FUZZy TimeE-critical Spatio-Temporal Pong (FUZZ-TEST Pong)

Prepared By:
Brandon Cook, Third Year in Aerospace Engineering, ACCEND (Accelerated Engineering Degree Program)
Sophia Mitchell, Third Year in Aerospace Engineering, ACCEND

Reviewed By:
Dr. Kelly Cohen
REU Faculty Mentor
Associate Professor
Aerospace Engineering & Engineering Mechanics
University of Cincinnati

Submitted To:
The 2012 Academic Year NSF AY-REU Program

Sponsored By:
NSF Type 1 STEP Grant
Grant ID No.: DUE-0756921

College of Engineering and Applied Science
Department of Aerospace Engineering,
University of Cincinnati
Cincinnati, Ohio

September 19 – December 7, 2012
Abstract

There are an ever growing number of aerospace applications demonstrating the emulation of human decision making using fuzzy logic. In an effort to satisfy this demand, a MATLAB simulation of the classic arcade game PONG was utilized in conjunction with a fuzzy logic intelligence system that uses real-time reasoning and situational awareness, to emulate a human player. This simulation incorporates an intelligent team of two players capable of moving with two degrees of freedom, both translation (up and down) and rotation (clockwise and counter clockwise), to control the trajectory of the ball. By capitalizing on the opposing team’s inertia, location, and gaming strategy, the fuzzy logic paddle can react to the infinite number of interactive situations to defeat its opponent. This game is one of many practical examples of how fuzzy logic can be implemented into real life robotic applications.

Acknowledgements

I would like to thank the NSF-REU program for providing me with the framework for which I could conduct this research. This research was made possible through funding provided by the NSF CEAS AY-REU Program, Part of NSF Type 1 STEP Grant (Grant ID No.: DUE-0756921). I would also like to thank my mentor, Dr. Kelly Cohen, for all of the guidance and encouragement he provided throughout the duration of my project.

Table of Contents

Abstract....................................................................................................................................................................................2
Acknowledgements..............................................................................................................................................................................2
Outline of Report ..............................................................................................................................................................................3
Problem Statement ...........................................................................................................................................................................3
Introduction....................................................................................................................................................................................3
Research Methods ..............................................................................................................................................................................6
Results and Discussion ......................................................................................................................................................................15
Conclusion ....................................................................................................................................................................................18
Future Work ..................................................................................................................................................................................18
References....................................................................................................................................................................................20
Appendix.....................................................................................................................................................................................21
Outline of Report

This report is divided into two major sections. The first section is an introduction to help the reader become familiar with the topic of fuzzy logic and the second is solely based on the work I have accomplished throughout the course of this research project.

The introductory section of the report begins with the problem statement that defines my purpose and objectives for this project. Next, it will move on to a brief explanation of fuzzy logic, and give a short literature review allowing the reader to see how this research fits into the progress of intelligent robotic systems and collaborative robotics.

The second part of the report will include my research methods describing the simulation environment. The results will come next to show the effectiveness of using collaborative linguistic reasoning. Finally, the conclusion and recommendations for future work will complete the research paper.

Problem Statement

There are an ever growing number of aerospace applications demonstrating the emulation of human decision making using fuzzy logic. The main research challenges include situational awareness and decision making in an uncertain, time critical, spatio-temporal environment. A completely autonomous, robotic, intelligent team would be very useful in applications including, but not limited to, space robotics, celestial body exploration and colonization, Unmanned Aerial Vehicles (UAVs), homeland security, and disaster relief programs.

Introduction

Fuzzy Logic

Fuzzy logic allows for classification of variables in manner which resembles human-like reasoning. Normally, in computer programming, binary logic is used, where classifications are either zero or one, on or off, black or white. However, in real world applications using a binary system for decision making would be ineffective, as events in the universe exist on a continuum. Fuzzy logic takes this continuum into account, allowing the program to make decisions between these definite boundaries.

For example, if the binary boundaries are zero and one, a fuzzy output could be anything between these two numbers.

Another way to think about fuzziness is through looking at a spectrum of light, as seen in Figure 1. While it’s obvious that the color orange appears on the spectrum, there is no specific point where orange begins or ends. If one hundred people were to point to their perception of an ideal orange on the spectrum, “orange” would become several places on the spectrum with varying degrees of red and yellow. Each response would be both unique and correct.

There are several different components to a fuzzy logic system. All fuzzy systems begin with an observed “input” variable that is taken into the system. This input is assigned a degree of membership to one, or more, fuzzy sets known as membership functions. These membership functions are then tied to output membership functions by linguistic fuzzy patches known as rule

Figure 1: The light spectrum is fuzzy, as is nature.
sets. These rule sets are IF-THEN statements that work by stating IF (input) THEN (output).

![Diagram of Max-Dot Inference and Combined Result of Both Rules]

**Figure 2: Fuzzy Linguistic Reasoning**

Figure 2 depicts how an output is obtained. For each variable, its degree of membership to each membership function is found. Then the operators AND or OR are used to create an output for each variable. As shown in Figure 2, when using AND the system takes the minimum of the membership values, while OR takes the maximum of the membership values. If there is only one input variable this final output is used; however, if there is more than one variable, the output sets for each variable (depicted as light and dark blue in Figure 2) is combined into a single unioned result. This result is then summed to find the center of mass of the piece. This center of mass point then becomes the crisp output of the fuzzy system.

**Literature Review**

Recent work in robotics has introduced several means for fuzzy robotic collaboration. Innocenti and Salvi [6] introduced the architecture of multiple cooperative controls where agents (specialized independent systems) within a single robot work together in a decentralized system to bring the robot to its goal. In the decentralized system, however, agent competition for importance showed potential for issues in certain situations where two agents would have equal importance, causing the system to fail.

H. L., Gupta and Messom [5] used the same multiple cooperative control architecture, however, their focus was on role assignment between a team of fuzzy robotic soccer players. This role assignment strategy was useful, however, once a robot got to the ball there was no strategy for kicking other than directing it towards the goal. Therefore, the means of role assignment in the research by H. L., Gupta and Messom was adjusted and developed in this paper’s research into a means for deciding between strategies within the simulation.

Matko, Klančar, and Lepetič [7] observed two things in their research: one being fuzzy filtering of cameras for use in robotic soccer, and the other using sequences of actions to anticipate what the opponent will do. While both are useful developments in robotics, the latter is an interesting step towards
more human-like situational awareness within robots. However, the trend continued with the robots not having much of a strategy once they got to the ball other than “kick it towards the goal”.

A very recent development by Barker, Cohen and Sabo [2] introduced the architecture of cascading fuzzy decision making. This layered reasoning closely emulates human reasoning, allowing for more sophisticated situational awareness, more advanced collaboration and the use of pre-existing “plans” for certain situations.

Furthermore, in an effort to enhance my knowledge on the topic of fuzzy logic, I looked into Type II fuzzy logic through the discussions of Mendel and Wu [10]. Although this type of fuzzy logic was not used in this research project, it is a highly valuable topic to be considered for future work. This type of logic differs greatly from Type I fuzzy logic (foundation of this research) in that it creates fuzzy boundaries on its membership functions. A graphical representation of such characteristics can be seen in Figure 3.

![Figure 3: Example of Type II Fuzzy Logic Membership Functions](image)

By creating indefinite boundaries to the membership functions, as opposed to the finite boundaries seen in Figure 2, the robot can begin to grasp topics that we as humans take for granted with regards to complexity. This standard of having fuzzy boundaries has a large number of applications in the real world, yet is quite challenging to understand.

Through the use of Type II fuzzy logic as a primary robotics logic system, the robot can begin to grasp these intangible concepts and begin to apply them comparable to that of a human. A fine example of this feat would be judging the degree of eye contact during a conversation with a fellow human. If one was to observe a conversation between two humans and then record on a scale from 1 to 10 the amount of eye contact that occurred between the two subjects, one would have a very challenging time defining how much eye contact is deserving of a 1, or 2, or 10. Furthermore, if this same trial was then conducted for a second observer, they too would have great difficulty completing this feat of forming the exact same boundary definitions as observer 1. Although they may be able to use logic and memory of past conversations to rank the trial similarly, there is no discrete boundary definition.
This is the very essence of Type II fuzzy logic. By having a “fuzzy” fuzzy set, through the implementation of fuzzy membership functions, the robot can begin to grasp its own view and knowledge on related topics. This type of logic is highly applicable for robots to understand human language, voice fluctuation connotations, and body tone.

Research Methods

PONG

The game of PONG was originally designed as a two-player, two-dimensional game, where the opponents line up across from each other hitting a ball back and forth as seen in the game of tennis. The object of the game is to hit the ball past the opponent into their respective goal. This game was considered a good choice for the simulation as it provided a dynamic, uncertain, spatio-temporal environment. It is dynamic due to real-time gameplay being a natural aspect of PONG, uncertain due to human input, and spatio-temporal since success in the game is highly dependent on being in the right place at the right time.

The base of this PONG simulator was written by Sophia Mitchell, which was originally based off of the 1980s Atari game PONG. This original PONG package incorporated an additional component from the classic arcade game, as it utilized two teams consisting of two players, as opposed to the classic one-on-one gameplay. This team PONG package was, however, solely capable of translating up and down and had a “convex” paddle bouncing strategy. This “convex” paddle was used to control the bounce trajectory of the ball as such: when the ball struck the paddle, the instantaneous location of the ball with respect to the center of the paddle, would create a normalized vector that determined the new trajectory angle of the ball. In other words, if the ball was struck at the top or bottom of the paddle the ball would travel upwards or downwards, respectively. As the ball was struck at more extreme ends of the paddle the ball would bounce at a larger angle off of the paddle. Therefore, the most offensive return of the ball would be accomplished by striking the very top-most or bottom-most portions of the paddle.

From this base version the new gaming capabilities were added. These gaming capabilities include the addition of a new degree of freedom for each paddle (rotation), new bouncing characteristics to change ball control strategies, and new Fuzzy Inference Systems that control the autonomous fuzzy paddles. Each of these characteristics has been successfully implemented into both a humans vs. robots game and a robots vs. robots game.

For the robots vs. humans gaming style the uncertainty is provided by the human players. Seeing as the robots have no insight into the mind of its human opponent, they can only react through situational awareness. By using sensors (inputs) the robots can be aware of where the opponents lie, how they are moving, where the ball currently is, and where it is going, to make intelligent decisions. Similarly, a robots vs. robots simulation was created to test if the robotic teams were well matched. Although small issues still lie in each game, the results of this research were found to be quite promising.

In order to gain the expertise on PONG necessary for the creation of winning fuzzy rule sets, a carrot-and-stick method was utilized. This method included careful observation of a real life tennis match, while paying attention to the subconscious methods the human team uses to score on the other team. These strategies were written down and later used in the creation of the fuzzy rule sets.
In addition to the real world observation, human vs. human interaction has already been observed through previous research. During my last research project, similar gaming characteristics were implemented into a singles PONG game. In order to observe how a human uses intuition to utilize the paddle angle to score on its opponent, a large amount of testing was conducted on the singles game. Through this research, and the real world scenario described above, the collaborative linguistic reasoning utilized by the autonomous agents was implemented effectively.

Due to previous experience in creating a singles PONG game and familiarity with the MATLAB code, the transition to a doubles game with the additional components integrated was quite a challenge. Although similar in nature, these characteristics did, however, take a large amount of time to work out bugs from the previous version, as well as implementing new boundary conditions into the game. Overall, the true challenge lied in implementing the real-world observations of human teams working together to beat their opponent.

To create the doubles human team vs. robot team game, the first step was to create the simulation. In order to do this rotation functions had to be employed, bounce functions had to be created, keystrokes from the human user had to be altered for rotation capabilities, and the boundary conditions had to be additionally altered so that the paddles couldn’t leave the playing field.

Once the game was functioning at a near optimal level, the autonomous fuzzy robotic team was created. To do this, extensive research had to be conducted to decide which inputs, rules, and outputs would be appropriate for creating the Fuzzy Inference System (FIS) files that would control the fuzzy players. An example of the doubles PONG setup can be found below in Figure 4.

![Doubles PONG configuration](image)

Figure 4: Doubles PONG configuration

**Gameplay Trouble-Shooting Methodology**

The first step to create a paddle capable of rotation was to implement previous research on how to rotate plots in MATLAB. From this previous research it was concluded that there was a MATLAB function named “rotate” that could rotate a plotted line or object around a specific axis of rotation. This function, however, had to be severely modified so it would perform in a desired manner during gameplay. These alterations included creating an output vector that assigned the new “x” and “y” parameters for the paddle. By creating this output vector one could use the rotate function to reassign the current vector of
the paddle to its new coordinates. A subroutine was added to the MATLAB script that continuously found the instantaneous center of the paddle so the paddle could be rotated about its origin.

From this point the new bounce capabilities of the ball had to be implemented. The new bounce capability was designed to allow the user to control the trajectory of the ball solely through rotation of the paddle. However, since the “convex” paddle was still in use, gameplay was very counter intuitive; for example, if the paddle was rotated clockwise, where one would expect the ball to bounce upwards, didn’t necessarily ball would behave as expected. To make the gameplay more intuitive and to allow the user to have more control of the ball, an angle of incidence approach was utilized. This bouncing strategy utilized the instantaneous vector normal to the orientation of the paddle, to find the angle of incidence the ball made with this centerline. Once the angle of incidence was found, the new rotate function was utilized to rotate the ball about that point. After the ball had successfully been rotated, a new normalized trajectory vector was calculated to send the ball in its new direction.

The next issue that had to be overcome was redefining the top and bottom boundaries of the court. When the paddle reached the top or the bottom of the court it was to stop movement so the paddle could not go out of bounds. In the original code, when the paddle reached the top or bottom of the court, the paddle location would be reassigned so that the top of the paddle was stuck flush with the top of the court. When the top of the paddle is rotated, however, a flush top portion of the paddle would make the paddle look oblong and disfigured. To overcome this obstacle I incorporated an IF-THEN statement in the code so that if the top-left point of the paddle was greater than the top of the court, then the paddle would be restricted to solely downward movement. Therefore, the paddle wouldn’t be able to move outside of the boundaries on the court, but still capable of rotation. Slight alterations between the singles game and the doubles game were made; including changing of physical limits to ensure no part of the paddle was removed from the playing field, yet could still cover all portions of the court.

In an effort to add the rotational degree of freedom to the autonomous agents, research on how the previous singles PONG functioned was conducted. This singles game utilized its input values to create optimal ball return capabilities. With the implementation of these characteristics into the doubles PONG game, several limitations were evident, and thus, had to be addressed.

The first major issue that was apparent throughout gameplay was the fuzzy controller not reaching its desired rotation angle. Although the command was properly sent to the paddle at its desired output, the paddle could not fully achieve this rotation. This limitation was due to the paddle not having the capability to both rotate and translate simultaneously. However, now that this capability is available for gameplay and has been optimized, the fuzzy paddle can rotate and translate freely. This allows for more reaction time from the paddle to get to its desired location with its desired rotational component. These characteristics were first troubleshooted in the singles gameplay, and then implemented into the doubles play.

Once each paddle was capable of these enhanced movement characteristics, the intelligent team was optimized to defeat the human opponent. In order to accomplish this, the strategy FIS file was altered accordingly to ensure that the paddles hit the ball to the correct location each time. Due to the back paddle sometimes hitting the ball into the back of its teammate and causing the fuzzy team to score on itself, the avoidance FIS was also modified.
Each of these additions was crucial to accomplish before trying to tackle the fuzzy logic portion of the project. With these changes the game had the full capability to play a fluent and bug-free humans vs. robots game.

**Fuzzy Paddles**

While gaining knowledge from the basic human vs. robot singles game, it was noted that the player could control where they want the ball to go by utilizing different angles of the paddle. A clockwise rotation hit the ball up, while a counter clockwise rotation hit the ball down. Hitting the ball with a non-rotated paddle would simply return the ball with no control over its trajectory, which was considered more defensive. Using the highest rotation the paddle (-33° or 33°), one could make the ball bounce in a very offensive move. When this strategy was used it was difficult for the human opponent to hit, especially at an increased ball speed due to the requirement of a shorter reaction time. The breakdown of the fuzzy paddle into these offensive and defensive angles can be seen in Figure 5.

![Figure 5: Breakdown of Fuzzy Paddle into Offensive and Defensive Areas](image)

A large aspect of the game of PONG is the user’s ability to calculate where they believe the ball will intercept their paddle; therefore, it was necessary for the robot to be able to do the same. A function was created within MATLAB that took into account the ball’s current position, movement vector, and speed. Using these variables a temporary vector was calculated by the robot modeling the trajectory of the ball’s movement. If the ball were to bounce off the top or bottom walls, these same bounce calculations were used to ensure the trajectory was correct. The point of intersection was then calculated, that is, the point where the ball was going to cross the autonomous paddle’s plane of translation.
Since the FIS file was to move the paddle up, down, clockwise, or counterclockwise, it was decided the output would alter a variable moving the paddle in those designated directions. When a human plays PONG, they use four buttons on the keyboard, one to move the paddle up, one to move the paddle down, one to rotate clockwise, and one to rotate counterclockwise. These up and down keys were connected to a variable in the MATLAB code that either added one (makes the paddle move up) or subtracted one (makes the paddle move down) to the center of the paddle’s current position. The left and right keys were connected to a variable in the code that either added 3° (counter clockwise rotation), or subtracted 3° (clockwise rotation), respectively. Therefore, the translation output for the fuzzy player was to be any number between these one and negative one limits, so the FIS file could move the paddle up or down as needed. The rotation output was to be any number between the three and negative three limits, so the FIS file could rotate the paddle clockwise or counterclockwise as needed.

Once the movement was made possible, the fuzzy paddle had to be manipulated so that it reached the ball intersection point. To do this the input variable was calculated to be the difference between the point of intersection, and location of the center of the paddle at that point in time. For each iteration of the program, the difference was re-calculated so the paddle would have a continuous input. The figures below shows that the FIS file created a continuous, smooth output for both the translation and rotation components.

Figure 6: Paddle Translation
With these output possibilities in place it was fairly simple to move the paddle to its desired location. For example, if the input value (ball intersection) was given membership to FarBelow the paddles current location, then the output (paddle movement) would be a crisp number with membership to DownFast. All of the rules used in this simulation used the AND iterator for overlapping functions, as it allowed for smooth movement and transitions between membership functions as the paddle’s center got closer to the point of intersection.

Once movement was possible, the logic then had to be created to design a competitive intelligent team. Much of the logic for the doubles game was inspired by professional doubles tennis matches: The Australian Open 2010 (Bryan Brothers) and the team of Federer and Mirka.

While observing doubles tennis it was noted that the two players used their own positioning to show their role in the game, as well as, show if the team feels the game is offensive or defensive. Generally the back of the doubles court is considered defensive area, since it allows one more time to see the ball coming, and there is less of a chance of error. If the team is leading in a game they are more likely to both move to the back, making them both defenders so the other team cannot score and catch up. Conversely, the front of the court (closer to the net) is considered and offense area, since it doesn’t allow the opponent much time to observe the ball’s movement before it gets to them. If the team is losing and wants to get a few quick scores in, they will both move to offensive positions closer to the net. In normal gameplay, having both players in the offensive or defensive positions is usually rare, because if one makes a mistake in offense there’s no backup plan, and if both are in defense, there is little chance of scoring. Therefore, the third way teams position themselves is with one player in the front playing offense, and one in the back playing defense. To cover as much of the court as possible, each player will stand about 1/4 of the court’s width from each lateral side. This third setup was used in this particular research.

Another note that was made was how the players communicated. While studying these games it was noted how the player in front would hold different hand signals behind their back to notify his partner.
and correspond certain plays or movements. These hand signals were an extremely efficient way of communicating and adapting to the game while playing, since pre-determined plays don’t always go as planned, and yelling across the court would leave no surprises for the opponent.

In this two player simulation, it was decided the most reasonable set up for players was the third set up mentioned above, with one player in front, one in back, and starting 1/4 of the court’s width from each side. To keep simplicity only one translation plane (up and down) was allowed for the paddle’s movements. This essentially meant the paddle in front was the offensive player, while the player in the back was considered the defensive player. FLIP 1 became the name of the player in the back, and FLIP 2 became the name of the player in front.

The question arose on how to exercise similar communication between the two fuzzy players that was observed between the two professional tennis players. It was not considered to be strategic for both players to be competing for the ball, as the player in front would usually get it, half the court would always be left open and more difficult to defend, and the strategy associated with player placement would not be utilized effectively. Looking back at the professional tennis game, the player in front, as stated before, would give hand signals to their partner telling them who should go, and what to look out for. To mimic this behavior it was decided that the robot in front in the offensive position would make the final decision on who should go, and then each robot, knowing the plan, would individually decide what it needed to do as an offensive or defensive player. To aid in the layered decision making that comes with communication and offensive/defensive strategies, cascading fuzzy logic was utilized. A step-by-step explanation of this communication is described in the figures and descriptions below.

![Collaborative reasoning used in FLIP](image-url)

**Figure 8:** Collaborative reasoning used in FLIP
Figure 8 above is an overview of the communication and thinking that goes on within the FLIP team. The instant one of the human player’s paddles hits the ball, this cascading fuzzy and communication system within the FLIP team works to find who should go for the ball, where each player needs to be, and what strategy they need to utilize. Figures 9 through 12 are divided into four areas for more simple explanation of each.

**Figure 9:** Collaborative Reasoning Part 1

Part 1 (Figure 9) shows the initial process that goes on within FLIP 2 who is in front and the team leader. FLIP 2 calculates the distance between it and the point of intersection, while also waiting for FLIP 1 to send its own distance information. Meanwhile, FLIP 2 also uses a FIS file to calculate the game type (described more in depth in the Results section), classifying the game as a certain amount offensive and a certain amount defensive. With all of this information, FLIP 2 then decides which fuzzy player should go for the ball.

**Figure 10:** Collaborative Reasoning Part 2
Part 2 (Figure 10) shows this decision making process, and the results for each decision. In an effort to optimize the effectiveness of the intelligent team, the paddle closest to the predicted intersection point was assigned to go hit the ball. With this being said, if FLIP 1 was in charge of hitting the ball, FLIP 2 then had to move away so that it would not accidentally hit the ball backwards into its own goal. To ensure this did not happen, an avoidance FIS was created. By taking into account the intersection point and the back paddles angle, the front paddle knew where the back paddle was located, and where it was trying to hit the ball. With these input values, the front paddle was then able to move out of the way of the ball's trajectory in a timely manner so that the ball does not hit it in the back.

With the ability to move the center of the paddle to the desired location at the ball intersection location, the paddle had to now decide where to hit the ball to score on its opponent. In order to find the optimal ball placement many inputs were created in the FIS file including: opponent location, opponent inertia, and game strategy. If FLIP 2 decides it should hit the ball, it uses its fuzzy situational awareness and these multiple inputs to create a unique output (angle of rotation) to hit the ball where the opponent has the most difficult time to intercept. For example, if both opponents are at the very top of the court, moving up, and the fuzzy paddle has been assigned an offensive strategy, the fuzzy paddle would want to hit the ball in an extremely offensive down strike, sending the ball to the very bottom of the court.

In addition the paddle must take into account its own location before setting its desired paddle angle. For example, if it is located at the very bottom of the simulation, and the best place to hit the ball is on the bottom of the opponent’s side, hitting the ball down will cause it to hit the wall and bounce up. In this position the best thing to do would be to hit the ball straight. Finally FLIP 2 rotates based on where it wants to hit the ball.

**Figure 11:** Collaborative Reasoning Part 3

Part 3 (Figure 11) depicts what FLIP 1 does before the decision has been made of who should go for the ball. At this stage the ball’s point of intersection is found, and the difference between where it is and where it needs to be is calculated.
Part 4 (Figure 12) shows the strategies FLIP 1 utilizes. If FLIP 2 tells it to go, it uses the same situational awareness as previously described for FLIP 2. However, if FLIP 2 tells it to wait, but that it needs help, it will go to where the ball would otherwise hit the center of the paddle. This ensures that if FLIP 2 misses, there is a backup plan. Although this strategy ensures that the ball will be hit, it does, however, leave the other portion of the court unguarded. If FLIP 2 tells it to wait and doesn’t need help, FLIP 1 moves to cover the other side of the board.

Once the autonomous fuzzy players were given the ability to decide what to do when the ball was going towards it, it was necessary to decide what should happen when the ball is moving away from it. To cover the most area, it was decided the paddles would reside at either the top quarter or bottom quarter of the board with no rotation. By dispersing the robots to opposite ends of the court, the intelligent team has a much greater probability to reaching the ball no matter where on the court it is hit. Due to not wanting to occupy the same section of the court (i.e. both paddles at the top quarter or bottom quarter at the same time) cascading logic was used. First, the FLIP 2 would stay in the section of the court that was closest to its current position, and then then FLIP 1 would move to the opposite side of the court.

Results and Discussion

This two player game worked very well in Beta testing. The ball trajectory calculation worked marvelously in finding the point of intersection, while the FIS file created very smooth movements to get the paddle to this desired location. Communication between the paddles proved to be effective and significantly aided to defeating their opponents.

The fuzzy situational awareness was developed with two inputs, one output, and several rules. The two inputs were the position of the opponents, and their movement (found from the variable containing a 1, 0, or -1 depending on what button the human user was pushing). The membership functions for the position divided the game court into five parts: very top, top, center, bottom, and very bottom. Input membership functions for the movement were: up, stationary, and down. The output
membership functions divided the court into the same five parts as the position input, and were used to determine the point the human players would have the most difficulty reaching. An example of the strategic FIS file for FLIP1 in the back can be seen in Figure 13. This figure shows the smoothness of the FIS output.

![Figure 13: Back Strategy FIS Output](image)

With the addition of more FIS files, the game slowed down considerably. The first considered solution considered was to simplify the FIS files, taking out some of the membership functions and rules. While this did speed up the game a little, the problem still persisted, and the strategies were no longer as precise. With the original FIS files back in the game, the simulation was tried on a different computer that had a higher processing speed. This allowed for the realization that the problem lay in the computer’s processing speed, and not the game itself. To allow for easier processing two things were then done within the MATLAB code. First, the function used to refresh the playing field was made independent of the game’s iterations. This was done by creating a timer that would refresh the game every 0.0001 seconds, which was enough to take away some of the choppiness of the game. The next change was to take the lines that loaded the FIS files and put them before the main loop for the simulation. In doing this, the necessary processing time was cut significantly as loading FIS files is not a small process. This change actually made the program progress too quickly for a human to react to the game, so a small pause was added to each iteration so, while smooth, the game worked at a playable speed.

Aside from logistical corrections to the script to create a more fluent game play, many changes had to be made to the collaboration aspect between the robots. On the previous version of this game the fuzzy paddles were programed to always return to their respective sides of the court when the ball was traveling away from them. However, this was not very effective due to having the paddles travel to the other end of the court and back nearly every volley. With this old strategy the opponent could easily capitalize on this predictive strategy to defeat the robot team. To fix this issue, the front paddle would stay on its respective side, and then tell the back paddle to move to the other side of the court to cover the most area at one time. The front paddle was chosen as the primary paddle to stay in its location due to having the shortest reaction time. By sending the back paddle to the other side of the court, it could have more time to hit the ball if it was quickly returned to that portion of the court. All other collaboration issues were usually caused by simple errors within the MATLAB code, and could easily be solved once the real problem was pinpointed.
At this time the MATLAB code can handle two types of gameplay, the first being a human team
playing against a fully operational fuzzy opponent, capable of both rotation and translation. The second
type of gameplay is where two robotic teams play against each other. This second game type was an
effective way to find the problems within gameplay and allow trouble-shooting to optimize the game.

Numerical results of Beta testing of the robots vs. robots game can be seen in Table 1. This table
shows the scores from a total of 30 serves to verify that the teams were equal in ability and capable of
being competitive. Clearly from these results both robot teams were highly competitive. During testing,
each volley would take nearly three minutes for one of the robots to score. Nominally, each point scored
was not due to bad fuzzy logic reasoning, but to glitches in the gameplay itself.

<table>
<thead>
<tr>
<th></th>
<th>Red Team</th>
<th>Blue Team</th>
<th>Winner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robots vs.</td>
<td>15</td>
<td>15</td>
<td>DRAW</td>
</tr>
</tbody>
</table>
| Robots        |           |           | Table 1: Doubles Robot Team vs. Robot Team Results

One major issue that was discovered through this portion of testing was the programs inability to
plot paddle rotation changes properly. With the enhanced ability to rotate and translate simultaneously, it
was necessary to first calculate the new rotated paddles vector quantities that composed the paddle’s
boundaries. Once the rotated vector quantity was calculated, a second vector quantity was calculated to
translate the paddle up or down accordingly. Lastly, the paddle would be plotted at this new position.
Due to the tremendous calculation speed and frequent changes of the new paddle vector, although the
calculations were made correctly, the plot would lag behind. This meant that although the paddle was
assigned to a rotation angle with value zero, on the screen it would be plotted at an angle. This would
allow the paddle to rotate more than desired (outside of the -33° to 33° operation range). An example of
this plotting error can be seen below in Figure 14.

![Rotation Error Example](image)

Clearly the front left paddle has rotated more than the allowable 33°, making the paddle
ineffective at hitting the ball. In order to compensate for these rotation errors, the paddles were reset to
their initial positions and orientations on the court after each point. This new feature made the game more
realistic and reflective of a real-life tennis match. During a tennis match, after every point has been scored, the teams will reset and begin the next volley.

Beta test results for the robots vs. humans game have been tabulated into Table 2 showing the results of two full games. For this portion of testing two veteran “gamers” were selected to test the effectiveness of the fuzzy logic. In the first match the human players were able to score two points on the robot opponents. The only two points that were scored on the robots were due to glitches in the gameplay. On the first goal the ball went through the paddle at the lower limit of the court, and on the second goal the front fuzzy paddle was unable to avoid a high angle trajectory hit from the backcourt paddle. In the second match between the robots and humans the robotic players won 21 to 0. By analyzing these results it is clear that the fuzzy team does a fine job at collaborating effectively to defeat the opposing team. It is quite obvious that the intelligent team was highly effective at communicating in a time-critical manner, while making adjustments to the infinite special scenarios to defeat its opponent.

<table>
<thead>
<tr>
<th></th>
<th>Robots</th>
<th>Humans</th>
<th>Winner</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robots vs. Humans</td>
<td>42</td>
<td>2</td>
<td>ROBOTS</td>
</tr>
</tbody>
</table>

Table 2: Doubles Human Team vs. Robot Team Results

This portion of beta testing was highly valuable in finding the small intricacies that could be improved upon in the future. Many of the items that were discovered were all gameplay errors that could be fixed with additional programing.

**Conclusion**

Creating an architecture that effectively uses cascading fuzzy logic and demonstrates collaboration between two fuzzy systems would be a useful development to fuzzy robotics. From these results it can be concluded that fuzzy logic is an effective means of imitating human reasoning and collaboration. This shows how fuzzy reasoning and fuzzy situational awareness can be utilized in emulating a human player.

**Future Work**

At this time I feel that this doubles game of PONG with the rotation factor included is nearly optimal. However, in the near future I hope to expand on this game to incorporate some more advanced features for more optimal gameplay. The main feature that needs to be incorporated is the ability to use extremely offensive moves and trickery to defeat its opponent. When trouble shooting the game for the humans vs. robots game I noticed that one can quite easily pass the ball between players to confuse its opponent. By passing the ball off of the back of the front player and then being struck again by the back player at a different angle, the opposing team would have less time to react and could be confused.

Another offensive passing strategy would be for the back paddle to hit the ball at an extreme trajectory angle towards the other paddle. This would pass the ball to the front player so it could be hit in the opposite direction. Again, this would limit the amount of reaction time from the opposing team and would be highly effective.

A third highly offensive strategy that could be implemented would be having both paddles act as if they are going for the ball (similar to the back fuzzy paddle backing up the front paddle). When both
paddles are in position to hit the ball either paddle could be used as the offensive player. Therefore, one could make the back paddle angled in one direction, and the front paddle angled in the opposite direction. At the last moment possible, the front paddle could move out of the way and allow the back paddle to return the ball. With this method, the opposing team would be clueless to who will hit the ball and wouldn’t be able to predict which direction the ball would be returned.

Lastly, the logistical error previously described in not plotting the correct paddle orientation needs to be fixed. By fixing or adding each of these components, the game would be bug free for users during gameplay and would prove to be a highly challenging game to defeat.
References


Appendix

FIS File Tables

Figure 15: Paddle Movement FIS (pdlemvmnt3)

Figure 16: Game Classification FIS (gametype)
Figure 17: Front Paddle Strategy FIS (frontstrat4)

Figure 18: Back Paddle Strategy FIS (backstrat2)
Figure 19: Front Paddle Avoidance Strategy FIS (frontavoid2)

Figure 20: Front and Back Paddle Situational Awareness FIS (oppaware_new)