University of Cincinnati

Date: 7/29/2011

I, Chelsea Sabo, hereby submit this original work as part of the requirements for the degree of Master of Science in Aerospace Engineering.

It is entitled:
UAV Two-Dimensional Path Planning In Real-Time Using Fuzzy Logic

Student's name: Chelsea Sabo

This work and its defense approved by:

Committee chair: Kelly Cohen, PhD
Committee member: Shaaban Abdallah, PhD
Committee member: Manish Kumar, PhD

UNIVERSITY OF CINCINNATI
UAV Two-Dimensional Path Planning

In Real-Time Using Fuzzy Logic

A thesis submitted to the
Graduate School
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of

Master of Science

in the Department of Aerospace Engineering
of the College of Engineering

by
Chelsea Sabo

B.S. University of Wisconsin - Madison

July 29, 2011

Committee Chair: Kelly Cohen, Ph.D.
ABSTRACT

There are a variety of scenarios in which the mission objectives rely on a UAV being capable of maneuvering in an environment containing obstacles in which there is little prior knowledge of the surroundings. In these situations, not only can these obstacles be dynamic, but sometimes there is no way to plan ahead of the mission to avoid them. Additionally, there are many situations in which it is desirable to send in an exploratory robot where the environment is dangerous/contaminated and there is a great deal of uncertainty. These scenarios could either be too risky to send people or not available to humans. With an appropriate dynamic motion planning algorithm in these situations, robots or UAVs would be able to maneuver in any unknown and/or dynamic environment towards a target in real-time. An autonomous system that can handle these varying conditions rapidly and efficiently without failure is imperative to the future of unmanned aerial vehicle (UAV).

This paper presents a methodology for two-dimensional path planning of a UAV using fuzzy logic. This approach is selected due to its ability to emulate human decision making and relative ease of implementation. The fuzzy inference system takes information in real time about obstacles (if within the agent’s sensing range) and target location and outputs a change in heading angle and speed. The FL controller was validated for both simple (polygon obstacles in a sparse space) and complex environments (i.e. non-polygon obstacles, symmetrical/concave obstacles, dense environments, etc). Additionally, Monte Carlo testing was completed to evaluate the performance of the control method. Not only was the path traversed by the UAV often the exact path computed using an optimal method, the low failure rate makes the Fuzzy Logic Controller (FLC) feasible for exploration. The FLC showed only a total of 3% failure rate, whereas an Artificial Potential Field (APF) solution, a commonly used intelligent control
method, had an average of 18% failure rate. Also, the APF method failed about 1/3 of the time for very dense environments (the FLC only had 5% failure rate). These results highlighted one of the advantages of the FLC method: its adaptability to additional rules while maintaining low control effort. Furthermore, the solutions showed superior results when compared to the APF solutions when compared to distance traversed. Overall, the FLC produced solutions that were on average only about 7.7% greater distance traveled (as opposed to 9.7% for the APF).
ACKNOWLEDGEMENTS

Author would like to thank Dr. Kelly Cohen, Manish Kumar & Dr. Shaaban Abdallah from the University of Cincinnati for all their guidance, support, and assistance.

Author would like to thank Steven Rasmussen at the Wright-Patterson AFRL for his support and the usage of his optimal solution methods and simulation for this research.

This work is dedicated to my mother, Dr. Sherry L. Sabo (1950-2010), who was unconditionally proud and supportive of her children. She was a role model on how to pursue a successful career and an inspiration to succeed in our own goals and dreams.
TABLE OF CONTENTS

ABSTRACT .................................................................................................................... ii

LIST OF FIGURES ........................................................................................................ viii

LIST OF TABLES ........................................................................................................... x

CHAPTER 1: INTRODUCTION ....................................................................................... 1
  1.1 Motivation............................................................................................................. 1
  1.2 Thesis Objective ............................................................................................... 4
  1.3 Problem Statement ......................................................................................... 5
  1.4 Assumptions .................................................................................................... 8

CHAPTER 2: LITERATURE REVIEW ............................................................................. 10
  2.1 Roadmap Methods .......................................................................................... 10
      2.1.1 Visibility Graphs ...................................................................................... 11
      2.1.2 Voronoi Diagrams .................................................................................. 12
  2.2 Cell Decomposition ....................................................................................... 13
  2.3 Potential Fields .............................................................................................. 14
  2.4 Fuzzy Logic .................................................................................................... 15
  2.5 Discussion ....................................................................................................... 19
  2.6 Contributions of the Work ........................................................................... 23

CHAPTER 3: METHODOLOGY ...................................................................................... 24
  3.1 Fuzzy Logic Controller ................................................................................... 24
      3.1.1 Input and Output Membership Functions ............................................ 25
      3.1.2 Decision-Making Rule Base ................................................................. 31
      3.1.3 Defuzzification ...................................................................................... 38
  3.2 Implementation Issues .................................................................................... 40
CHAPTER 4: DEVELOPMENT ................................................................. 44
  4.1 Testing Environment ........................................................................ 44
    4.1.1 Obstacle Representation .......................................................... 44
    4.1.2 Model Discretization ............................................................... 45
    4.1.3 UAV Operating Envelope ......................................................... 46
    4.1.4 Performance Metrics .............................................................. 47
    4.1.5 Monte Carlo Simulation Parameters ......................................... 48
  4.2 Potential Field Method .................................................................... 49
  4.3 Optimal Solution ............................................................................. 51

CHAPTER 5: RESULTS .............................................................................. 53
  5.1 Control Methodology Validation ..................................................... 53
    5.1.1 Constraint Verification ........................................................... 54
    5.1.2 Complex Environments .......................................................... 56
    5.1.3 Target Re-tasking ................................................................. 57
  5.2 Performance .................................................................................... 58

CHAPTER 6: CONCLUSIONS ................................................................. 65

CHAPTER 7: FUTURE WORK .................................................................... 68

REFERENCES .......................................................................................... 71
LIST OF FIGURES

Figure 1: UAV in Inertial Reference Frame .............................................................................. 5
Figure 2: Sensing Radius of Agent ................................................................................................. 7
Figure 3: Example of a Visibility Graph ......................................................................................... 11
Figure 4: Example of a Voronoi Diagram ...................................................................................... 12
Figure 5: Example of the Cell Decomposition Method ................................................................. 13
Figure 6: Example of the Potential Field Method ......................................................................... 14
Figure 7: Example of a FL Membership Function ......................................................................... 16
Figure 8: Example of a FL Input ..................................................................................................... 17
Figure 9: Example of a FL Input ..................................................................................................... 17
Figure 10: Example of a FL Output ................................................................................................. 18
Figure 11: FL Totaled Outputs from Evaluated Rules .................................................................... 19
Figure 12: Fuzzy Logic Control System ......................................................................................... 24
Figure 13: Input: Distance between Obstacle and Agent (Very Far is for an obstacle out of sensing range) ............................................................................................................... 26
Figure 14: Input: Angle between Obstacle and Agent Heading Angle ........................................... 27
Figure 15: Input: Distance between Target and Agent ................................................................. 27
Figure 16: Input: Angle between Target and Agent Heading Angle ............................................. 28
Figure 17: Output: Percentage of Maximum Velocity .................................................................. 29
Figure 18: Output: Heading Angle Deviation ............................................................................... 30
Figure 19: Process of Defining Rules and Tuning Membership Functions ..................................... 32
Figure 20: Situation in which no Obstacles are in Sensing Range and the UAV is far from its Target .......................................................... 33
Figure 21: Situation in which the UAV is close to its Target .......................................................... 33
Figure 22: Heading Angle Control Surface when Target is Far Away ......................................... 35
Figure 23: Velocity Control Surface when Target is Far Away ..................................................... 35
Figure 24: Situation in which Obstacles are in Sensing Range ...................................................... 36
Figure 25: Heading Angle Control Surface when Target is in Range ......................................... 37
Figure 26: Velocity Control Surface when Target is in Range ..................................................... 38
Figure 27: Example Output Membership Functions ................................................................. 38
Figure 28: Example Union of Output Membership Functions .................................................... 39
Figure 29: Example of the use of the Centroid Method to Calculate a Crisp Output ..................... 39
Figure 30: Agent Redefining a Large, Concave Obstacle ............................................................. 42
Figure 31: Agent Redefining Two, Symmetrical Obstacles ............................................................ 43
Figure 32: Example of Obstacle Environment within an Enclosed Grid ........................................ 45
Figure 33: Examples of the Fuzzy Logic Controller with Polygon Obstacles .............................. 53
Figure 34: Example of the Fuzzy Logic Controller for Constraint Verification ......................... 54
Figure 35: Acceleration versus Time of the Example for Constraint Verification ....................... 55
Figure 36: Heading Angle Rate versus Time of the Example for Constraint Verification ............ 55
Figure 37: Controlled Heading Angle and Actual Heading Angle versus Time of the Example .... 56
Figure 38: Controlled Velocity and Actual Velocity versus Time of the Example ....................... 56
Figure 39: Examples of the Fuzzy Logic Controller with Complex Obstacles ............................ 57
Figure 40: FLC with Mid-Flight Retasking .................................................................................. 58
Figure 41: Example of a FLC Solution (Left) versus an Optimal Control Solution (Right) .................................. 59
Figure 42: Example of a FLC Solution (Left) versus a Potential Field Solution (Right) ............................... 59
Figure 43: Extra Distance Traversed for the Approximate Solutions in Simple (Least Dense) Environments .............................. 61
Figure 44: Extra Distance Traversed for the Approximate Solutions in Moderate (Medium Density) Environments ........................................................................................................... 62
Figure 45: Extra Distance Traversed for the Approximate Solutions in Complex (Very Dense) Environments ........................................................................................................................ 62
Figure 46: Extra Time Travelled for the Approximate Solutions in Simple (Least Dense) Environments 63
Figure 47: Extra Time Travelled for the Approximate Solutions in Moderate (Medium Density) Environments ........................................................................................................................ 63
Figure 48: Extra Time Travelled for the Approximate Solutions in Complex (Very Dense) Environments ........................................................................................................................ 64
Figure 49: Mutli-Agent Fire-Fighting Scenario ........................................................................................................ 70
LIST OF TABLES

Table 1: Advantages and Disadvantages of Several, Common Path Planning Techniques ......................... 21
Table 2: Output Heading Angle Change for Varying Target Angles (Obstacle/s at 'Very Far') ............... 34
Table 3: Output Velocity Change for Varying Target Distances (Obstacle/s at 'Very Far') ....................... 34
Table 4: Output Heading Angle Change for an Obstacle within the Sensing Range ................................ 36
Table 5: Output Velocity Change for an Obstacle within the Sensing Range ........................................... 37
Table 6: Output Heading Angle Change for an Obstacle with Sensing Range (Obstacle Distance is Far & Obstacle Angle is Pos Big or Neg Big) ........................................................................... 37
Table 7: UAV & Simulation Parameters ..................................................................................................... 47
Table 8: Control Solution Parameters ...................................................................................................... 48
Table 9: PF Simulation Parameters ........................................................................................................... 51
Table 10: Performance Results – Percentage of Failures & Control Effort ............................................. 60
CHAPTER 1: INTRODUCTION

Unmanned Air Vehicles are being increasingly used for missions that are undesirable for humans [1]. From an operational perspective, it is imperative that these UAVs are able to conduct its mission with a certain level of autonomy. A fundamental ability of autonomous UAVs is motion/path planning. This requires that UAVs navigate to targets in the quickest and safest manner. Moreover, in real world settings, it is often the case that the information about the environment is incomplete, and the scenario itself evolves dynamically, or often, both. The subject of this thesis is dynamic motion planning of UAVs in an uncertain environment.

In this chapter, the motivation is developed and several possible applications described. Furthermore, the research objective is explained, problem statement is laid out, and several assumptions that were made are expressed.

1.1 Motivation

Increasingly over the years, autonomous robots and vehicles are being used to perform missions that are considered “dull, dirty, and dangerous” in both military and civil operations, such as operations in nuclear power plants, for the exploration of Mars, to investigate behind enemy lines in battle, wild-fire surveillance, border patrols, and weather forecasting [1] - [2]. A large amount of research has gone into the development of Unmanned Aerial Vehicles (UAVs) for use in military tasks. Military UAVs are currently being used to perform intelligence, surveillance, and reconnaissance (ISR) functions as they are often characterized as being dull, dirty, and/or dangerous. ISR systems are comprised of various means for acquiring and processing information needed by military commanders/national security decision makers. The spending on ISR systems alone, a major portion of U.S. intelligence spending, amounts to
approximately $40 billion annually [3]. In an asymmetric warfare setting, such as the conflicts in Iraq and Afghanistan, Unmanned Aerial Vehicles (UAVs) have clearly demonstrated a profound operational impact on ISR missions. In March 2010, the US military revealed that it would be completing one million flight hours for its UAS inventory in April 2010 which includes surpassed 700,000 flight hours for the MQ-1B Predator UAS [4]. Furthermore, Predator-series flight hours have seen tremendous growth in recent years, with annual totals increasing from 80,000 hours in 2006, to 130,000 hours in 2007, 235,000 hours in 2008, and 295,000 hours in 2009 [5]. This growing trend is a clear indicator for the paradigm shift which has taken place in recent years.

As the trend develops toward the increasing use of UAVs, it becomes necessary to allocate and control them effectively. Currently, Control Sciences of multiple platforms that are in use largely consists of Unmanned Air Vehicles (UAVs) working individually; that is, they hardly communicate at all between each other. They are also incapable of being rerouted or re-tasked in flight, which is crucial as the mission objectives change, threats evolve, environment changes, or where there is no prior information about the scenario. Due to these limitations, it has been identified that a main concern for UAV growth is autonomous and intelligent control [6]. This would help UAVs to act and react more like their manned counterparts (i.e. piloted aircraft).

Additionally, there are a variety of scenarios in which the mission objectives rely on a UAV being capable of maneuvering in a dynamic, changing environment or in an environment in which there is no prior knowledge of the surroundings. Many military applications depend on UAVs avoiding threats due to anti-aircraft missiles or enemy warplanes. In this situations, not
only are these obstacles dynamic, there is no way to plan ahead of the mission to avoid them due to uncertainty associated with the "Fog of War".

There are further situations in which it is desirable to send in an exploratory robot where the environment is unknown or there is incomplete information about the environment. Again, these scenarios could either be too risky to send people or not available to humans (i.e. Mars). For instance, the Mars exploration rover has several sites of interest, where water was suspected to exist and mineral deposits, and drives there to perform studies. In this case, the rover has little-to-no previous information about its environment and expensive (and irreparable) equipment that might not survive a crash [7]. At this point, we are not capable of sending humans to Mars, but we are capable of sending robots with limited sensing and maneuvering capabilities. With the appropriate path planning algorithms, the rover could explore, take video footage and samples, and relay data back to Earth.

An important civilian application concerns wildfires, which cause destruction of thousands of acres, millions of dollars in damage, and cost many people their lives [8]. Because forest fires can change rapidly depending on environmental conditions and become difficult to combat, people are trying to develop new ways to deal with these fires. UAV’s are being used more often in these situations, and it is expected that their numbers will continue to grow [2]. Presently, the Ikhana, a NASA funded civilian version of the Predator drone, is being used as a detection platform and for real-time data relay for wildfires with in California. It has onboard infrared sensors that can penetrate thick smoke and haze and it is capable of communicating information about size, intensity, and movement of fires over a long period of time. This acquired data is overlaid on Google Earth maps and then transmitted in real-time to firefighting commanders [9]. It these situations, it would reduce the burden on the incident commander to be
able to send a fire-fighting agent to a retardant drop site autonomously (without the need to program obstacles and run path planning algorithms). Real time decision-making would also be essential when combating wildfires, as situations can change rapidly due to fluctuating environmental conditions (wind, fuel, terrain, etc).

In all these scenarios, mission effectiveness is dependent upon, or could be improved by, these vehicles autonomously path planning in their environment. Because most effective path planning algorithms rely on complete prior knowledge of the environment, it severely limits the capabilities of UAVs. Furthermore, these algorithms cannot attain solutions for complex environments (these problems are known to be NP-hard), and fairly accurate, real-time solutions are the reasonable expectation. Therefore, a more sophisticated method to two-dimensional path planning is needed to not only allow for maneuvering in an unknown environment, but is necessary to move in a dynamic and changing environment in real-time, an important ability for realistic missions. To do this, UAVs must be able to motion plan in a dynamic sense.

1.2 Thesis Objective

The objective of this thesis to propose an algorithm for path planning in two-dimensions for an UAV that would allow it to operate in real-time and in an unknown, obstacle environment.

In the scenario studied in this research effort, an UAV uses basic heuristics to travel to a known target while avoiding various, static obstacles. A sensor system is used to provide feedback about the obstacles in the UAV’s local environment. This information is then processed at each sampling time and used as one of the factors to determine the motion of the UAV for obstacle avoidance and path planning to the target.
Given a starting point and a target location, a path planning strategy must take into account dynamic and kinematic constraints of the vehicle, as well as information about obstacles, to generate a path. Therefore, this work was done to:

1. Model an Unmanned Air Vehicle including kinematic and dynamic constraints

2. Develop a control strategy for motion planning: path planning to a target while using sensor information about the local environment for obstacle avoidance

3. Validate the control strategy in a MATLAB simulation

4. Analyze and interpret contributions and limitations of the methodology

1.3 Problem Statement

The problem of path planning in two-dimensional space is formulated by generating a path between a known initial state, \( O (x_o, y_o) \), and final state, \( T (x_t, y_t) \). Therefore, the kinematic equations for a UAV in this circumstance are a function of the inertial position \( (x,y) \), the cruise velocity \( (v) \), and the heading angle \( (\Theta) \) as shown in Figure 1.

![Figure 1: UAV in Inertial Reference Frame](image-url)
Second-order differential equations describing the aircraft autopilot system were developed by Buzogany [11]. Since we are concerned with two-dimensional path planning, the altitude response was dropped. Therefore, we are only interested in controlling the velocity and heading angle where \( v_c \) and \( \Theta_c \) are the control inputs, respectively. The motion of the UAV can be described as follows in Eqn. (1) [12] where \( \tau_v \) and \( \tau_\theta \) are the time delays associated with controlling the velocity and heading angle, respectively:

\[
\begin{align*}
\dot{v} &= \frac{1}{\tau_v} (v_c - v) \\
\dot{\theta} &= \frac{1}{\tau_\theta} (\theta_c - \theta)
\end{align*}
\] (1)

Further development by Dong [13] illustrates the UAV in the inertial reference frame (Figure 1), and the position of the UAV can be defined using the heading angle (\( \Theta \)) and distance from the origin (\( l \)):

\[
X = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} l \cos \theta \\ l \sin \theta \end{pmatrix}
\] (2)

It follows that

\[
\dot{X} = \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} v \cos \theta \\ v \sin \theta \end{pmatrix}
\] (3)

and that

\[
\ddot{X} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} \dot{v} \cos \theta - v \dot{\theta} \sin \theta \\ \dot{v} \sin \theta + v \dot{\theta} \cos \theta \end{pmatrix}
\] (4)

By combining Eqns. (1) and (4), the kinematic equations can be written in the form
\[
\dot{x} = \begin{pmatrix}
\frac{1}{\tau_v} (v_c - v) \cos \theta - v \frac{1}{\tau_\theta} (\theta_c - \theta) \sin \theta \\
\frac{1}{\tau_v} (v_c - v) \sin \theta + v \frac{1}{\tau_\theta} (\theta_c - \theta) \cos \theta
\end{pmatrix}
\]

(5)

where the control inputs are the velocity \((v_c)\) and heading angle \((\Theta_c)\).

Both the velocity and heading angle \([14]\) are constrained as follows as shown below in Eqn. (6) and (8), respectively. Additionally, the acceleration and heading angle rate is bounded to prevent instantaneous changes as shown below in Eqn. (7) and (9), respectively.

\[
v_{\text{min}} \leq v \leq v_{\text{max}}
\]

(6)

\[
-a_{\text{max}} \leq \dot{v} \leq a_{\text{max}}
\]

(7)

\[
-\theta_{\text{max}} \leq \theta \leq \theta_{\text{max}}
\]

(8)

\[
-\omega_{\text{max}} \leq \dot{\theta} \leq \omega_{\text{max}}
\]

(9)

In this problem setup, the agent has a sensing range that is considered able to sense obstacles within ±90° and within a certain sending radius as shown in Figure 2.

![Figure 2: Sensing Radius of Agent](image)
1.4 Assumptions

For the purpose of this investigation, several simplifying assumptions were made without taking away from the applicability of the developed approach. While the control methodologies developed in this research (see Chapter 3) can be extended to be used in three dimensions, it is assumed that all motion is in two dimensions; that is, the altitude is removed as a degree of freedom (altitude is constant). It is also assumed that wind is negligible (the UAV cruise speed is much greater than wind speed). Furthermore, the aircraft is a point mass, and therefore, no moment effects are considered. Additionally, the dynamic constraints of the aircraft are assumed to be known.

To isolate the performance of the control methodology, several assumptions were made on the capabilities of the sensor system. The agent is assumed to be able to detect obstacles within its sensing range in real-time and define them accordingly. The capabilities of sensors are discussed briefly by Mujumdar and Padhi [15], and while further work needs to be done in this area to extract information from the sensors accurately, this is a field of extreme interest for ongoing research.

The position of the UAV and the target location is assumed to be known by a GPS (Global Positioning System) and accurate. Furthermore, this information is updated at a reasonable sampling time. Additionally, the start location of the UAV and the target positions are given by some outside planner (known GPS locations), and there are no obstacles at these locations.
Finally in this work, it is assumed that a feasible solution exists. That is, 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 an obstacle.
CHAPTER 2: LITERATURE REVIEW

In this chapter, a review of literature related to the research objective is presented. Path-planning and motion planning algorithms encompass a variety of applications and needs. Many areas, such as networking, graph theory, combinatorial optimization, and data structuring have all contributed to the field. For the most part, path planning can be simplified to navigating a robot to a distance point safely i.e. avoiding collisions along the way. To make scenarios realistic, algorithms must be able to handle varying numbers of objectives, complex tasks, different constraints, and uncertainty [16]. In this research objective, the robot is a UAV (Unmanned Aerial Vehicle), and the point can be any number of tasks (basically involves visiting spatially different locations). It is necessary that the UAV be able to avoid obstacles and no fly zones, and in reality, these will always have some degree of uncertainty. Thus, an approach is needed that can provide the required navigational decision-making given the inherent uncertainty associated with this class of problems. In this problem statement, uncertainty occurs when the vehicle having has no prior knowledge of its environment, there are dynamic obstacles, and target itself is dynamic. While there is much research in the field of motion planning, much of it can be boiled down into several categories of approaches. Therefore, the focus of this survey is on roadmap methods like visibility graphs and Voronoi diagrams, cell decomposition, potential fields, and fuzzy logic, as it is a basis for the approach used here.

2.1 Roadmap Methods

Roadmap methods create a graph that connects an initial, starting point to a final, target location around obstacles within the environment. Roadmaps utilize the following definitions: the total space in the environment in called the C_space, the free space not containing any obstacle
is called C\textsubscript{free}, and the space containing the obstacles is called C\textsubscript{obstacle}. Once the environment is defined, a graph is then created that connects vertices in the C\textsubscript{free} enclosed within C\textsubscript{space} [17]. Finally, this map can then be searched for a path from start to finish, generating a feasible route using a brute force method or an algorithm like A* [18].

2.1.1 Visibility Graphs

Visibility graphs create a line-of-site map that is formed by first defining all obstacles as polygons (the dark solid polygons in Figure 3) [10]. Generally, in this method it is assumed that even non-polygonal obstacles can be enclosed within polygons. From these polygons, the set of vertices, V, and edges that connect these vertices, E, that comprise these obstacles make up a graph, G = (V, E). In Figure 3, this is indicated by the thin, solid lines. In this graph, the two vertices that make up the start and end point, and edges that connect them to the graph (the dashed lines in Figure 3), are included in G, but the edges that intersect are not included in E. That is, edges that intersect obstacles are not incorporated [10], [17]. Once the graph is created, a shortest route from the start point to the end point is calculated (the bold lines in Figure 3).

![Figure 3: Example of a Visibility Graph](image-url)
2.1.2 Voronoi Diagrams

Voronoi diagrams are similar to visibility graphs in that they also require polygon obstacles and create a set of vertices from which to construct the path. However, they create those vertices in the $C_{\text{free}}$ away from the obstacles. First, the boundary of the $C_{\text{space}}$ is enclosed by a rectangle and assigned as an obstacle. Then, points are calculated as the locus between two or more obstacle edges (including the boundaries of $C_{\text{space}}$). Once these vertices are determined, the points are connected into a graph along the edges of each obstacle. Finally, the initial and final points are incorporated finding the point along an edge of the diagram which increases the distance from the obstacle the fastest [10], [17]. Once this graph is formed, we can again use any number of algorithms to calculate the shortest path. An example of a visibility graph is shown in Figure 4 [10].

![Figure 4: Example of a Voronoi Diagram](image-url)
This approach tends to maximize the distance between the vehicle and the obstacles, which can be advantageous when there is any uncertainty to the obstacles. However while conservative, this method can be costly when trying to minimize the distance or time travelled.

2.2 Cell Decomposition

Both exact and approximate cell decomposition methods are common in motion planning literature. The methodology involves decomposing $C_{\text{free}}$ into cells and like roadmap methods, requires that all obstacles are polygons. Essentially, the $C_{\text{space}}$ is broken into cells (see Figure 5) and all cells that are adjacent to obstacles are considered off-limits [10]. All of the free cells are then used to try to construct a path from the initial to the final point. If no path is found, then the cells are then decomposed into smaller cells and a new search is completed.

![Figure 5: Example of the Cell Decomposition Method](image)

The benefits of this method are that it is practical to implement above two-dimensions and relatively quick to compute. However because it is an iterative process, it is not necessarily practical to compute online as there is no guarantee when, or if, a solution is found. Additionally while there are both exact and approximate cell decomposition methods, the approximate method (shown in the figure above) can provide very suboptimal solutions. For example, in Figure 5 it is
probably more efficient to take the path above the obstacle than the one indicated (below the obstacle).

### 2.3 Potential Fields

Potential fields for motion planning were originally used for on-line collision avoidance for when a UAV does not have prior knowledge of the obstacle but senses them in real-time. The relatively simple concept treats the vehicle as a point under the influence of an artificial potential field where the variations in the space represent the structure of the environment. The attractive potential reflects the pull of the vehicle to the goal and the repulsive potential reflects the push of the UAV from the obstacles [10]. Therefore, the environment is decomposed into a set of values where high values are associated with obstacles and low values are associated with the goal.

![Figure 6: Example of the Potential Field Method](image)

Several steps are used to construct the map using potential fields. First, the target point is assigned a large negative value and $C_{\text{free}}$ is assigned increasing values as the distance from the goal increases (Figure 6 on left). Again typically, the inverse of the distance from the goal is
used as a value [10]. Second, $C_{\text{obstacle}}$ is assigned as the highest values and $C_{\text{free}}$ is assigned decreasing values as the distance from the obstacles increases. Typically, the inverse of the distance from the obstacle is used as a value. Finally, the two potentials in $C_{\text{free}}$ are added and a steepest descent approach is used to find an appropriate path from start point to the end point (see Figure 6 on the right) [19].

2.4 Fuzzy Logic

Fuzzy logic, based on multi-valued logic, provides a unique method for encoding knowledge about continuous variables. Even though most of the information we process every day is fuzzy, machines are programmed to only handle crisp or binary numbers. Therefore, fuzzy logic was proposed by Lofti A. Zadeh [20] in 1965 as a way to more accurately capture the real world. Experience from the past decade, with the successful marketing of a wide variety of products based on the Fuzzy Logic, has shown that for certain applications this approach can lead to lower development costs, superior features, and better end product performance. One of the inherent properties of fuzzy logic systems is that it has the capability of being a universal approximator. Additionally, this system has the ability to utilize expert heuristic knowledge of operation of controlled systems including physical intuition; capacity to successfully handle uncertainties and nonlinearities; and the existence of a variety of tools that assist in studying and building efficient fuzzy systems in relatively short times. In recent times, the advantages of fuzzy logic systems have made them attractive candidates for use in expert systems. Due to these advantages, fuzzy logic excels in circumstances where the environment is continually changing and incomplete information is available.

Fuzzy logic controllers map crisp inputs to crisp outputs through three main phases: fuzzification, inference (fuzzy rules), and defuzzification. Fuzzification converts crisp numbers
to fuzzy membership functions, the inference system relates inputs to outputs via “if-then” rules, and the defuzzification converts the totaled outputs from the evaluated rules back to a crisp value. This is done as follows:

A membership function is a function that defines the degree of membership of a value, X, of its entire domain. The value of the function, or the degree of membership, always lies between the interval [0,1] where 0 would equate with no membership and 1 would be complete membership. This is in contrast to typical, crisp values that can either be only 0 or 1. Furthermore, the domain of a value is typically broken into several membership functions that correspond to the ranges over which the input can be divided. For example, see Figure 7 [21]. The value here would be the voltage and the x-axis corresponds to the range over which the values of the voltage could take. Only one membership function is shown in this diagram: low voltage. The membership function shows the degree of membership for low voltage over a subset of the domain. If a crisp input (1.5 in this case) was to be read (shown below), the degree of membership could be calculated (0.5 in this case).

![Figure 7: Example of a FL Membership Function](image-url)
An entire value defined over its possible domain with multiple membership functions could look as follows in Figure 8, Figure 9, or Figure 10 [21]. For a fuzzy logic control system, the values are both the inputs and outputs of the system. Therefore, each input and output is broken down into a set of membership functions.

Figure 8: Example of a FL Input

Figure 9: Example of a FL Input
Once the membership functions for each input and output is defined, the rules are established to relate them. These rules relate individual membership functions and take the form of *If-Then* rules. If the height and vertical velocity, in Figure 8 and Figure 9 respectively, are used as inputs, and the control force in Figure 10 is used as an output, an example for a landing control problem [21] would look as follows:

**IF** the height is *large* and the vertical velocity is *zero*, **THEN** the control force is *down large*.

Once a set of rules is created, they can be evaluated based on the crisp inputs given. If the value of the crisp input falls within the domain of the membership function, then the rules involving those membership functions “fire” and are evaluated. The degree to which a rule fires corresponds to the degree of membership of the crisp value in the membership function. Finally, all the outputs from the evaluated rules are totaled and defuzzified (Figure 11). Defuzzification can be done in various ways. Several commonly used ones are: max membership output, where the output is found as the corresponding value of the maximum membership value (in Figure 11).
the value would be 0); centroid method (in Figure 11 the value would be 5.8); and weighted average method (the mean of each output membership function is used to calculate an overall average) [21].

![Image of FL Totaled Outputs from Evaluated Rules]

**Figure 11: FL Totaled Outputs from Evaluated Rules**

### 2.5 Discussion

Because path planning and motion planning algorithms have been studied for many years, excellent strategies have been developed for path planning in obstacle environments. Those methods that have been studied and utilized most tend to be visibility graphs, Voronoi diagrams, and cell decomposition. While they each have their advantages, they also have downfalls (see Table 1). The biggest benefit to using visibility graphs is that they are optimal. This is due to the fact that all paths are formed along the edges of obstacles. However, this is one of the disadvantages of this method also. If there is any uncertainty to the obstacles size or shape, the path can easily become infeasible. Furthermore, once extended to the three-dimensional case, these solutions no longer guarantee optimality. On the other hand, Voronoi diagrams are able to handle a little uncertainty well, because they create paths that inherently
steer clear of the obstacles. However, they are not optimal and can often create paths that intuitively do not make much sense. Also like visibility graphs, it can be very difficult to implement this method above 2-D. Finally, both exact and approximate cell decomposition methods have been studied. Again, the downfalls to these methods require complete prior information and all obstacles to be polygons. While this is more practical to implement above two-dimensions than roadmap methods, it can be difficult for very complex environments. Furthermore, if the problem is not solvable, this method will not recognize that [10]. For example, if all possible paths are blocked by obstacles, this algorithm will iteratively search for a solution (creating smaller cells and searching for a path) indefinitely.

Much focus has also been on intelligent strategies like potential fields [22] - [25] as they have shown excellent results for comparable situations. While this method cannot guarantee optimality, it has shown to produce comparable results for good algorithms. Furthermore, it is excellent in the case of uncertainty and easy to extend to 3D. Also, this method is quick to compute and therefore, practical to do online. All of these benefits, and the fact that optimal solutions are computationally intractable for complex scenarios, confirm intelligent control methods as the practical solution to these problems. However, one of the major disadvantages to this method is the tendency to get caught in local minima, causing them to fail. This is due to the fact that the methodology basically acts as a fastest descent approach. For example, this occurs when a vehicle encounters a C-shaped obstacle. While research has been done to work around this problem, this singular problem is the main focus creating a path planning algorithm based on the artificial potential field approach [19]. For example, different techniques try inserting intermediate nodes around the obstacles to help the vehicle navigate out.
In addition to artificial potential fields (APFs) as an intelligent control method, Fuzzy logic has shown excellent results for path tracking and collision avoidance for both mobile robots and UAVs [13], [26] – [34]. Additionally, fuzzy logic has similar benefits to APFs (i.e. adaptability to uncertainty, near-optimal results, ability to compute online, etc), as well as the flexibility to work around some of the issues APF methods have with local minima. Path tracking for mobile robots show high-quality solutions [27] - [31] using the path error as inputs and the torque of wheels as outputs. Additionally, there has been some work that uses the benefits of fuzzy logic combined with other intelligent control techniques like genetic algorithms, potential fields, and neural networks for path planning of robots [26], [33] - [34]. When introducing stationary obstacles, and exploiting information about these obstacles (i.e. distance and angle) path tracking using fuzzy logic for UAVs has showed promising results [13].

Table 1: Advantages and Disadvantages of Several, Common Path Planning Techniques

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>XX</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complex Obstacles</td>
<td></td>
<td></td>
<td>XX</td>
<td>X</td>
<td>XX</td>
</tr>
<tr>
<td>Higher Dimensions</td>
<td></td>
<td></td>
<td>XX</td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>Computable Online</td>
<td></td>
<td></td>
<td></td>
<td>XX</td>
<td>XX</td>
</tr>
<tr>
<td>Uncertainty</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>XX</td>
</tr>
<tr>
<td>Finds Impossibilities</td>
<td>XX</td>
<td>XX</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ease to Implement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>XX</td>
</tr>
</tbody>
</table>
Motion planning in real-time is becoming an increasingly studied field as mobile robots and Unmanned Aerial Vehicles become more autonomous to keep up with growing demands [15], [35] - [36]. However, many of the optimal strategies require complete information of the environment apriori [10], [37] - [38] and fail in the presence of uncertainty, both requirements of autonomous vehicles expecting to operate in realistic environments. For example, Zelek [25] has shown that a robot can use potential fields to navigate around an unknown environment in real-time successfully. Furthermore, methods like rapidly-exploring random trees and probabilistic roadmap methods [15], [39] have shown successful motion planning in real-time. These methods are sample-based and create random nodes toward the general direction of the goal vertex. One downfall to most of these real-time planning techniques is they require the user to recompute a solution when new information is presented [38], [40]. Often this can be computationally difficult and not very realistic to do online (especially in complex environments).

While the literature shows the advantages of using fuzzy logic for path tracking and planning for robots and path tracking for UAVs, to the best of the author’s knowledge it does not address motion planning for UAVs in real-time.
2.6 Contributions of the Work

Similar strategies used for path tracking, motion planning, and obstacle avoidance of mobile robots and UAVS using fuzzy logic are exploited here in this research. While there has been extensive work done for mobile robots, the applications to UAVs has been limited. Therefore, this work extends previous research on path tracking with obstacle avoidance and is motivated by the fact that these methodologies have shown excellent results in the past. This strategy takes information about obstacles and a target location to motion plan in real-time. The contributions of this work can be summarized as follows:

- The solution takes UAV dynamics into account.
- It is a robust system in that it makes no assumptions about the environment apriori. Therefore, it works well with limited or no information.
- The system isn’t just reactionary to obstacles: it continues to path plan towards the target while avoiding obstacles. This allows for better solutions and makes it less likely to get caught in local minima.
- No assumptions are made about the target location: it can be static or dynamic.
- Fuzzy Inference Systems (FISs) are fast and can be implemented online with limited onboard processor capability.
- The fuzzy system has a very low failure rate.
- The fuzzy system yields near optimal performance in real-time!
CHAPTER 3: METHODOLOGY

Due to the advantages that fuzzy logic presents, as discussed in section 2.4, in circumstances where the environment is continually changing and incomplete information is available, it is the basis behind the control strategy presented here for motion planning. In this chapter, the overall control system is described in detail, including control inputs and outputs, the decision-making rule base, and defuzzification calculation, and the methodology used for practical implementation issues.

3.1 Fuzzy Logic Controller

Like previously discussed, fuzzy logic manipulates inputs using heuristic knowledge and typically converts them to outputs in three stages: fuzzification, decision-making logic, and defuzzification. The fuzzy logic controller can be visualized in Figure 12.

---

**Figure 12: Fuzzy Logic Control System**
Fuzzification takes analog inputs and converts them to a continuous value between 0 and 1 based on their degree of membership in each function. A knowledge base is then used to form a set of rules relating the inputs and outputs in the form of if-then statements. The outputs are finally converted back to crisp numbers using a defuzzification method.

3.1.1 Input and Output Membership Functions

For this problem setup, four inputs are used for the fuzzification interface and two outputs are given after defuzzification. Inputs into the system are as follows: 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 obstacle inputs were used by Dong [13] for fuzzy path tracking and showed good results for obstacle avoidance. The target inputs were chosen based on the assumption that there is only minimal amount of information about the target; i.e. GPS coordinates. The outputs for the system were also used by Dong [13] and based on much literature that uses these as the control inputs to a UAV system.

The distance to the obstacle (Figure 13) is described by four membership functions:

Close

Medium

Far

Very Far (Out of Sensing Range)
Figure 13: Input: Distance between Obstacle and Agent (Very Far is for an obstacle out of sensing range)

The angle between the obstacle and the UAV (Figure 14) is described by six membership functions:

NB: Negative Big

NM: Negative Medium

NS: Negative Small

PS: Positive Small

PM: Positive Medium

PB: Positive Big
Figure 14: Input: Angle between Obstacle and Agent Heading Angle

Similar to the obstacle distance, the distance to the target (Figure 15) is described by three membership functions:

On Top

Medium

Far

Figure 15: Input: Distance between Target and Agent
Finally, the error between the heading angle and the target angle (Figure 16) is described by seven membership functions:

- NB: Negative Big
- NM: Negative Medium
- NS: Negative Small
- Z: Zero
- PS: Positive Small
- PM: Positive Medium
- PB: Positive Big

With these inputs and a rule base, the control input for the system is obtained. That is, the outputs of the fuzzy inference system, the percent of the maximum velocity and the heading angle change, are used as the control into the system. Therefore, the outputs of the FIS are the
percent of maximum velocity and the heading angle change. The Mamdani method is used here, which expects the output membership functions to be fuzzy sets (as opposed to crisp values).

The output velocity (Figure 17) can be broken into four membership functions:

VS: Very Slow
S: Slow
F: Fast
VF: Very Fast

![Velocity Graph]

Figure 17: Output: Percentage of Maximum Velocity
The output angle change (Figure 18) is parallel to the target angle. Therefore, there are seven membership functions described by:

- **NB**: Negative Big
- **NM**: Negative Medium
- **NS**: Negative Small
- **Z**: Zero
- **PS**: Positive Small
- **PM**: Positive Medium
- **PB**: Positive Big

![Heading Angle](image)

*Figure 18: Output: Heading Angle Deviation*
3.1.2 Decision-Making Rule Base

Rules relating the inputs and outputs for the fuzzy logic controller are set up in the form of if-then statements and are based on heuristics and human experience. The rules for the fuzzy inference system can be summed up in some simple decision making logic. There are a total of 40 rules for this setup, and the rules can be broken up into two situations: if there is an obstacle within the sensing range or not. If there is no obstacle detected, the distance to the obstacles is set to “very far” away (much larger than the sensing radius). This was done so that motion planning to the target is the priority when obstacles are not in range, and that avoiding obstacles is the priority when obstacles are in range. Additionally, since it is assumes that the UAV only has local information (information that is within the sensing range), the only data it can make decisions on when the sensing range is clear is about the target. The process in which the rules were developed and how the membership functions were tuned can be described by Figure 19 [41]. Rules were formed until reasonable solutions were found, and then the membership functions were tuned until the quality of the solutions reached a plateau.
The main objective of the controller when there is no obstacles within its sensing range becomes to plan a direct path to the agent’s primary target. This is done by altering the UAV’s heading angle to match that of the angle of the target in the inertial reference frame (Figure 20) or by driving the error between the two angles to zero (Eqn. (10)). For the cases in which there are no obstacles in range and the target is very far away, the UAV tends towards its maximum operating speed (Eqn. (10)).
Figure 20: Situation in which no Obstacles are in Sensing Range and the UAV is far from its Target

\[ e = |\theta - \varphi| \quad e \to 0 \quad v \to v_{\text{max}} \quad (10) \]

When the agent reaches its target location (Figure 21), the agent alters its velocity to slow down and complete mission objectives while still driving the error to zero (Eqn. (11)).

Figure 21: Situation in which the UAV is close to its Target

\[ e = |\theta - \varphi| \quad e \to 0 \quad v \to v_{\text{min}} + a \cdot v_{\text{max}} \quad a = \text{const.} \quad (11) \]
The rules implemented with the FIS can be summarized in Table 2 and Table 3 below. Furthermore, the rules can be read as follows (example from Table 2 and Table 3):

**IF** the angle between the target and agent heading angle is *negative big*, **THEN** the heading angle change is *negative big*.

**Table 2: Output Heading Angle Change for Varying Target Angles (Obstacle/s at 'Very Far')**

<table>
<thead>
<tr>
<th>Angle between Target and Agent Heading Angle</th>
<th>Output Angle:</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>NM</td>
</tr>
<tr>
<td>NB</td>
<td>NM</td>
</tr>
</tbody>
</table>

**Table 3: Output Velocity Change for Varying Target Distances (Obstacle/s at 'Very Far')**

<table>
<thead>
<tr>
<th>Distance between Target and Agent Location</th>
<th>Output Velocity:</th>
</tr>
</thead>
<tbody>
<tr>
<td>On Top</td>
<td>Medium</td>
</tr>
<tr>
<td>VS</td>
<td>S</td>
</tr>
</tbody>
</table>

The output control surface for the heading angle change (Figure 22) and the velocity (Figure 23) are shown below.
When obstacles are detected within the sensing range (Figure 24), the agent alters its velocity and heading angle using information about obstacle distance, obstacle angle, and target location to avoid, and then recover, from the obstruction. The agent must slow down and change course to avoid it. This involves driving the heading angle error to around 90° (Eqn. (12)).
Figure 24: Situation in which Obstacles are in Sensing Range

\[ e = |\theta - \beta| \quad e \to 90^\circ \quad v \to v_{min} + a \cdot v_{max} \quad a = \text{const}. \quad (12) \]

Once it is clear of this obstacle it can continue its path toward the target. The set of rules that describe the change in heading speed and angle when an obstacle is detected can be summed in Table 4, Table 5, and Table 6. The very last set is for when obstacles are at extreme angles and pose no threat of collision (indicated by the ** in Table 4). In these conditions, the rules set in Table 6 are followed. That is, the UAV can head toward the target at a reduced speed.

**Table 4: Output Heading Angle Change for an Obstacle within the Sensing Range**

<table>
<thead>
<tr>
<th>Obstacle Distance</th>
<th>PB</th>
<th>PM</th>
<th>PS</th>
<th>NS</th>
<th>NM</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far</td>
<td>**</td>
<td>Z</td>
<td>NS</td>
<td>PS</td>
<td>Z</td>
<td>**</td>
</tr>
<tr>
<td>Medium</td>
<td>Z</td>
<td>NS</td>
<td>NM</td>
<td>PM</td>
<td>PS</td>
<td>Z</td>
</tr>
<tr>
<td>Close</td>
<td>NS</td>
<td>NM</td>
<td>NB</td>
<td>PB</td>
<td>PM</td>
<td>PS</td>
</tr>
</tbody>
</table>
Table 5: Output Velocity Change for an Obstacle within the Sensing Range

<table>
<thead>
<tr>
<th>Obstacle Distance</th>
<th>PB</th>
<th>PM</th>
<th>PS</th>
<th>NS</th>
<th>NM</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Far</td>
<td>F**</td>
<td>VF</td>
<td>F</td>
<td>F</td>
<td>VF</td>
<td>F**</td>
</tr>
<tr>
<td>Medium</td>
<td>F</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>F</td>
</tr>
<tr>
<td>Close</td>
<td>S</td>
<td>VS</td>
<td>VS</td>
<td>VS</td>
<td>VS</td>
<td>S</td>
</tr>
</tbody>
</table>

Table 6: Output Heading Angle Change for an Obstacle with Sensing Range
(Obstacle Distance is Far & Obstacle Angle is Pos Big or Neg Big)

<table>
<thead>
<tr>
<th>Output Angle:</th>
<th>NB</th>
<th>NM</th>
<th>NS</th>
<th>Z</th>
<th>PS</th>
<th>PM</th>
<th>PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>NB</td>
<td>NM</td>
<td>NS</td>
<td>Z</td>
<td>PS</td>
<td>PM</td>
<td>PB</td>
</tr>
</tbody>
</table>

The output control surface for the heading angle change (Figure 25) and the velocity (Figure 26) are shown below.

Figure 25: Heading Angle Control Surface when Target is in Range
3.1.3 Defuzzification

The defuzzification stage takes the outputs from the if-then rule base and converts it to a crisp number. Because the Mamdami method is used here [21], outputs are also in the form of membership functions. As each rule is evaluated, the output membership functions are evaluated based on the degree to which the rule fired. Figure 27 shows two examples of output membership functions being evaluated at different degrees [30].
The total output area becomes the union of all the outputs from the corresponding rules that fired (see Figure 28) [30].

![Figure 28: Example Union of Output Membership Functions](image)

Once the total area from all the rules is summed, an appropriate method for determining a crisp output is chosen. Here, the centroid method, a common technique for defuzzification, is used. In this method, the geometric center of the area is computed (Figure 29) according to Eqn. (13) to find a crisp output, $z^*$. 

![Figure 29: Example of the use of the Centroid Method to Calculate a Crisp Output](image)
3.2 Implementation Issues

One of the main disadvantages to an intelligent control method like artificial potential fields for path planning is the tendency to get caught in local minima. For fuzzy logic, difficulty can occur at symmetrical or large, concave obstacle. However, the advantage to using fuzzy logic is that it is adaptable to incorporating additional rules to circumvent these issues. In this circumstance, the FIS uses only one point as the input to calculate the distance to the obstacle and always uses the closest point to the obstacle. Due to this, additional logic must be included.

To avoid this issue, a simple logic was implemented and verified to redefine obstacles that come within the agent’s sensing range. If the agent appropriately evaluates the obstacle within the sensing range, it can overcome these issues. That is, if the agent encloses the areas within a certain minimum, “safe” radius (i.e. “redefines” its environment and what it sees as an obstacle), it can navigate out of the situations that cause these minima.

The mathematical formulation is as follows: If the $\mathbb{R}^2$ global map is discretized, $O = \{O_1, O_2, \ldots, O_n\} \subset \mathbb{R}^2$ is the set of polygon (both concave and convex) obstacles where $E_O$ makes up the obstacle edges. In the discretized space, the $E_O$ obstacle edges can be described by a set of points $P = \{p_1, p_2, \ldots, p_m\}$ that make up the edges.

Let $(x_V, y_V)$ denote the position of the UAV at a given time. Then, the set of points that are within the sensing range, $r_s$, of the UAV is described by $P_I = \{p_{i1}, p_{i2}, \ldots, p_{il}\}$ where $P_I \subset P$ for which the following is true:

\[
z^* = \frac{\int \mu_c(z) \cdot z \cdot dz}{\int \mu_c(z) \cdot dz}
\] (13)
\[ \sqrt{(p_{\text{hi},x} - x_v)^2 + (p_{\text{hi},y} - y_v)^2} \leq r_s \]  

(14)

For this set of obstacle points within the UAV’s sensing range, the agent then redefines the space as follows:

For 1 to \( l \) where \( i, j \in l \) and \( i \neq j \), if

\[ \sqrt{(p_{\text{li},x} - p_{\text{lj},x})^2 + (p_{\text{li},y} - p_{\text{lj},y})^2} \leq r_f \]  

(15)

where \( r_f \) is the “safe” distance from the UAV to an obstacle, then \( P_N = \{p_{N1}, p_{N2}, \ldots p_{Nk}\} \) becomes a set of points that creates a new obstacle edge between points \( p_i \) and \( p_j \) where the set of points \( P_N \) is defined by Equation (16).

\[
m = \frac{(p_{\text{li},y} - p_{\text{lj},y})}{(p_{\text{li},x} - p_{\text{lj},x})} \quad b = p_{\text{li},y} - p_{\text{li},x} \cdot m \]

\[ y_k = m \cdot x_k + b \quad \text{for } x_k = p_{\text{li},x} \text{ to } p_{\text{lj},x} \]

(16)

While this is relatively simple logic, it holds under several assumptions and conditions. This holds as long as the minimum “safe” radius is large enough in comparison to the vehicle minimum turning radius (often a valid assumption). Therefore, if the sensing range is large, the vehicle will have enough time to turn to avoid the obstacle.

This logic holds for not only concave obstacles, but also for convex obstacles. If a concave obstacle is too small for the UAV to turn out of, the algorithm would define the whole area as an obstacle. This would make sense since the UAV would not be able to navigate in and out of the obstacle, and therefore, the UAV never will make its way into the obstacle in the first
place. For convex obstacles, the agent would just be redefining an area inside the obstacle (an area that the agent would already be unable to reach).

If the concave obstacle is large enough for the UAV to venture into, the UAV will be able to redefine the obstacle and navigate out, even with a minimum turn radius. This is shown below in Figure 30, where the blue is the sensing range and red is the obstacle (both original and redefined as it carries on).

![Figure 30: Agent Redefining a Large, Concave Obstacle](image)

For symmetrical obstacles, the UAV redefines as the area between the two as an “obstacle.” Therefore as the UAV approaches the obstacle and it comes into view, if multiple pieces (or two different obstacles here) appear in the sensing range and the distance between them is less than a safe distance, the UAV defines the entire distance between them as an obstacle. This implies that the area is “unsafe” to travel through, and the UAV would navigate around the two obstacles as though they were one. Several steps that the agent would take in this scenario are shown below in Figure 31.
Figure 31: Agent Redefining Two, Symmetrical Obstacles
CHAPTER 4: DEVELOPMENT

Seeing as potential fields have been used frequently as an intelligent control method for motion planning, it the basis for comparison for methodology developed here. Furthermore, the fuzzy logic controller is compared to an optimal path planning algorithm that uses visibility graphs. This was done to analyze the overall performance of the FLC. In this chapter, the algorithms based on potential fields and visibility paths are described, as well as the testing environment used in the Monte Carlo runs. That is, the representation of obstacles, the discretized model, and the parameters for the runs are described.

4.1 Testing Environment

The simulations described in this chapter and presented in the next were implemented in MATLAB programming language and on a Windows machine with an Intel Core i5 2.4 GHz processor and 4.0 GB of RAM.

4.1.1 Obstacle Representation

The testing environment is an enclosed grid where each vertex is identical and indicates the presence of an obstacle (see Figure 32 below). If there is an obstacle at that vertex, it is marked as 1. If not, it is marked as 0. Therefore, all obstacles are constructed as a series of vertices. The vehicle is assumed to be able to sense and define obstacle appropriately. That is, the vehicle searches its sensing range at each time step for obstacles and if it comes upon a vertex defined as a 1, it knows it is an obstacle.
While this is a simple representation of an environment, it has several useful advantages that are exploited. First, the FLC utilizes the closest point of the obstacle within the sensing range, and this is much easier to compute if the environment is discretized. Furthermore, the volume of the space that is composed of obstacles can help define the complexity of the environment.

Both polygonal and non-polygonal obstacles are tested in this researched, but the non-polygonal obstacles are only used for validation of the methodology. This is due to the fact that the optimal approach used for comparison is capable of handling only polygons.

4.1.2 Model Discretization

In addition to discretizing the environment, the UAV dynamics described in Section 1.3 were discretized. The sampling rate, T, is the time that the UAV position, UAV heading angle, UAV velocity, UAV sensing range, and target location were updated. Furthermore, a step in the
simulation time is represented by \( k = 0, 1, 2 \ldots \), each cell in the environment is equal to \( 1m \), and area of each map is \( 3.5km \times 3.5km \). Therefore, the model is updated according to the following:

\[
x[(k + 1)T] = x(kT) + T v(kT) \cos \theta(kT) \tag{17}
\]

\[
y[(k + 1)T] = y(kT) + T v(kT) \sin \theta(kT) \tag{18}
\]

\[
\theta[(k + 1)T] = \theta(kT) + T \omega(kT) \tag{19}
\]

\[
v[(k + 1)T] = v(kT) + \frac{T}{\tau_v} \left[ v_c(kT) - v(kT) \right] \tag{20}
\]

\[
\omega[(k + 1)T] = \omega(kT) + \frac{T}{\tau_\theta} \left[ \theta_c(kT) - \theta(kT) \right] \tag{21}
\]

Where the difference between the control input and the actual need to satisfy the constraints for acceleration and angular acceleration (Eqn. (22) and Eqn. (23)):

\[
|v_c(kT) - v(kT)| \leq a_{max} \tag{22}
\]

\[
|\theta_c(kT) - \theta(kT)| \leq \omega_{max} \tag{23}
\]

### 4.1.3 UAV Operating Envelope

In the simulations described here, the parameters of the UAV maneuverability are summarized below in Table 7. These parameters are described in the previous section and in Section 1.3 as the constraints of the UAV kinematics [12], [37]. In the table below, \( r_s \) is the sensing range of the UAV, \( r_f \) is the safe range for a UAV from an obstacle, and \( T \) is the sampling rate of the UAV, target, and obstacle information.
### Table 7: UAV & Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_s$</td>
<td>50 m</td>
</tr>
<tr>
<td>$r_f$</td>
<td>25 m</td>
</tr>
<tr>
<td>$v_{max}$</td>
<td>10 m/s</td>
</tr>
<tr>
<td>$v_{min}$</td>
<td>3 m/s</td>
</tr>
<tr>
<td>$a_{max}$</td>
<td>3 m/s$^2$</td>
</tr>
<tr>
<td>$\theta_{max}$</td>
<td>30°</td>
</tr>
<tr>
<td>$\omega_{max}$</td>
<td>25 rpm or 2.618 °/s</td>
</tr>
<tr>
<td>$\tau_{\theta}$</td>
<td>0.4 secs</td>
</tr>
<tr>
<td>$\tau_{\psi}$</td>
<td>4 secs</td>
</tr>
<tr>
<td>$T$</td>
<td>0.05 secs</td>
</tr>
<tr>
<td>$T_{exec}$</td>
<td>Time step, k, that last target is service</td>
</tr>
</tbody>
</table>

#### 4.1.4 Performance Metrics

Monte Carlo experiments were completed to analyze the performance of the different methods and gain insight to the contributions and limitations of the control method presented here. To do so, various performance metrics were calculated for each run. The performance error with respect to the optimal solution was calculated for both distance and time (though the fuzzy logic controller was tuned to “optimize” distance and not time). Also, the control effort was computed as the sum of the square of the heading angle change at each time step. The heading angle rate was used to determine a quantitative metric for the required path changes of each control method. Finally, the number of errors was recorded over all the simulations to determine failure rate. A failure was considered if the UAV never reached the target and/or the UAV “crashed” into an obstacle. For each of the three control methodologies implemented, the
previous was recorded where fuzzy logic control has subscript \( flc \), potential field method has subscript \( pf \), and optimal has subscript \( o \). These values and their calculation are shown below in Table 8.

Table 8: Control Solution Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Traversed</td>
<td>( D )</td>
<td>( \sum_{k=0}^{T_{\text{exec}}} \sqrt{(x[k+1]-x[k])^2 + (y[k+1]-y[k])^2} )</td>
</tr>
<tr>
<td>Time Traveled</td>
<td>( T )</td>
<td>( \sum_{k=0}^{T_{\text{exec}}} \sqrt{(x[k+1]-x[k])^2 + (y[k+1]-y[k])^2} ) ( v[k] )</td>
</tr>
<tr>
<td>Control Effort</td>
<td>( C )</td>
<td>( \frac{1}{T_{\text{exec}}} \sum_{t=0}^{T_{\text{exec}}} \omega(k)^2 )</td>
</tr>
<tr>
<td>Failure Rate</td>
<td>( F )</td>
<td># of failures/300</td>
</tr>
<tr>
<td>Obstacle Area</td>
<td>( OA )</td>
<td>Area of Obstacles (( \text{km}^2 ))/12.25 (( \text{km}^2 ))</td>
</tr>
<tr>
<td>Optimality</td>
<td>( J )</td>
<td>( \frac{D_{pf}}{D_o} ), ( \frac{D_{flc}}{D_o} ), ( \frac{T_{pf}}{T_o} ), ( \frac{T_{flc}}{T_o} )</td>
</tr>
</tbody>
</table>

### 4.1.5 Monte Carlo Simulation Parameters

Again, Monte Carlo testing was completed to evaluate the performance of the solutions obtained. Each simulation contained a fixed number of polygon obstacles, a fixed starting position for the UAV, and a fixed number of targets. There were three types of obstacles total, and the size and shape of each of the three were fixed. In each simulation, the obstacle type (so it could be any one of the three types), the location of the target/s, the location of the obstacles were all random. Three categories of obstacle density were used for comparing each method, and 300 runs of each of the three categories of obstacle density were used (900 simulations...
total). Simple (least dense) cases had 5 obstacles and an obstacle area percentage of less than 30%, moderate (medium density) had 10 obstacles and an obstacle area of less than 50% and greater than 30%, and complex (very dense) had 15 obstacles and an obstacle area of greater than 50%. The reason that the obstacle area was not consistent throughout the simulations is that the random aspect of the location of the obstacles allowed them to overlap.

4.2 Potential Field Method

As potential fields are widely used and the leading intelligent control method for dynamic path planning, it is the technique used for comparison here. Potential fields have been shown to provide near-optimal solutions and a viable alternative to optimal control strategies. The method is completed as follows:

Initially, the attractive potential and the repulsive potential are defined according to Eqn. (24) and Eqn. (25), respectively, and the overall potential field is defined according to Eqn. (26) where \( d(X, \cdot) \) is the distance between the UAV location (X) and the goal or the obstacle. The constants \( \alpha \) and \( \beta \) are scaling forces for the potentials, respectively. The distance \( d_0 \) is used to described the safe distance from an obstacle (or the distance in which the obstacle repulsive force no longer has an influence on the vehicle).

\[
U_{\text{Target}} = \frac{1}{2} \alpha \cdot d(X, \text{goal})^2 \\
U_{\text{Obst}} = \begin{cases} 
\frac{1}{2} \beta \cdot \left[ \frac{1}{d(X, \text{obst})} - \frac{1}{d_0} \right]^2 & d(X, \text{obst}) \leq d_0 \\
0 & d(X, \text{obst}) > d_0 
\end{cases} \\
U_{\text{Total}}(X) = U_{\text{Target}}(X) + U_{\text{Obst}}(X)
\]
The resulting attractive force on the vehicle, determining its movement towards the target, is calculated as follows in Eqn. (27). Similarly, the repulsive force on the vehicle, determining its movement away from obstacles, is calculated as follows in Eqn. (28), and the total force is calculated in Eqn. (29).

\[ F_{Target}(X) = -\nabla U_{Target}(X) = -0.5 \cdot \alpha \cdot \nabla d(X, \text{goal}) / d(X, \text{goal}) \]  
\hspace{1cm} (27)

\[ F_{Obst}(X) = \nabla U_{Obst}(X) = \begin{cases} 
\beta \cdot \frac{[d_0 - d(X, \text{obst})]}{d_0 \cdot d(X, \text{obst})^2} \cdot \nabla d(X, \text{obst}) & d(X, \text{obst}) \leq d_0 \\
0 & d(X, \text{obst}) > d_0 
\end{cases} \]  
\hspace{1cm} (28)

\[ F_{Total}(X) = F_{Target}(X) + F_{Obst}(X) \]  
\hspace{1cm} (29)

From this, the control velocity input and heading angle input [42] can be calculated as shown in Eqn. (30) and Eqn. (31), respectively.

\[ v_c = \sqrt{(\lambda_1 F_{Total,x})^2 + (\lambda_2 F_{Total,y})^2} \]  
\hspace{1cm} (30)

\[ \theta_c = \tan^{-1} \left( \frac{\lambda_2 F_{Total,y}}{\lambda_1 F_{Total,x}} \right) \]  
\hspace{1cm} (31)

Finally, in addition to meeting the simulation parameters previously defined, the Potential Field has the following additional parameters (Table 9):
Table 9: PF Simulation Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>10</td>
</tr>
<tr>
<td>$\beta$</td>
<td>500</td>
</tr>
<tr>
<td>$d_0$</td>
<td>50 m</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>10</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>10</td>
</tr>
</tbody>
</table>

4.3 Optimal Solution

The methodology used here for optimal control solution is a roadmap method, visibility graph, and assumes the environment is completely known prior to the mission and that it is unchanging throughout. Again, it is implemented as a comparison for performance (i.e. distance traversed and time traveled). Since the complete environment is known apriori, no-fly zones and other obstacles can all be represented as polygons. Even obstacles that have curved shapes or are open can be enclosed within a polygon here. This enclosure would be considered the “safe” distance around the obstacle. Additionally, each polygon is padded to allow for a minimum turn radius for the UAV to allow for planning between corners of polygons.

From the polygons within the area, a visibility graph is created by connecting the corners of each polygon. Also included in this visibility graph is the vertex of each target. The Floyd-Warshall algorithm [43] is then used to calculate the shortest path between target locations and the UAV initial position. The algorithm is a type of dynamic programming and calculates the shortest path by comparing all possible paths through the graph between each pair of vertices. It incrementally improves an estimate on the shortest path between two vertices, until the estimate
is known to be optimal. The shortest path is then smoothed to ensure that it is dynamically feasible using the algorithm in [44].
CHAPTER 5: RESULTS

In this chapter, the results of testing the Fuzzy Logic Controller are presented. This includes validation of the methodology and analysis of the performance. Therefore, it is shown here that the system meets the specified constraints; is capable of operating in complex environments and capable of target re-tasking in-flight; and performs excellent when comparing the distance traversed, time travelled, and control effort to optimal and potential field solutions.

5.1 Control Methodology Validation

Like mentioned in the previous chapter, the Fuzzy Logic Controller (FLC) was validated in complex environments. Examples of circumstances with polygon obstacles are shown below in Figure 33. It is shown that the FLC navigates to the targets in real-time while avoiding obstacles. Like mentioned previously, all environments were formed such that a plausible solution could be found. This means that the UAV can adequately navigate around obstacles and that targets are a “safe” distance from the obstacles.

![Figure 33: Examples of the Fuzzy Logic Controller with Polygon Obstacles](image-url)
5.1.1 Constraint Verification

Beyond validation that the technique allowed for motion planning in an unknown environment, it was essentially that this logic works within the specified constraints of the vehicle. Therefore, examples were run to verify that the system was doing so. Shown below is an example of the fuzzy logic controller (Figure 34) and the corresponding acceleration (Figure 35) and heading angle rate (Figure 36). It was verified that the system never operated outside the constraints.

Figure 34: Example of the Fuzzy Logic Controller for Constraint Verification
Furthermore, the controls for this example are shown (Figure 37 and Figure 38). It is seen that there is a short time delay for both the velocity and heading angle change as expected.
Figure 37: Controlled Heading Angle and Actual Heading Angle versus Time of the Example

Figure 38: Controlled Velocity and Actual Velocity versus Time of the Example

5.1.2 Complex Environments

Again, the methodology was tested against obstacles that can cause local minima for potential fields and obstacles that can be complex for traditional path planning algorithms (non-polygons). Examples of these circumstances are shown below in Figure 39. It is shown in these figures that the FLC is able to navigate around these obstacles using the simple logic described.
This illustrates the advantage to using fuzzy logic in that it’s adaptable to additional logic, even when only using local information to path plan.

5.1.3 Target Re-tasking

As discussed previously, the use of fuzzy logic as a path planner was verified on various situations and environments. Because this is done in real time, another advantage of using fuzzy logic is the ability to re-task a UAV mid-flight. This was verified by adjusting the objective location in the middle of the simulation. An example of the solution can be seen below in Figure 40 with various numbers of targets.
5.2 Performance

As described in the previous chapter, the fuzzy inference system was compared to both an optimal path planning methodology and an artificial potential field method. An example of the comparison between a fuzzy logic control solution and an optimal control solution is shown below in Figure 41 in a relatively complex environment. Additionally, an example of a comparison between a fuzzy logic control solution and an artificial potential field solution is shown in Figure 42. As can be seen, the potential field solution and fuzzy logic solution are very comparable.
While it is important that the FLC is at least somewhat comparable to other methods in terms of distance and time travelled, the most significant contribution of this method is the reduced failure rate. It can be seen in Table 10 that the percentage of failures for the PF method (average of about 18% overall) over the FLC (average of about 3% overall) is significantly higher. Furthermore as the environment gets increasingly complex, the number of failures for
the PF method is about 1/3 of the total cases (as opposed to 5% for the FLC). These high numbers of failures for the PF method (especially for complex environments) make it unreliable to implement realistically. Additionally, although the optimal solution has no failures, it should be noted that if there is any uncertainty in the environment this percentage would increase dramatically. On the other hand, it is likely that the fuzzy logic controller would perform the same as it makes no assumptions about the environment a priori.

Table 10: Performance Results – Percentage of Failures & Control Effort

<table>
<thead>
<tr>
<th></th>
<th>Failures: Simple Environments</th>
<th>Failures: Moderate Environments</th>
<th>Failures: Complex Environments</th>
<th>Average Control Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimal</strong></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.0158</td>
</tr>
<tr>
<td><strong>FLC</strong></td>
<td>0.02</td>
<td>0.03</td>
<td>0.05</td>
<td>0.0621</td>
</tr>
<tr>
<td><strong>PF Method</strong></td>
<td>0.03</td>
<td>0.19</td>
<td>0.34</td>
<td>0.5169</td>
</tr>
</tbody>
</table>

Also, the control effort across methods is compared. The effort shown in Table 10 is the average over all simulations for each method. As can be seen, the FLC outperforms the PF method by almost 10 times. Additionally, the FLC is on the same order of magnitude as the optimal in terms of the control effort. This detail, as well as the significantly less failures, makes it a feasible approach to utilizing in real-time.

Because path planning algorithms are generally used to solve for the minimum distance travelled or the minimum time to navigate objectives, these were used for comparison for performance. This is also done because the fuzzy inference system developed for path planning only relies on local information and a set of coordinates for the target, it is likely that the UAV will not always take the optimal path (and could possibly take a very suboptimal path). The question becomes as to how suboptimal this path is, and how often are these very suboptimal
solutions taking place. That is, if hundreds of random obstacle environments are produced, how much further is the UAV using the FIS travelling than the branch-and-bound method? Also of interest is how much longer the UAV taking to traverse its path (not necessarily the same as the distance for the FIS and PF method, because the velocity is not constant). As mentioned previously, the environments were broken down into three categories of density: simple, moderate, and complex. Figure 43, Figure 44, and Figure 45 show the number of cases (out of 300 for each category) as the optimality of the solution decreases in terms of distance traversed for each category, respectively. Additionally, Figure 46, Figure 47, and Figure 48 show the number of cases (out of 300 for each category) as the optimality of the solution decreases in terms of time travelled for each category, respectively.

![Distance Comparison for Simple Environments](image)

Figure 43: Extra Distance Traversed for the Approximate Solutions in Simple (Least Dense) Environments
Figure 44: Extra Distance Traversed for the Approximate Solutions in Moderate (Medium Density) Environments

Figure 45: Extra Distance Traversed for the Approximate Solutions in Complex (Very Dense) Environments
Figure 46: Extra Time Travelled for the Approximate Solutions in Simple (Least Dense) Environments

Figure 47: Extra Time Travelled for the Approximate Solutions in Moderate (Medium Density) Environments
As can be seen in the previous plots, the fuzzy logic solutions perform very well when compared to optimal solutions. 66% of cases lie within 90% of the optimal in terms of distance traversed, and 56% of cases lie within 80% of the optimal in terms of time traveled. Also, all of the cases have an average of being within 7.6% over all cases (as compared to 9.7% for the PF method). The main discrepancy in the time until completion comes from the necessity of the UAV to slow down near the obstacle so that it can adequately maneuver without collision. However, this allows it to have less failures overall. It should also be noted that the optimal solution does not take vehicle dynamics into consideration and assumes that it can travel along the edge of an obstacle. Realistically, the optimal solution would be altered slightly to allow for a little “padding” around the obstacles. The fuzzy logic solution already does this. Therefore, the fuzzy logic solution is closer to the optimal cases than is indicated here.
CHAPTER 6: CONCLUSIONS

Presented here is the formulation and validation of a Fuzzy Logic Controller (FLC) for motion planning in real-time in a two-dimensional, unknown environment. The fuzzy inference system was verified on various, stationary obstacles and for moving targets (re-tasking mid-flight). The FIS was compared to an optimal approach that uses visibility graphs but requires complete information about the environment apriori, and an algorithm based on Potential Fields (PFs) that also motion plans in real-time.

In real life, aircraft may have previous knowledge about some of these obstacles, but it is often the case for certain missions that there is little or no information about the environment. This is partly due to the more frequent use of UAVs for “dull” missions like surveillance and exploration. Besides, sometimes it is better to make no assumptions about the environment than to make wrong assumptions that generate an initial solution that will constantly need to be updated throughout the mission. Additionally in many scenarios, it is important that the aircraft is capable of performing evasive maneuvering throughout the mission. Maneuvering capability is also vital when mission objectives change and correspondingly, the targets’ locations or objectives change. These situations require a control method that allow an UAV to make local decisions about obstacle avoidance while motion planning. Ideal characteristics of a “good” controller with these objectives is one that has little or no failures, plans a near-optimal path, and uses a minimal amount of control effort.

The comparison to other path planning methods showed that the FLC gets good results, and they can be obtained online with any type of environment. That is, the control logic required only local knowledge to navigate a UAV to a final objective, assuming a feasible solution exists.
This was shown to be very reliable, with only about a 3% error rate over all cases. The errors that did occur were when the UAV navigated into an area that did not have an outlet, and it couldn’t appropriately maneuver out. While this method isn’t faultless, the APF method showed a much higher failure rate at about 18% overall (34% for complex environments). Additionally, the APF method used significantly more control effort (about 10 times more overall) than the FLC.

It was also shown that in most cases when the environment is relatively simple (only contains polygons or “closed” obstacles) the FIS produces near-optimal solutions consistently. That is, the FLC was able to perform within 5.7% of the optimal solution in simple environments, within 10.7% for moderate environments, and within 6.5% for complex environments for distance travelled (7.7% overall). This beat the overall performance for the APF method at 9.7%. While the performance is less reliable for obstacle enriched environments, the results are still able to be obtained in real-time and with reasonable closeness to optimality. As environments get increasingly complex or progressively larger, the optimal solution will be harder to obtain in a reasonable time. This will make it impossible to rely on acquiring an optimal solution even if the environment is completely known prior to flight.

The results presented in this work show that the FL algorithm outperforms the APF method in reliability of success of a solution, in near-optimality, and amount of control effort used. What’s more, it was able to do so in real-time with local information, which the optimal solution was not. While quantitatively it is obvious that the FLC outperforms the APF and optimal methods, there are several qualitative advantages to the fuzzy controller. The reduced failure rate is a distinguishing attribute of fuzzy logic. Fuzzy Logic allows the user to capture much more information about the environment in a more efficient manner. Additionally, the
tools available allow the user to easily develop and manipulate FISs. This makes it a very powerful tool that allows the user to see the effects of incorporating additional information with minimal effort.
CHAPTER 7: FUTURE WORK

The advantages of using Fuzzy Logic (FL) for a motion-planner were demonstrated in this research: few failures, near-optimal paths, and low control effort. It is these advantages that make it a good tool for further development. Some possible obvious future works that weren’t explored in this research are motion-planning in three dimensions, avoidance of dynamic obstacles, and more robustness testing.

Because FL is such an exceptional tool, the methods described here for two-dimensional path planning can easily be extended to three dimensions. The same logic for obstacle avoidance and motion planning applies, and the incorporation of supplementary inputs, outputs, and rules is easy and does not add very much to the computational complexity. This same reasoning also applies when allowing for dynamic obstacles. More information would need to be accounted for about the obstacle (as opposed to only using a single point’s location and orientation). While the current logic could work for dynamic obstacles, more investigation needs to be done so as to avoid failures when the UAV and an obstacle are on a path for collision. Not only would the obstacle’s location and orientation need to be taken into account, but the velocity (magnitude and direction) would also be needed. Though again, adding inputs, outputs, and rules is relatively simple and straightforward when using FL.

The parameters used for the simulation were a result of referencing others’ work and fine-tuning. More work needs to be completed to determine the effects of varying parameters. If all parameters are scaled similarly, the performance of the controller should be in line with what is presented here. However, this needs to be verified in addition to analyzing the performance if one or two parameters vary. It is suspected that if the UAV had a significantly large sensing
radius relative to the size of the obstacles, very few failures (if any) would occur. Alternatively, if the sensing range is much smaller, it is likely that more failures would take place. The effects of varying these, as well as other (i.e. time delays, maximum and minimum velocities, safe radius, sampling time, etc), parameters on the performance of the FL controller need to be examined further.

While the control methodology works well, it has several other limitations that could be improved upon further also. One assumption that was made in the simulations was that a feasible solution existed. This might not always be the case when maneuvering in an unknown environment. It can’t always be assumed that the target is going to be far enough away from an obstacle. Additionally, this logic only keeps track of its environment at each time step. If a global picture of the scenario was kept, it could be used to help make better decisions in a global sense or take more information into account in the FIS. This means it could provide solutions that are closer to optimal and have fewer failures. This would be especially useful when there exists a solution, but it is very difficult to find.

Another improvement that could be made is in the logic used to redefine the local environment and that allows the UAV to navigate out of concave and symmetrical obstacles. While the failure rate is significantly reduced, the failure rate isn’t zero, and it limits the algorithm to be used only in static environments. Even though UAVs are becoming more accessible (available and cheaper), any failure is a loss of resources. Because the logic creates a virtual obstacle between any obstacles detected, it could misrepresent the environment and miss a possible solution if an obstacle is dynamic. Again, if a global picture of the scenario was kept, and if multiple time steps were stored, sufficient logic could allow for this problem to be solved without difficulty. Fuzzy logic is flexible and allows for the addition of more rules easily.
In collaboration with other engineering departments and the department of applied science at the University of Cincinnati, the future of this research is to bring about a paradigm shift in the management of aerial resources for the control of wildfires. Emerging technologies are bringing about new methods of sensing and detecting fires, predicting fire growth in real-time and collaborating vehicles intelligently. While path planning presents a small portion of the big picture, slowly building in complexity will lead to significant results. Incorporating these results into a multi-agent wildfire fighting system, various agents work together to combat a fire. In this more complicated situation, there would be multiple UAVs used to both detect and extinguish; multiple fires would be ablaze; and in some instances, only partial information would be available about the fires or terrain. Employing the expertise of woodland firefighters, a way to optimize the resources on hand given the information coming in from various sensors and agents can be developed and used for decision-making. By making use of autonomous systems, it would free up the available people to better allocate resources depending on the situation. An example of this multi-agent fire-fighting scenario is presented below in Figure 49.

![Multi-Agent Fire-Fighting Scenario](image)

Figure 49: Multi-Agent Fire-Fighting Scenario
REFERENCES


   http://www.asdnews.com/news/27127/Predator-Series_UAS_Family_Achieves_1_M_Flight_Hours.htm


   http://marsrovers.nasa.gov/.


     Avoidance of UAV’s – Fuzzy Logic Approach,” North Carolina A&T State Univ., The 2005 

14. Dubins, L. E., “On Curves of Minimal Length with a Constraint on Average Curvature, and 
     with Prescribed Initial and Terminal Positions and Tangents,” American Journal of 

     Avoidance of Unmanned Aerial Vehicles,” Indian Institute of Science, Journal of Aerospace 


     Vehicles,” Wiley, West Sussex, United Kingdom, Jan., 2011.


