USING A FUZZY LOGIC CONTROL SYSTEM FOR AN XPILOT COMBAT AGENT

ANDREW HUBLEY AND GARY PARKER
Department of Computer Science
Connecticut College
New London, CT
{ahubley, parker}@conncoll.edu

ABSTRACT—Fuzzy logic controllers provide a means of reasoning in uncertain and non-specific situations. By abstracting input values into membership sets, linguistic rule sets can be applied to reason with the information. In the space combat game Xpilot, it is reasonable to use a crisp, rule-based system for an autonomous controller, but such controllers can be outperformed by fuzzy systems. In this paper, we use fuzzy logic to develop a more versatile controller that can deal with complicated situations effectively. Previous artificial intelligence research in Xpilot-AI has not yet made use of fuzzy systems for agent control.

Key Words: Xpilot, Fuzzy Logic, Control, Autonomous Agent, Xpilot-AI

1. INTRODUCTION

The ability to adapt to uncertain or unpredictable situations is crucial to a combat agent. Fuzzy logic, the logic on which fuzzy control is based, is more comparable to human thinking and natural language than traditional logical systems, and previous research shows that fuzzy logic controllers yield results superior to conventional control algorithms [1]. Our aim was to develop a more effective combat agent than previous rule based agents, using fuzzy logic as a reasoning system.

Previous researchers have implemented fuzzy expert systems as problem solving tools. Although this is very different from a combat controller, the ability to reason with vagueness and dynamic inputs is important in both situations. An example is the research done by Adnan Shaout, Brady King, and Luke Reisner from the University of Michigan–Dearborn. Their research involved the implementation of a fuzzy rule-based system in a version of the classic game Pac-Man. They used fuzzy rule sets to control the traditional Pac-Man ghosts, and they were successful in designing an effective controller. A commonality between our research and that of Shaout, King, and Reisner is that in both cases, the framework used in the implementation resembles the BDI (Beliefs-Desires-Intentions) model [2]. The BDI model is a software architecture based on human reasoning, based on abstracting information from the environment as 'beliefs,' the goals of the agent's as 'desires', and the plans and actions taken by the agent as 'intentions.' Y. Li, P. Musilek, and L. Wyard-Scott from the University of Alberta developed a fuzzy logic agent to play a version of the 1985 Nintento Entertainment System game BattleCity. Their agent employs fuzzy membership and rule sets, and uses BDI.net, a lightweight BDI framework that works with all programming languages that support the Microsoft .NET framework [3].

Xpilot has been a platform for many agent control research topics. Matt and Gary Parker have conducted extensive research on evolutionary computation (EC) used to learn Xpilot control programs. In one example, they used EC applied to neural networks [4] to produce an effective controller, but this required a long learning process to reach the desired effectiveness of the agent. They have also done research on evolving parameters for Xpilot agent control [5]. This also was successful in creating a competitive combat agent, but has a similar 'learning curve' to reach the desired effectiveness. A fuzzy control system is not a learning controller; it is provided a rule set for decision making. This allows for immediate effectiveness of the agent, but leaves the reasoning framework up to the developer.

In this paper we present the implementation of a fuzzy expert system as a controller for an Xpilot agent. Specifically, we use separate fuzzy membership functions to handle each variable on which the agent's decision making is based.

2. XPILOT-AI

Xpilot (Figure 1) is an open-source multiplayer 2-dimensional space shooter game. It implements physics that provide a realistic feel for a frictionless space environment. It accurately represents physical forces including acceleration, inertia, and momentum. There is a wide range of maps varying in size, and the game mode can be free for all, team play, or capture the flag. In certain maps there are variations available that include weapon upgrades and ship modifications. For the purposes of our research, we use a simplified version of Xpilot. There are no upgrades and no fuel restrictions; the purpose of this research is to design a combat-specific controller.

Xpilot is normally played by a human player, who has control over the ships thrusting, shooting, and turning. The goal is to destroy the enemy ship while avoiding enemy fire and obstacles.

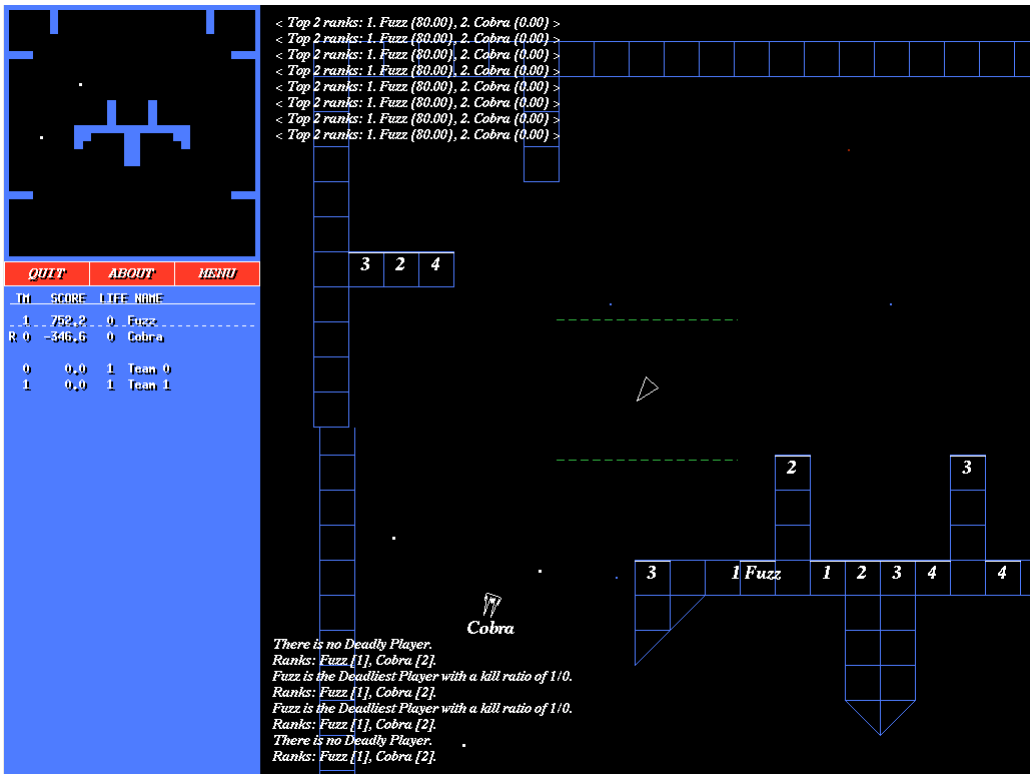

Figure 1. Screenshot of Xpilot Gameplay

Xpilot-AI (www.xpilot-ai.org) is a library of commands and calculations for an autonomous agent in the Xpilot game environment. It allows the user to access information about the current frame, including distance to walls, track, heading, number of enemies, velocity, and many other crucial variables that are present in a combat situation. It also enables control of turning, shooting, and thrusting. With these tools, an autonomous agent can compete in the Xpilot environment.

3. FUZZY LOGIC

Fuzzy logic is a logical system that works with data that does not have precisely defined values. Fuzzy systems typically employ rules to translate vague terms, such as an uncertain assumption, into system outputs [3]. Fuzzy sets are comprised of membership functions are used to determine the values for a particular variable. For instance, if the fuzzy variable is height and consists of three membership functions—short, average, and tall, the fuzzy set will calculate the

membership for each. Thus a person can be defined not simply as short or tall but as a combination of degrees of shortness, averageness, and tallness. These values are then evaluated in combination with another fuzzy variable as rules. For example if the person is tall and the door is low, a rule will give the instruction to duck. Multiple rules such as this one are predetermined and incorporated into the system. The system takes in crisp inputs (the man is 6' 8", the door is 6'), fuzzifies them (the man is a member of *tall* to a strong degree, the door is a member of *medium* to a strong degree), evaluates based on the rules (if man is tall and door is medium, duck), defuzzifies the results of the rules being applied (duck how much?), and produces a crisp output (duck to 5').

These principles can be applied to the Xpilot environment with the Xpilot-AI library. By using values provided by Xpilot-AI about the current situation, it is possible to calculate the appropriate action for the agent to take. For an Xpilot combat agent, the values used as input include wall distance, velocity, heading, track, enemy distance, shot distance, and shot angle. Each of these values is updated every frame, and the controller must provide the robot with an action to take each frame, whether it be to shoot, to thrust, or to turn a certain direction and angle. The input values are evaluated by the fuzzy controller, which provides a crisp output that is used as the deciding factor in what to tell the agent to do.

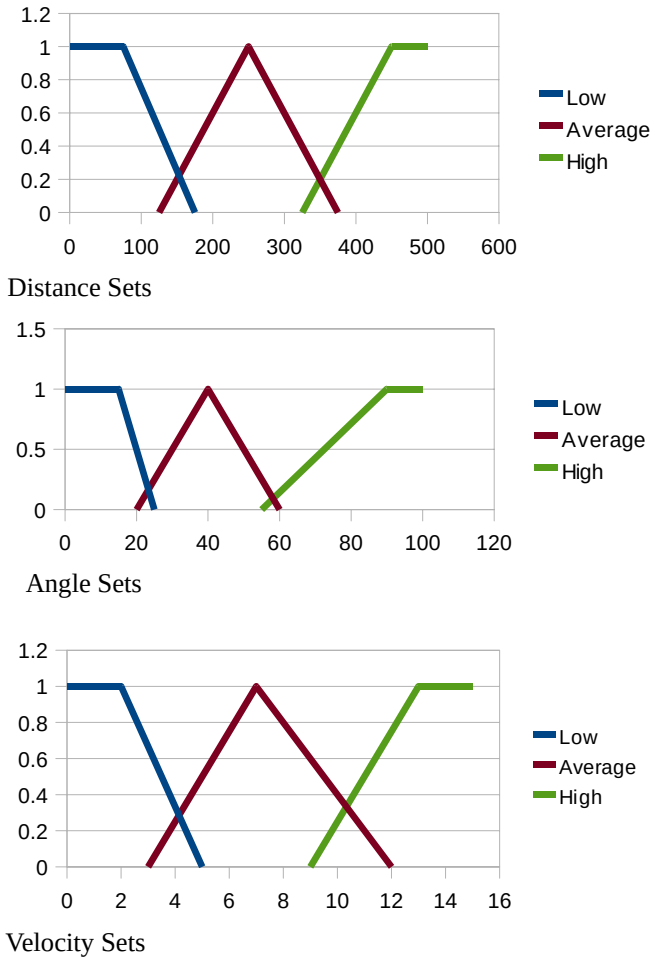

Figure 2. Fuzzy Membership Sets

4. FUZZY LOGIC CONTROLLER DEVELOPED FOR XPILOT AGENT

The fuzzy logic controller that we developed implements three fuzzy sets (Figure 2), each with three degrees of membership—low, medium, and high, which are used in three different rule sets that each produce a crisp value. The values that are passed through these membership sets are those provided by the Xpilot-AI library.

The rule sets are classified as wall avoidance, defense, and offense (Tables I, II, III). Wall avoidance is based on a combination of the ship's distance to the wall and its velocity, defense is based on the turn angle between the ship's heading and incoming fire and the distance to it, and offense is based on the turn angle and distance to the enemy. All three of these rule sets are evaluated every frame. Each input value is assigned a membership value between 0 and 1 (0 meaning not a member and 1 meaning the most possible membership) according to the membership set it is applied to. These are then evaluated by the linguistic rule sets. The final step is to defuzzify the resulting answer, which we do by finding the weighted average of the results from the rule sets.

Table I. Wall Avoidance

Wall vs Velocity	Close	Average	Far
Slow	Low	Low	Average
Average	Low	Average	High
Fast	Average	High	High

Table II. Defense

Shot-angle vs shot-distance	Small	Average	Large
Small	Low	Low	Average
Average	Low	Average	High
Large	Average	High	High

Table III. Offense

Enemy-angle vs enemy-distance	Close	Average	Far
Small	High	High	Average
Average	High	Average	Low
Large	Average	Low	Low

Each of the three rule sets produces a crisp value between 0 and 100, and the largest of these values is selected for use, based on the idea that the largest crisp value is seeking the most action from the ship and thus should be prioritized. (Figure 3) For example if the crisp values for wall avoidance, defense, and offense are 25, 80, and 63 respectively, 80 would be selected. The action of the ship is based on the selected value, and each available action (thrusting, turning, and shooting) has a range in which it is activated. For example, if the value is between 25 and 90 the ship fires a shot, because if the value is below 25 there is most likely no enemy near, and if it is above 90 there is most likely imminent danger so shooting is not a priority.

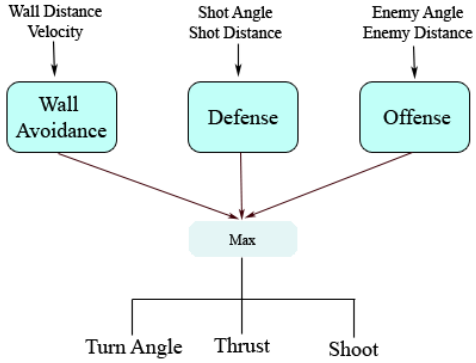

Figure 3. Framework

5. RESULTS

When we first tested our agent, we were running it against the built-in Xpilot agent. At first, the ranges that we implemented to handle the crisp outputs were not very effective, so the agent did not perform as well as we had hoped. By modifying these ranges, we could change how the agent would react to different levels of danger. For example, we encountered a problem that when the enemy was an "average" distance away, the robot would ignore the enemy and focus on avoiding walls. By checking what crisp value was calculated for offense in that situation, we could manipulate the ranges to cause our agent to focus on the enemy when in range. In order to monitor the crisp values produced by the fuzzy system at runtime, the agent displays the results of the three fuzzy rule sets, which it is basing actions upon, and whether the closest enemy is on the screen or on the radar.

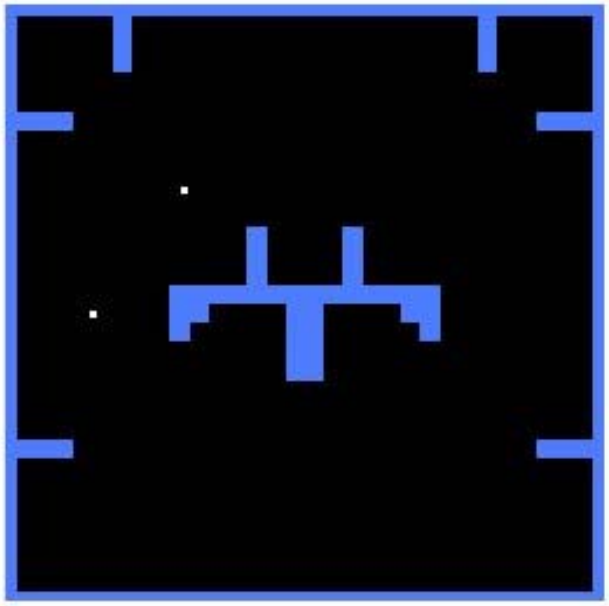

Figure 4. Map of Combat Arena. Blue areas are obstacles, white dots are player ships.

The fuzzy sets that we implemented in the first version of the agent were not very effective. We found that our original values for wall distance and enemy distance were too high, so the robot would not respond properly. By shifting the fuzzy sets to be smaller, the agent became more sensitive to smaller distances, and acted accordingly. Once the agent was running effectively, we ran tests against three rule-based robots. Each test trial ran for two hours on the map called Lifeless (Figure 4) at 25 frames per second. The combat is round-based, such that after one ship is destroyed each ship is moved to a new starting location for the next round.

The three rule-based agents that the fuzzy system was tested against in combat were Sel, Morton, and Cobra. Sel is the most advanced rule-based controller developed for Xpilot-AI to date. Morton is a basic combat agent written as a simple example of an Xpilot-AI agent and is made available on the Xpilot-AI website. Cobra is the built in Xpilot agent and is the least effective of the rule based controllers. Five separate trials were run for each combatant pair. Our agent outperformed all of the rule based systems overall (Figure 5).

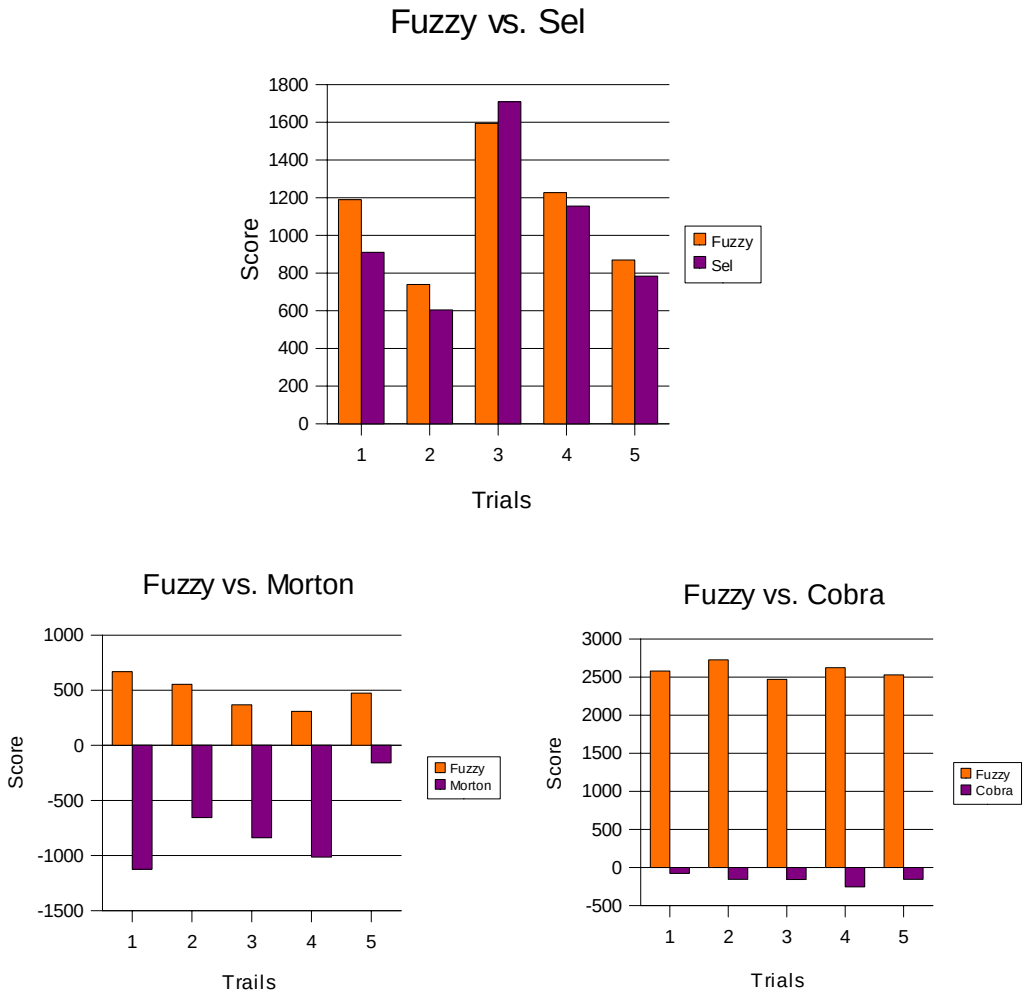

Figure 5. Final Scores of Combat Tests Between Fuzzy and Rule Based Agents

The behavior of the fuzzy agent is of interest, as it employs a very different strategy than the rule based agents. As opposed to aggressively seeking out opponents to attack them, it plays more defensively; It does not move around the map as much as other agents, and therefore it collides with walls much less frequently than the more aggressive agents. The fuzzy agent spends most rounds keeping its distance firing precise shots at the enemy, and will only pursue enemies if they get close and then attempt to flee.

The scoring system of Xpilot is such that a player gains points for killing an opponent and loses points for dying. When killed by a player, the amount lost is equal to the amount gained by the other player. When a player gains a significant lead, they earn less points per kill, and when a player is significantly behind, they lose less points per kill. The score results from the first round are as such because Fuzzy and Sel rarely crashed into walls and were very closely matched in combat. Morton has very low negative scores because it is a bit reckless and crashes into obstacles often. Fuzzy's scores were very high in the tests against Cobra because Cobra rarely crashed into walls but was heavily outmatched in combat by Fuzzy.

6. CONCLUSIONS

We successfully designed and implemented a fuzzy control system for an agent in the Xpilot-AI space combat game. Although the results were less successful on initial testing than we had hoped, after some minor adjustments, the agent consistently outperformed the built in robot in Xpilot, as well as the sample Xpilot robot known as Morton. It also won four out of five games against Sel, an advanced rule based conroller. The agent was a success, and we have concluded that a fuzzy control system can be very successful in Xpilot.

In future research, it would be advantageous to develop a more aggressive agent, as while ours can out-survive and defeat other agents, it does not make a priority of chasing enemies down. In addition, several of the fuzzy parameters needed to be tweaked to attain our successful results; other future research will be to use evolutionary computation to learn an effective combination of specific membership sets and action activation ranges.

REFERENCES

1. Lee, Chuen Chien “Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I”, IEEE Transactions on Systems, Man and Cybernetics, Vol 20, 1990.
2. Shaout, A., King B., Reisner, L. “Real-Time Game Design of Pac-Man Using Fuzzy Logic,” The international Arab Journal of Information Technology, Vol. 3, No. 4, October 2006.
3. Li, y., Musilek, P., Wyard-Scott, L. “Fuzzy Logic in Agent-Based Game Design,” Fuzzy Information 2004. Proceedings of the IEEE Annual Meeting of the North American Fuzzy Information Processing Society 2004.
4. Parker, G., Parker, M. “The Evolution of Multi-Layered Neural Networks for the Control of Xpilot Agents,” The IEEE Symposium on Computational Intelligence and Games (CIG 2007).
5. Parker, G., Parker, M. “Evolving Parameters for Xpilot Combat Agents,” The IEEE Symposium on Computational Intelligence and Games (CIG 2007).