move towards desired zone efficiently and precisely.

Algorithm Comparison

In this section, a comparison between Fuzzy Kinodynamic RRT and Artificial Potential Field (APF) will be shown in the following figures and tables. In the simulation, three group path planning results of both Fuzzy Kinodynamic RRT versus APF will be compared in the same scenarios.

The path of APF is generated in another simulation in MATLAB with the same set of obstacle points. Then this trajectory is imported into the same scenario of Fuzzy Kinodynamic RRT simulation for better visualization.

The first scenario is the same environment in section 5.2. As shown below, the starting coordinate is (90, 95) and end coordinate is (90, 60). The final trajectory generated by Fuzzy Kinodynamic RRT algorithm is shown as blue line in Figure 5.21.

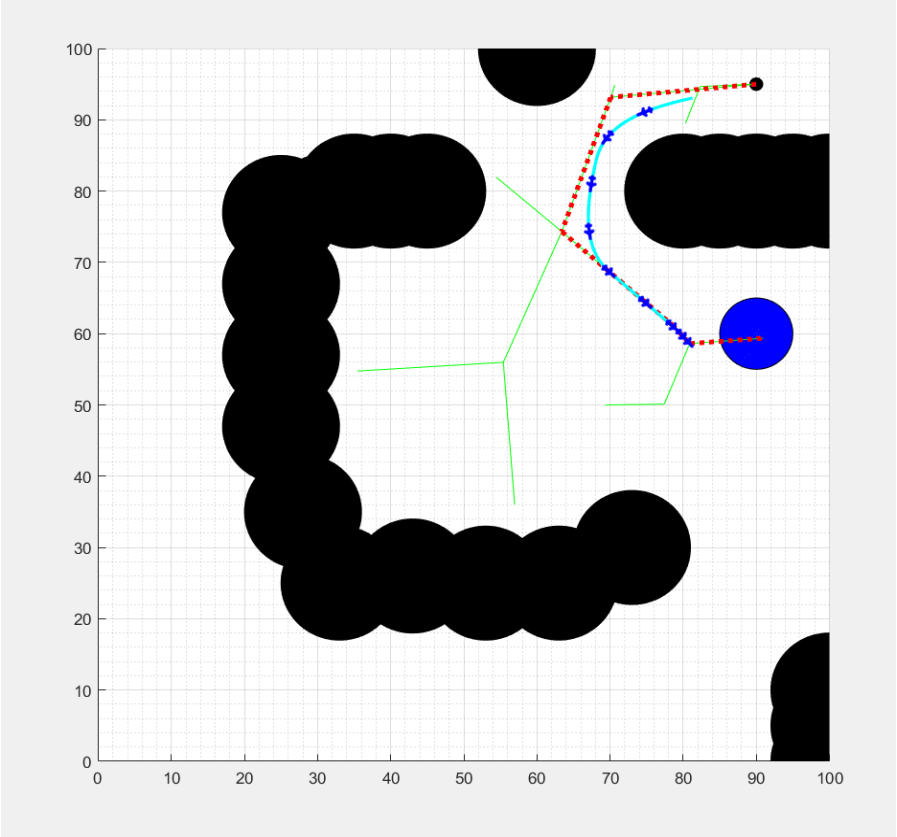

Figure 5.21 Fuzzy Kinodynamic RRT scenario one

However, APF suffers from local minima problem where the attractive forces and repulsive forces are balanced, so the agent is trapped into the local minima and stuck at one point as shown in Figure 5.22.

Local minima problem can also be found in the cases of closely spaced obstacles or dead end. Obviously, path planning fails when APF comes into local minima problem in some cases and it cannot generate the collision-free trajectory.

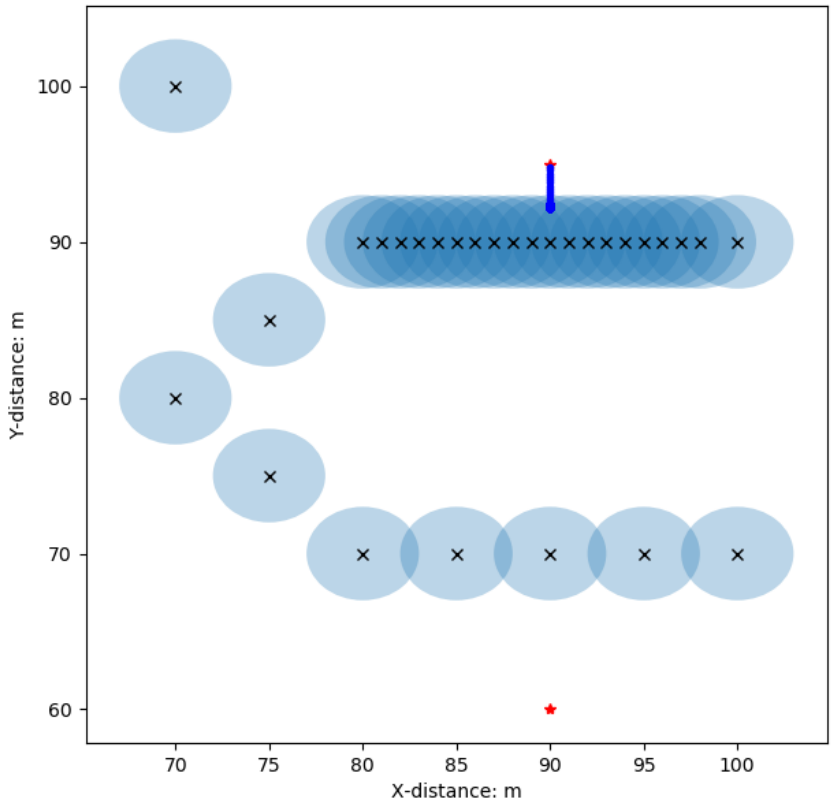

Figure 5.22 APF scenario one

The second case is shown in Figure 5.23 with starting coordinate (80, 60) and end point (8, 12). The path generated by Fuzzy Kinodynamic RRT is the blue one, while the trajectory generated by APF is the red line.

In this case, APF can generate the right path, and the trajectory is along the obstacles with the appropriate repulsive force radius which is adjusted in codes. The execution time of both algorithms are very similar.

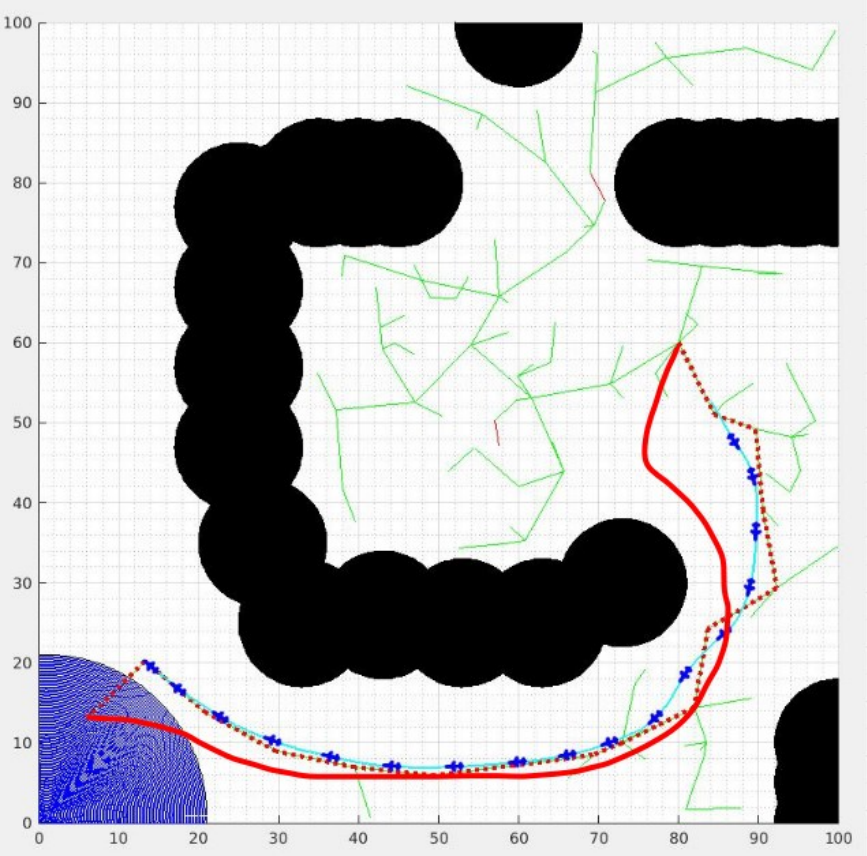

Figure 5.23 Fuzzy Kinodynamic RRT vs APF scenario two

The third case is shown in Figure 5.24 with starting coordinate (5, 60) and end point (85, 70). The path generated by Fuzzy Kinodynamic RRT is the blue one, while the trajectory generated by APF is the red line.

In this case, since the end zone is broad and the obstacles does not block the way when the nodes are stretching out towards the destination. Thus, the Fuzzy Kinodynamic RRT method gives a shorter path and costs less computer computation time.

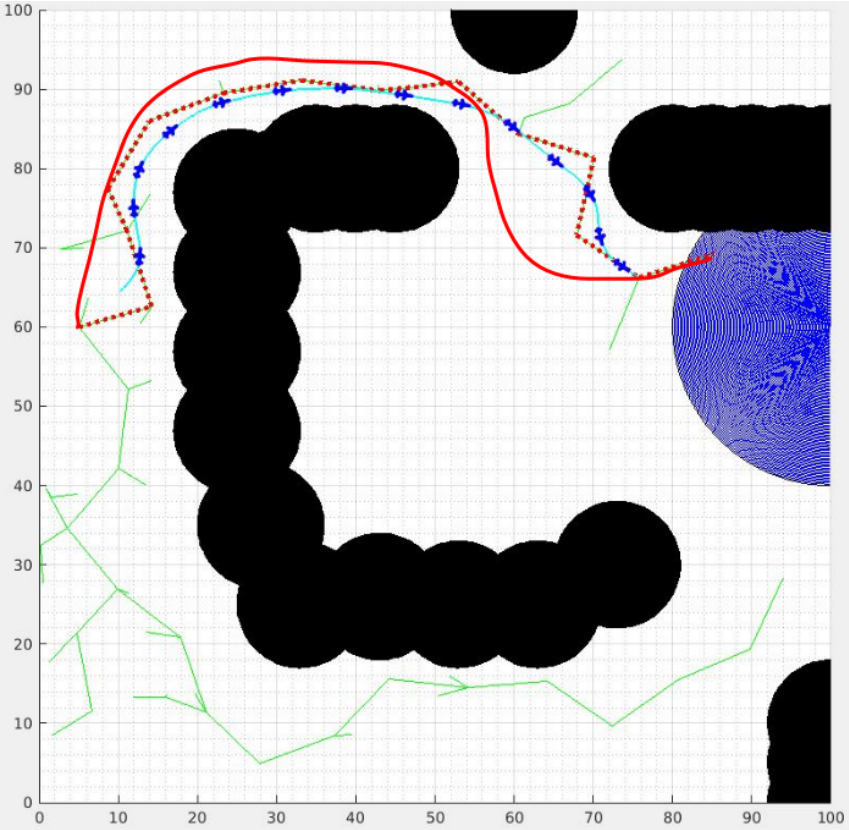

Figure 5.24 Fuzzy Kinodynamic RRT vs APF scenario three

The starting point, end point, execution time and path planning result of these two algorithms are shown in Table 5.2.

Table 5.2 Fuzzy Kinodynamic RRT vs APF details

	Start Point	*End Point*	*Execution Time*	*Result*
Fuzzy Kinodynamic RRT	**(80, 60)**	**(8, 12)**	**3.285s**	Success
	(5, 60)	**(85, 70)**	**3.692s**	Success
	(90, 95)	**(90, 60)**	**2.371s**	Success
APF	**(80, 60)**	**(8, 12)**	**3.592s**	Success
	(5, 60)	**(85, 70)**	**4.916s**	Success
	(90, 95)	**(90, 60)**	**Infinity**	Failed

In the three comparison groups, the Fuzzy Kinodynamic RRT method costs less execution time than APF, and the trajectories generated by two algorithms are very similar when successful. However, APF is not as general as proposed Fuzzy Kinodynamic RRT to use, since path planning may be trapped at one point when attractive forces and repulsive forces are balanced.

5.3 Fuzzy Logic Test in 3D environment

5.3.1 AirSim Block Environment Test

Finally, we implement the fuzzy logic method for obstacle avoidance in AirSim block environment. At first, a drone moves straight ahead towards an orange ball at a constant speed. The fuzzy logic system is activated when the front distance of the drone to obstacle is less than or equal to a specific value which can be set in program. Then the drone avoids the ball by flying to left or right. The fuzzy logic is the same as implemented on 2D fuzzy logic inference. It will fly with a certain altitude, and continues to move forward when fuzzy logic system is disconnected since there are no obstacles on the way.

The simulation of UAV on Unreal Engine is done using Fuzzy logic method to realize obstacle avoidance. The process of UAV obstacle avoidance from side for the same orange ball is shown in Figs. 5.25, 5.26 and 5.27. The simulation results verify the effectiveness and accuracy of the proposed Fuzzy logic technique for obstacle avoidance of UAV in the block environment.

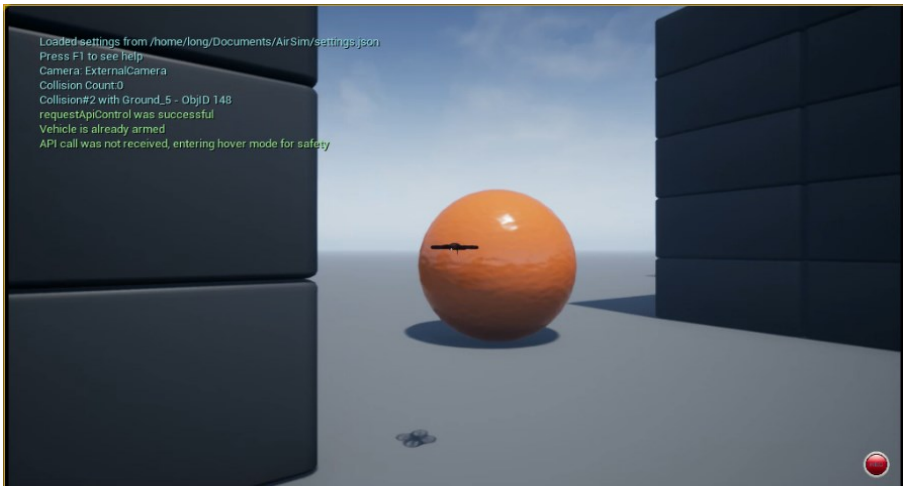

Fig. 5.25 Side obstacle avoidance start pose

Fig. 5.26 Side obstacle avoidance process pose

Fig. 5.27 Side obstacle avoidance end pose

5.3.2 AirSim Landscape Environment Test

Furthermore, the improved fuzzy logic controller is tested on AirSim Landscape environment, which is a big and complicated map in Unreal Engine with a lot of trees as the obstacles. This method is applied and only tested with the AirSim built-in UAV, in Unreal Engine, Windows 10 platform. The overview of the Landscape environment is shown in Figure 5.28 and Figure 5.29. Although the fuzzy logic methodology is designed with the assumption of constant altitude for the UAV, the method can be implemented in three-dimensional environment autonomous navigation and obstacle avoidance in real time.

Figure 5.28 Landscape Overview1

Figure 5.29 Landscape Overview2

The core part of this environment is selected for testing the improved fuzzy logic controller in Figure 5.30. As shown in the figure, the start point is located at initial position of the coordinate, and the destination is marked with red star. The main objective is UAV autonomous navigation by implementing fuzzy logic controller to drive the UAV to arrive at the destination and avoid obstacles (forests in this environment) in real time.

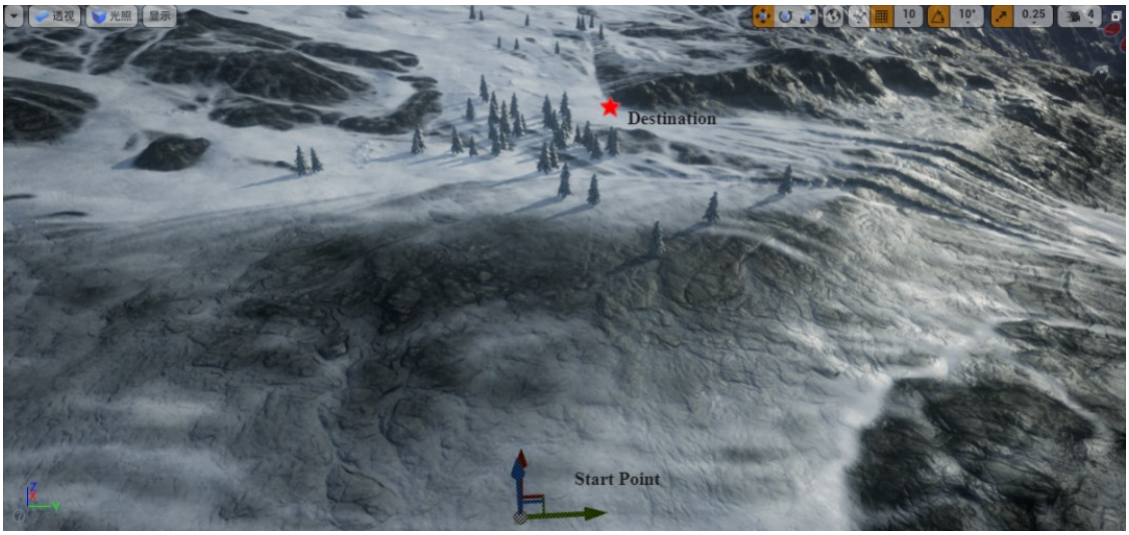

Figure 5.30 UAV start point and destination

There are some default settings and configurations for the simulation as follows:

First, the initial altitude of the UAV is set to be 5 meters above the ground. Moreover, as the coordinate frame shown in Figure 5.30, the UAV is initially heading forward along the red one. Furthermore, the position coordinates of the target and the UAV are known with the help of GPS. Also, the AirSim has its built-in distance sensors [60] for developers to use. The setting codes regarding GPS and distance sensor in this simulation are shown in Figure 5.31.

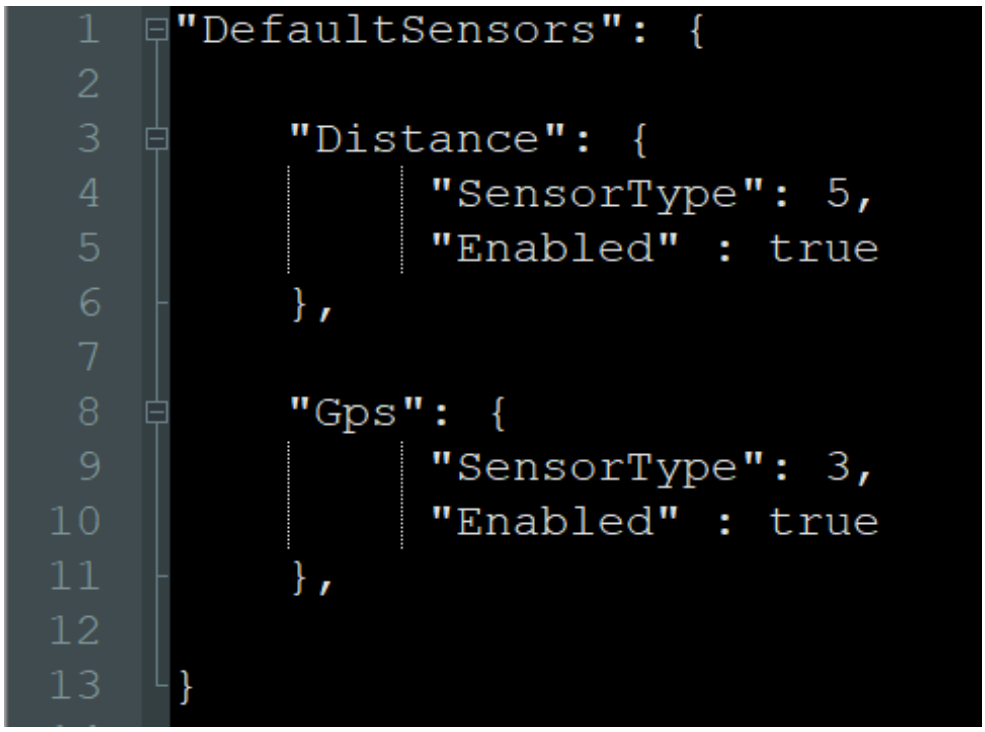

Figure 5.27 Sensors setting

The whole procedure of the UAV autonomous navigation using the improved fuzzy logic controller in the simulation are illustrated in details as follows:

First, the drone is taken off at the starting position as shown in Figure 5.32. In this circumstance, there are no obstacles within the sensing radius (D_o = Very Far). The main objective of the fuzzy logic controller is to drive the UAV to fly towards the destination directly. Since there is error between the heading angle of the UAV and target, the fuzzy logic controller tends to change the heading angle Θ_c to Θ_t for driving the UAV towards the target. In this case, the obstacles are very far away from the UAV, and the target is on the UAV's right side with small angles.

Hence, Rule 8(If D_o = Very Far and Θ_t = Positive Small, then Θ_c = Positive Small) is fired for heading angle control. In addition, Rule 1 (If D_o = Very Far and D_t = Far, then V = Very Fast) is also fired for speed control, because the distance between the UAV and the target is far. The UAV is flying straight towards the target by setting the heading angle Θ_c to Θ_t with a fast speed as shown in Figure 5.33.

Figure 5.32Take off

Figure 5.33 Rule1 and Rule8 fired

Then, the UAV continues to fly until it is about to comes to the first obstacle on the way in Figure 5.34.

Figure 5.34 First obstacle on the way

In this case, the distance between the UAV and the object is medium, and the angle between the UAV and the object is positive small. Hence, Rule 18 (If D_o = Medium and Θ_o = Positive Small, then V = Slow and Θ_c = Negative Medium) is fired

to drive the drone to turn left and avoid the obstacle. The output of the fuzzy logic controller are slow speed and negative medium heading angle change. As shown in Figure 5.35, Rule 18 is fired to help the UAV to avoid the obstacle with a low speed.

Figure 5.35 Rule 18 fired

After avoiding the first obstacle, the distance between the UAV and obstacle is very far away and the distance between the UAV and the target is medium. Thus, Rule 2 (If D_o = Very Far and D_t = Medium Distance, then V = Slow) is fired. Also, Rule 9 (If D_o = Very Far and Θ_t = Positive Medium, then Θ_c = Positive Medium) is fired since the angle between the heading angle of the UAV and the target becomes larger after turning left for avoiding the first obstacle. Hence, the Rule 2 and Rule 9 are fired to drive the UAV flying towards the target directly with a low speed as shown in Figure 5.36.

Figure 5.36 Rule2 and Rule9 fired

Then, the UAV is about to come to the second obstacle on the way in Figure 5.37. Similar to the case of avoiding the first obstacle, Rule 18 (If D_o = Medium and Θ_o = Positive Small, then V = Slow and Θ_c = Negative Medium) is fired to drive the drone to turn left and avoid the second obstacle. The UAV successfully avoids the obstacle with a low speed as shown in Figure 5.38.

Figure 5.37 Second obstacle on the way

Figure 5.38 Rule18 fired

Then the UAV comes to the third obstacle on the way. In this case, Rule 17 (If D_o = Medium and Θ_o = Negative Small, then V = Slow and Θ_c = Positive Medium) is fired to drive the UAV to turn right with a slow speed to avoid the obstacle as shown in Figure 5.39.

Figure 5.39 Rule17 fired

Finally, the distance between the UAV and the target is very small and the UAV is about to arrive at the destination. Also, there is no obstacle in sensing range and the distance between the UAV and the obstacle is very far. In this case, Rule 3 (If D_o = Very Far and D_t = Close, then V = Very Slow) is fired to make the UAV move slowly to the final destination.

Figure 5.40 Rule3 fired

Figure 5.41 Mission Completed

5.4 Discussion

As shown in the simulation above, firstly the designed fuzzy logic inference is tested on three basic cases and three complicated cases. The whole obstacle avoidance procedure is illustrated step by step, and the result shows all rules are working successfully. Then the proposed Fuzzy Kinodynamic RRT is tested in simulation. With the combination of designed fuzzy logic controller and rapidly-exploring random tree, this method can improve the efficiency of RRT and give an optimized trajectory solution which the UAV can follow in the simulation. Finally, the improved fuzzy logic controller which embeds path planning and is more robust and efficient is designed to avoid obstacles for the UAV in real time.

Chapter 6 Conclusions and Future Works

6.1 Conclusions

In this dissertation, several methods of UAV path planning and obstacle avoidance are proposed. The main contributions of the research work are summarized as follows:

- A fuzzy inferencing system is developed for supporting the UAV to avoid obstacle dynamically in unknown environment. This fuzzy system consists of two inputs and one output. The rules for different inputs and output for the fuzzy logic inference are set up in the form of "IF-THEN" statements, and are based on heuristics and human experience with navigating through an environment, which is similar to driving a car.
- Fuzzy-Kinodynamic RRT is a combination method which uses RRT algorithm to do global path planning and utilizes fuzzy logic system to avoid obstacles. The UAV starts to follow the path generated by global path planning algorithm and the fuzzy logic system is activated when it comes across new obstacles. The UAV can avoid obstacles dynamically according to the rules designed in this research work and then fly back to the previous path.
- A more sophisticated and robust fuzzy logic controller with four inputs, two outputs and totally 40 fuzzy logic rules is designed for dynamically path planning and obstacle avoidance in unknown environments without the support of global path planning as implemented in Fuzzy Kinodynamic RRT method.
- This dissertation proposes an algorithm on the combination of UAV global path planning and sensor-based real-time obstacle avoidance, and validates the effectiveness of this method through simulation mainly on Unreal Engine AirSim platform.

6.2 Future works

Based on the current research in this dissertation, the following future directions are outlined:

- The real flight test has not been implemented even though this method works well in the simulation. The physical test for UAV can be done with Pixhawk and distance sensors.
- The proposed Fuzzy Kinodynamic RRT performs well in simple environment but has its own constrains. The agent needs to recall RRT to generate path again when the fuzzy logic inference is disabled but the agent is too far way to go back to the previous trajectory, which can reduce the efficiency in this case.
- The fuzzy logic algorithm developed in this work was primarily designed for 2D environment and as the next step it may be extended to more general 3D cases. The early simulations that provided in section 5.3.1 illustrate the algorithm's potential for such these cases.
- The UAV dynamics are not considered by Fuzzy Kinodynamic RRT method. Future work will include the UAV dynamics and make an improvement for the algorithm.
- The proposed methods are currently tested in the environment where there are only obstacles and one drone. Future work will include the simulation where other drones are added into the map for validation.

Bibliography

[1] Y. J. Heo and W. K. Chung, "RRT-based path planning with kinematic constraints of AUV in underwater structured environment," 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), 2013, pp. 523-525, doi: 10.1109/URAI.2013.6677328.

[2] G. F. Shao, Z. S. Li, Y. L. Wen and L. M. Zhuang, "The Behavior Coding of Artificial Life Body Based on Dynamic Potential Field Approach," 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 2006, pp. 2546-2550, doi: 10.1109/WCICA.2006.1712821.

[3] P. Crepon, A. M. Panchea and A. Chapoutot, "Reliable Motion Plannning for a Mobile Robot," 2018 Second IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, 2018, pp. 413-418, doi: 10.1109/IRC.2018.00085.

[4] J. Cao, Y. Li, S. Zhao and X. Bi, "Genetic-Algorithm-Based Global Path Planning for AUV," 2016 9th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2016, pp. 79-82, doi: 10.1109/ISCID.2016.2027.

[5] J. Cao, Y. Li, S. Zhao and X. Bi, "Genetic-Algorithm-Based Global Path Planning for AUV," 2016 9th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 2016, pp. 79-82, doi: 10.1109/ISCID.2016.2027.

[6] D. W, C. L, N. G, Y. S, T. G and G. L, "Local Path Planning of Mobile Robot Based on Artificial Potential Field," 2020 39th Chinese Control Conference (CCC), Shenyang, China, 2020, pp. 3677-3682, doi: 10.23919/CCC50068.2020.9189250.

[7] Y. Huang, H. Hu and X. Q. Liu, "Obstacles avoidance of artificial potential field method with memory function in complex environment," 2010 8th World Congress on Intelligent Control and Automation, Jinan, China, 2010, pp. 6414-6418, doi: 10.1109/WCICA.2010.5554309.

[8] C. W. Warren, "Global path planning using artificial potential fields," Proceedings, 1989 International Conference on Robotics and Automation, Scottsdale, AZ, USA, 1989, pp. 316-321 vol.1, doi: 10.1109/ROBOT.1989.100007.

[9] K. V and J. L. Yu, "3D path planning for mobile robots using annealing neural network," 2009 International Conference on Networking, Sensing and Control, Okayama, Japan, 2009, pp. 130-135, doi: 10.1109/ICNSC.2009.4919259.

[10] M. Ohtani, H. Iwai and H. Sasaoka, "Improvement of position estimation accuracy using multiple access points in terminal position estimation based on position fingerprint," 2014 International Symposium on Antennas and Propagation Conference Proceedings, Kaohsiung, Taiwan, 2014, pp. 399-400, doi: 10.1109/ISANP.2014.7026698.

[11]M. A. BÜLBÜL, C. ÖZTÜRK, V. İLÇİ and Ý. M. OZULU, "Two-Dimensional Error Estimation in Point Positioning with Fuzzy Logic," 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 2018, pp. 1-4, doi: 10.1109/IDAP.2018.8620901.

[12]X. Y. Xu, J. Xie and K. Xie, "Path Planning and Obstacle-Avoidance for Soccer Robot Based on Artificial Potential Field and Genetic Algorithm," 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 2006, pp. 3494-3498, doi: 10.1109/WCICA.2006.1713018.

[13]Z. Q. Yang, L. B. Liu, Z. H. Tan and W. L. Liu, "Application of Adaptive Genetic Algorithm in flexible inspection path planning," 2008 27th Chinese Control Conference, Kunming, China, 2008, pp. 75-80, doi: 10.1109/CHICC.2008.4605656.

[14]P. Tang, C. Gao, C. Tang, G. Lee and F. Lu, "An improved genetic algorithm for optimizing resource allocation using knowledge evolution and natural evolution," 2010 World Automation Congress, Kobe, Japan, 2010, pp. 1-5.

[15]J. Martin, H. T, J. Findlay, N. Stoesser, L. Pankhurst, I. Navickaite, N. Maio, D. W. Eyre, G. Toogood, N. M. Orsi, A. Kirby, N. Young, J. F. Turton, R. L. Hill, K. L. Hopkins, N. Woodford, T. E. Peto, A. S. Walker, D. W. Crook, M. H. Wilcox, Covert dissemination of carbapenemase-producing Klebsiella pneumoniae (KPC) in a successfully controlled outbreak: long- and short-read whole-genome sequencing demonstrate multiple genetic modes of transmission, Journal of Antimicrobial Chemotherapy, Volume 72, Issue 11, November 2017, Pages 3025–3034.

[16]L. Zhang, H. Jiang, F. Wang, D. Feng and Y. Xie, "T-Sample: A Dual Reservoir-Based Sampling Method for Characterizing Large Graph Streams," 2019 IEEE 35th International Conference on Data Engineering (ICDE), Macao, China, 2019, pp. 1674-1677, doi: 10.1109/ICDE.2019.00170.

[17]S. Karaman, and F. E (2011) “Sampling-based Algorithms for Optimal Motion Planning”, Int. Journal of Robotics Research, vol. 30, no. 7, pp. 846–894.

[18]C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), volume 2, pages 1178–1183. IEEE, 2003.

[19]G. I, J. P. (2005) “Improving the efficiency of Rapidly- exploring Random Trees Using a Potential Function Planner”, Proc. of 44th IEEE Conf. on Decision and Control, and the European Control Conference, pp. 7965–7970.

[20]N. Pradhan, T. Burg and S. Birchfield, "Robot crowd navigation using predictive position fields in the potential function framework," Proceedings of the 2011 American Control Conference, San Francisco, CA, 2011, pp. 4628-4633.

[21]X. Y. W, X. J. L, Y. G, S and R. W, "Bidirectional Potential Guided RRT* for Motion Planning," in IEEE Access, vol. 7, pp. 95046-95057, 2019.

[22]C. Moon and C. W, "Kinodynamic Planner Dual-Tree RRT (DT-RRT) for Two-Wheeled Mobile Robots Using the Rapidly Exploring Random Tree," in IEEE Transactions on Industrial Electronics, vol. 62, no. 2, pp. 1080-1090, Feb. 2015.

[23]L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma and N. Zheng, "A fast RRT algorithm for motion planning of autonomous road vehicles," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp.

[24]K. S, M, A. (2008) “Density Avoided Sampling: An Intelligent Sampling Technique for Rapidly-Exploring Random Trees”, Eighth Int. Conf. on Hybrid Intelligent Systems, HIS, pp.672–677.

[25]E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd and L. E. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," in IEEE Transactions on Robotics, vol. 21, no. 4, pp. 597-608, Aug. 2005.

[26]K. J. and L. S. M. (Apr. 2000) RRT-connect: “An efficient approach to single-query path planning”, in Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 995–1001.

[27]V. Vonásek and R. Pěniĝka, "Sampling-based motion planning of 3D solid objects guided by multiple approximate solutions," 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 2019, pp. 1480-1487.

[28]D. Zhang, Y. Xu and X. Yao, "An Improved Path Planning Algorithm for Unmanned Aerial Vehicle Based on RRT-Connect," 2018 37th Chinese Control Conference (CCC), Wuhan, 2018, pp. 4854-4858.

[29]D. Brandt, "Comparison of A and RRT-Connect Motion Planning Techniques for Self-Reconfiguration Planning," 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, 2006, pp. 892-897.

[30]C. Lau and K. Byl, "Smooth RRT-connect: An extension of RRT-connect for practical use in robots," 2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), Woburn, MA, 2015, pp. 1-7.

[31]M. Ragaglia, M. Prandini and L. Bascetta, "Poli-RRT*: Optimal RRT-based planning for constrained and feedback linearisable vehicle dynamics," 2015 European Control Conference (ECC), Linz, 2015, pp. 2521-2526.

[32]K. W Lee, J. C. Koo, H. R. Choi and H. Moon, "An RRT* path planning for kinematically constrained hyper-redundant inpipe robot," 2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI), Goyang, 2015, pp. 121-128.

[33]L. Chen, Y. Shan, W. Tian, B. Li and D. Cao, "A Fast and Efficient Double-Tree RRT-Like Sampling-Based Planner Applying on Mobile Robotic Systems," in IEEE/ASME Transactions on Mechatronics, vol. 23, no. 6, pp. 2568-2578, Dec. 2018.

[34]C. Wang and M. Q. M, "Variant step size RRT: An efficient path planner for UAV in complex environments," 2016 IEEE International Conference on Real-time Computing and Robotics (RCAR), Angkor Wat, 2016, pp. 555-560.

[35]L. E. Kavraki, M. N. Kolountzakis and J. Latombe, "Analysis of probabilistic roadmaps for path planning," in IEEE Transactions on Robotics and Automation, vol. 14, no. 1, pp. 166-171, Feb. 1998, doi: 10.1109/70.660866.

[36]A. Sanchez and R. Zapata, "Sensor-based probabilistic roadmaps for car-like robots," Proceedings of the Fifth Mexican International Conference in Computer Science, 2004. ENC 2004., Colima, Mexico, 2004, pp. 282-288, doi: 10.1109/ENC.2004.1342618.

[37]Z. Lee and X. Chen, "Path planning approach based on probabilistic roadmap for sensor based car-like robot in unknown environments," 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583), The Hague, 2004, pp. 2907-2912 vol.3, doi: 10.1109/ICSMC.2004.1400774.

[38]F. Yan, Y. Zhuang and J. Xiao, "3D PRM based real-time path planning for UAV in complex environment," 2012 IEEE International Conference on Robotics and Biomimetics (ROBIO), Guangzhou, 2012, pp. 1135-1140, doi: 10.1109/ROBIO.2012.6491122.

[39]J. Chen, Y. Zhou, J. Gong and Y. Deng, "An Improved Probabilistic Roadmap Algorithm with Potential Field Function for Path Planning of Quadrotor," 2019 Chinese Control Conference (CCC), Guangzhou, China, 2019, pp. 3248-3253, doi: 10.23919/ChiCC.2019.8865585.

[40]C. Ju, Q. Luo and X. Yan, "Path Planning Using an Improved A-star Algorithm," 2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan), Jinan, China, 2020, pp. 23-26, doi: 10.1109/PHM-Jinan48558.2020.00012.

[41]O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," Proceedings. 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 1985, pp. 500-505, doi: 10.1109/ROBOT.1985.1087247.

[42]J. Lee, Y. Nam and S. Hong, "Random force based algorithm for local minima escape of potential field method," 2010 11th International Conference on Control Automation Robotics & Vision, Singapore, 2010, pp. 827-832, doi: 10.1109/ICARCV.2010.5707422.

[43]Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," Proceedings. 1991 IEEE International Conference on Robotics and Automation, Sacramento, CA, USA, 1991, pp. 1398-1404 vol.2, doi: 10.1109/ROBOT.1991.131810.

[44]A. Torralba, F. Colodro and L. G. Franquelo, "A fuzzy-logic controller with on-chip learning, employing stochastic logic," Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference, Orlando, FL, USA, 1994, pp. 1759-1764 vol.3, doi: 10.1109/FUZZY.1994.343949.

[45]T. M. Sugeno "Fuzzy Control of Model Car" Journal of the Robotic Society of Japan vol. 6 no. 6 pp. 536-541 Dec. 1988.

[46]Z. Zhao, W. Xie and A. B. Rad, "A Cascaded Fuzzy Model of Friction over Large Temperature Variation," NAFIPS 2006 - 2006 Annual Meeting of the North American Fuzzy Information Processing Society, Montreal, Que., 2006, pp. 160-165.

[47]L. A. Zadeh, "Toward a restructuring of the foundations of Fuzzy logic (FL)," 1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228), Anchorage, AK, USA, 1998, pp. 1676-1677 vol.2.

[48]S. Li and X. Sun, "A Real-Time UAV Route Planning Algorithm Based on Fuzzy Logic Techniques," 2006 6th World Congress on Intelligent Control and Automation, Dalian, 2006, pp. 8750-8753.

[49]G. Zhou, N. Wang, X. Lu and J. Ma, "Research on the Fuzzy Algorithm of Path Planning of Mobile Robot," 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), Dalian, 2017, pp. 633-637.

[50]L. A. Zadeh, "Fuzzy logic," in Computer, vol. 21, no. 4, pp. 83-93, April 1988, doi: 10.1109/2.53.

[51]J. Yen, R. Langari. 1999. Fuzzy Logic: Intelligence, Control and Information.

[52]S. Wen and L. Wang, "A study on obstacle avoidance for mobile robot based on Fuzzy logic control and adaptive rotation," Proceedings of the 10th World Congress on Intelligent Control and Automation, Beijing, 2012, pp. 753-757, doi: 10.1109/WCICA.2012.6357978.

[53]J. Berisha, X. Bajrami, A. Shala and R. Likaj, "Application of Fuzzy Logic Controller for obstacle detection and avoidance on real autonomous mobile robot," 2016 5th Mediterranean Conference on Embedded Computing (MECO), Bar, 2016, pp. 200-205, doi: 10.1109/MECO.2016.7525740.

[54]R. Malhotra and A. Sarkar, "Development of a Fuzzy logic based mobile robot for dynamic obstacle avoidance and goal acquisition in an unstructured environment," Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics., Monterey, CA, 2005, pp. 1198-1203, doi: 10.1109/AIM.2005.1511173.

[55]T. Fernando, H. Gammulle and C. Walgampaya, "Fuzzy logic based mobile robot target tracking in dynamic hostile environment," 2015 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), Shenzhen, 2015, pp. 1-6, doi: 10.1109/CIVEMSA.2015.7158609.

[56]T. Bresciani, “Modelling, Identification and Control of a Quadrotor Helicopter”, Master’s thesis, Lund University, Sweden, 2008.

[57]Y. Chai, L. Jia and Z. Zhang, "Mamdani Model Based Adaptive Neural Fuzzy Inference System and its Application in Traffic Level of Service Evaluation," 2009 Sixth International Conference

on Fuzzy Systems and Knowledge Discovery, Tianjin, China, 2009, pp. 555-559, doi: 10.1109/FSKD.2009.76.

[58] L. E. Buzogany, M. Pachter, and J. J. Azzo, "Automated control of aircraft in formation flight," in Proceedings of the AIAA Guidance, Navigation, and Control Conference, vol. 3, pp.1349–1370, 1993.

[59] T. D. Dong, X. H. Liao, R. Zhang, Z. Sun, and Y. D. Song, "Path tracking and obstacle avoidance of UAVs—fuzzy logic approach," in Proceedings of the 14th IEEE International Conference on Fuzzy Systems (FUZZ '05), pp. 43–48, North Carolina A&T State University, May 2005.

[60] Sensors in AirSim: Weblink: https://microsoft.github.io/AirSim/sensors.