A Novel Hybrid Fuzzy A* Robot Navigation System for Target Pursuit and Obstacle Avoidance

Antony P. Gerdelan
Computer Science
Institute of Information and Mathematical Sciences
Massey University, Albany
New Zealand
E-mail:
Anthony.Gerdelan.1@uni.massey.ac.nz
antononthego@hotmail.com

Dr. Napoleon H. Reyes, Ph.D.
Computer Science
Institute of Information and Mathematical Sciences
Massey University, Albany
New Zealand
E-mail: n.h.reyes@massey.ac.nz

***Abstract*— Fuzzy Logic and the A* algorithm are two complementary systems that lend themselves amenable for target pursuit and obstacle avoidance. Fuzzy Logic is renowned for its ability to inculcate into a system a reactive and very refined robot movement, but is relatively inapplicable for long-term path planning. On the other hand, A* is very popular for its forward planning feature, but does not lend itself amenable for very fine robot movements as the algorithm tends to become exponentially computationally expensive. By integrating these two architectures into one Hybrid navigation system, we are able to combine their strengths and eliminate their weaknesses. This paper presents a Hybrid Fuzzy A* navigation system architecture that seamlessly integrates these two popular techniques, allowing for real-time pursuit and obstacle avoidance.**

I. INTRODUCTION

Experiments with robot navigation using potential fields [6, 7, 8, 9] and other methods for pursuit and evasion have shown moderately successful results, but when using these methods in real-world applications, we often observe cases where robots blunder into situations that cause them to become trapped - a result of the *greedy* (short-sighted) nature of these systems.

The framework presented in this paper uses a two-layered approach; at the top layer we have a planning module which calculates an intelligent *shortest-path* to the destination or the object being pursued, evading hostile agents, and avoiding *local maxima* - situations that would trap or delay a greedy system. At the bottom layer of the framework we have a reactionary module which refines robot movement; navigating smoothly toward the target, and around any immediate obstacles; controlling the speed and rotation of the robot. The top and bottom layers in this framework use a path-finding system based on the A* algorithm, and a Fuzzy Logic Control system, respectively.

II. THE PROBLEM DOMAIN

The game of robot soccer has provided an excellent test-bed for this research. Not only must a navigation system for soccer robots be effective, but in order to win it must also be *better* than the system employed by the competing team. Robot soccer thus stimulates research for improved navigation systems.

Robot soccer provides the additional challenges demanded by the high speed of the game; robots must *pursue* a dynamic target (the ball), avoid static obstacles and *evade* multiple competing robots. Robot soccer therefore demands a dynamic navigation system that operates in real-time, with only a small window of CPU time for calculation.

The problem domain for which this framework is intended consists of:

- a complex, three-dimensional environmental
- a dynamic, real-time environment
- a dynamic, moving target to pursue (the ball)
- static target locations (goal areas and field positions)
- multiple dynamic obstacles (all other robots)
- multiple hostile agents to evade (competing robots)
- static obstacles (walls and boundaries)

It is essential for any robot control system operating in the real world to remain in sync with the state of the robot. It is of little use to calculate a highly refined instruction for a robot, if, in the time of calculation, the robot or dynamic obstacles have moved so far that the instruction is no longer relevant.

A modern, competitive robot soccer control system operating in this problem domain must complete all calculation, including vision and other sensory calculation in less than 33ms. Therefore, a system implementing the framework described in this paper must strike a satisfactory balance between optimality and calculation speed.

III. GENERAL SYSTEM ARCHITECTURE

The architecture outlined in this paper is divided into two main layers of intelligent control; a planning layer, and a reactionary layer. *Figure 1* illustrates the entire framework, with additional modules for environment processing and actuator control interpolation.

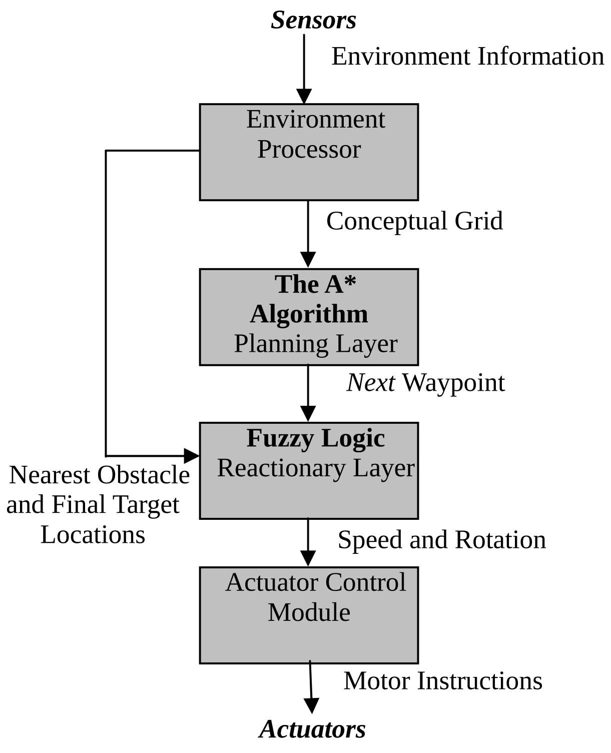

Fig 1 – Layered Framework Architecture

Environment information collected by the robotic system's sensors is analysed and key components are identified. A conceptual grid is created for the A* algorithm.

Processed environment information and the conceptual grid are passed to the planning layer, where a path to the goal of pursuit is calculated. The *next* point along the path is passed down to the Fuzzy Control layer as a target.

The reactionary Fuzzy Control Layer takes both the target information from the planning layer, and environment information detailing the nearest obstacle, and decides whether to engage in target-seeking or obstacle-avoidance behavior, or a combination of both. Fuzzy Outputs for speed and rotation components are defuzzified and sent to a module that decomposes outputs into actuator control (motor) instructions.

IV. THE ALGORITHMS

A. A Path-Finding for a Static Environment*

The A* algorithm [10, 11, 12] is regarded as the fastest optimal search algorithm, and for this reason has been used with success as a path-finding method for CPU-intensive computer games.

Computer games employing A* for path-finding [11] are generally *tile-based*. That is, the environment is built like a chess board – ideal for A*, as each board square or grid cell corresponds to a node in the search domain.

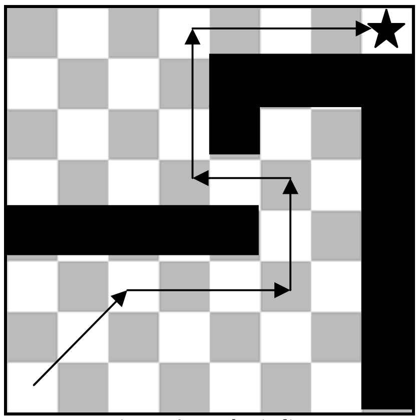

Fig 2 – A Path Finding*

Obstacles in the environment occupy entire grid cells, and these nodes can therefore be excluded from the search domain. Fig 2 illustrates a path calculated through an environment excluding obstructed nodes. Other nodes are awarded an *f-score* from the formula:

$$f^* = h^* + g^*$$

Where *h** represents the result of a heuristic function, for example, the Euclidean distance from the examined node to the goal node, and where *g** represents the result of a cumulative *cost* function, for example, the sum distance of traveling to that node along the path from the start node.

The algorithm builds a path to the goal node by selecting a string of nodes with the smallest *f-scores*.

The great strength of employing an A* search algorithm for path-finding is its ability to *think ahead*; producing a long-term plan for shortest-path movement toward a goal with a minimal amount calculation.

B. A Path-Finding for a Dynamic Environment*

For A path-finding to operate in a real, dynamic environment, a two-dimensional grid, representative of the real environment, must be generated. Grid cells must be of a size large enough to comfortably contain any of the obstacles. More complex grids with hexagons or other shapes for cells will create more detailed paths, but require more calculation than grids composed of squares.*

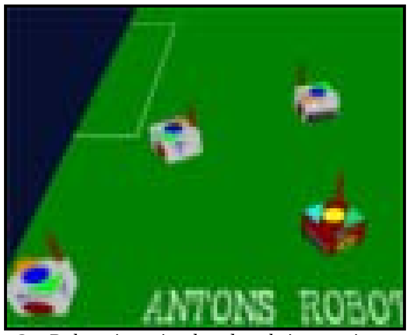

Fig 3 – Robots in a simulated real-time environment

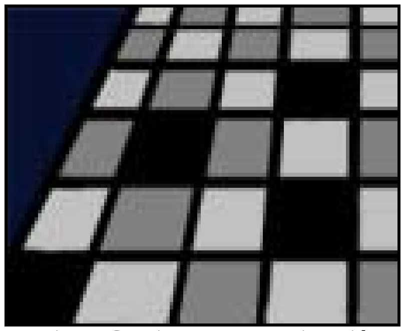

Fig 4 – Creating a representative grid

Referring to *Figs. 3 and 4*, we can see that the grid in *Fig. 4* has been created to represent the environment in *Fig. 3*. The four cells in *Fig. 4* that would contain obstacles from the complex environment have been excluded from the search domain.

A simple grid like the one in *Fig. 4* can be represented by a data structure as simple as a two-dimensional array or bit vector.

Once a simple grid representation has been created, the path-finding algorithm can operate on the grid, and translate the path into a series of *waypoints* for real-world navigation. This is analogous to how a driver would refer to a roadmap to plan a trip.

In addition, the grid must be re-generated regularly to take into account the movement of any dynamic obstacles. Dynamic A* path-finding differs dramatically from a static, pre-calculating system, in that the A* path must always exclude the *current* node, that is, the cell currently occupied by the robot agent can not be part of the path.

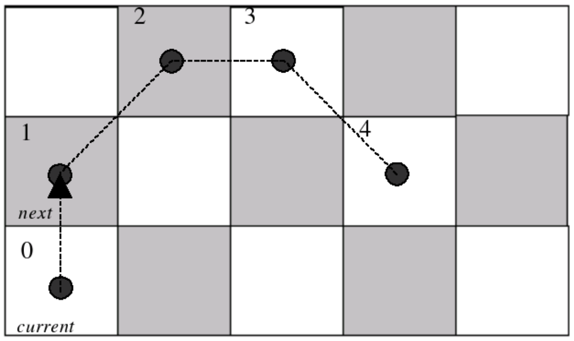

Fig 5 – Dynamic A Effective Path*

Referring to *figure 5* we can see that applying A* in a dynamic environment produces an optimal path, but because in a dynamic system we consider only the *next* point on the path (path point number 1) it is possible for an agent directly following the path to cut over grid cells containing obstacles.

Consider a robotic agent for which the set of waypoints in *figure 5* has been generated. Initially situated in grid cell 0, the agent would head towards the centre of cell 1. At the moment the agent crosses from cell 0 to cell 1, the calculated path has changed (refer to *figure 6)*.

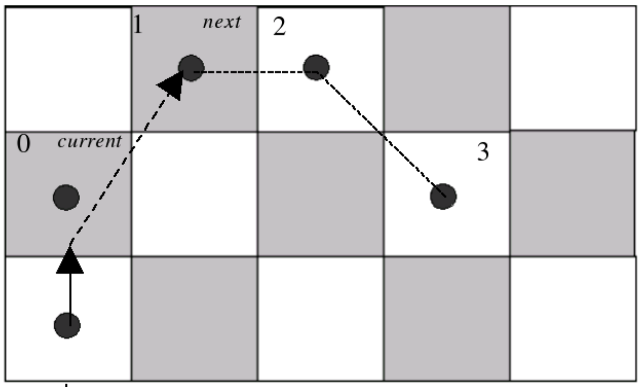

Fig 6 – Dynamic A effective path upon reaching next cell*

The square *now* occupied by the agent is said to be cell 0 and the next waypoint cell 1 (formerly cell 2 from *figure 5*). From the edge of the new cell 0, the agent now heads directly for the next cell. Note that the agent never arrived at the centre of the cell. In this way the effective path cuts corners.

The A* algorithm *alone* is therefore not suitable as a total robot navigation system, but requires an additional layer of refinement.

C. Fuzzy Logic Control System for Robot Navigation

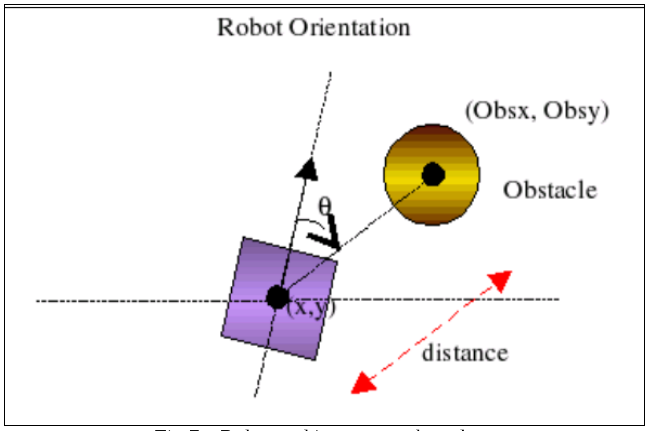

Fig 7 – Robot and its nearest obstacle

Fuzzy control systems for robot navigation [1, 2, 3, 4, 13] require input information about a robot's target; the distance between the robot and the target, and the angle between the robot's heading and the target. The fuzzy system will also require information about the *nearest* obstacle to the robot; again the distance and the angle (refer to *figure 7*).

The controller can then decide if it needs to switch between obstacle avoidance fuzzy rules and target seeking fuzzy rules.

Input information is *fuzzified* into fuzzy set memberships for *distance* and *angle* (to target or obstacle).

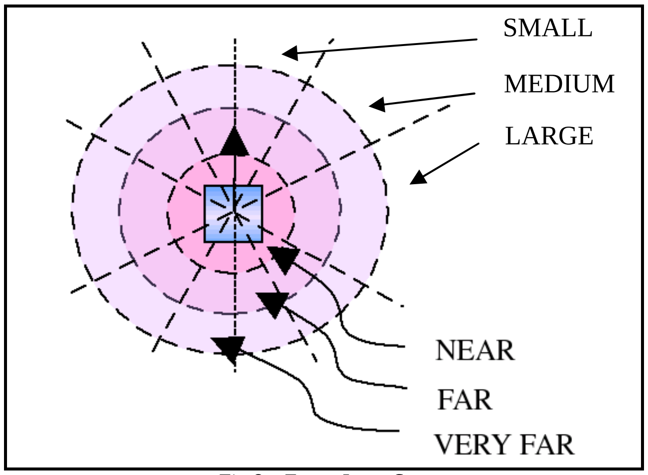

Fig 8 – Fuzzy Input Sets

Referring to *figure 8*, input distances can be classified as members (or partial members) of the fuzzy sets 'near', 'far', or 'very far', input angles as 'small', 'medium', or 'large'.

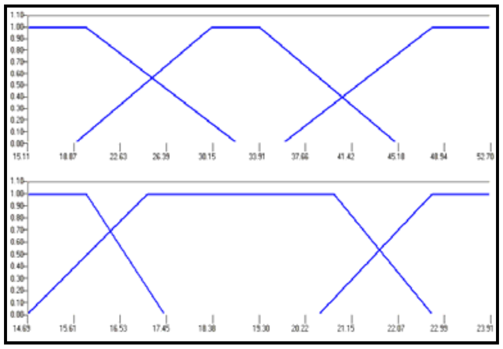

Fig 9 – Fuzzy Set Membership Functions for Target Angle and Distance

Trapezoidal fuzzy set membership functions to define memberships must be created for inputs to be fuzzified (*figure 9*).

Using fuzzified inputs, a set of fuzzy rules is matched to determine a collection of fuzzy outputs for both the speed and rotation of the robot.

As an example of a fuzzy rule, we can have:

If Obstacle is ***Near*** *and the Angle is* ***Small*** *(between robot and obstacle) then* ***Turn Very Sharp****.*

	NEAR	***FAR***	***VERY FAR***
SMALL	Very Sharp	Sharp Turn	Med Turn
MEDIUM	Sharp Turn	Med Turn	Mild Turn
LARGE	Med Turn	Mild Turn	Zero Turn

Fig 10 – Output rotation for obstacle distance vs. angle

	NEAR	***FAR***	***VERY FAR***
SMALL	Very Slow	Slow Speed	Fast Speed
MEDIUM	Slow Speed	Fast Speed	Very Fast
LARGE	Fast Speed	Very Fast	Top Speed

Fig 11 – Output speed for obstacle distance vs. Angle

Fig. 10 provides an example of a 3x3 Fuzzy Associative Memory Matrix (FAMM); matching fuzzy input sets for distance (columns) and angle (rows) to an obstacle, with an output set for the rotation component. *Figure 11* gives us the counterpart FAMM of fuzzy output sets for speed components.

Fuzzy output sets for speed and rotation are weighted together to produce one *defuzzified* value each using a *centre of gravity* function. The function is of the form:

$$\frac{m0 * w0 + m1 * w1 + ... + mn * wn}{w0 + w1 + ... + wn}$$

Where *m* is a 'mid-value' for fuzzy output set; for example 'Very Slow' might have a mid-value of 3km/h, and 'Top Speed' might have a mid-value of 15km/h, and where w is the weight value, between 0 and 1, of a particular fuzzy output set.

The defuzzified values are then given to a separate module that will interpolate the speed and rotation component into actuator or motor control instructions, specific to the type of vehicle being controlled.

IV. EXPERIMENTS

Experiments have been conducted in a real-time 3-D simulation environment to separately test the results of a fuzzy logic control system, and the A* algorithm. A simulation was also built to test a system using a layered fuzzy logic and A* approach.

Figure 12 illustrates the possible outputs for an agent seeking the target (marked with an 'X') from different positions in the environment. These outputs were produced from a simulation using only a fuzzy control system for navigation, and clearly indicate the change of rotation outputs from the fuzzy system.

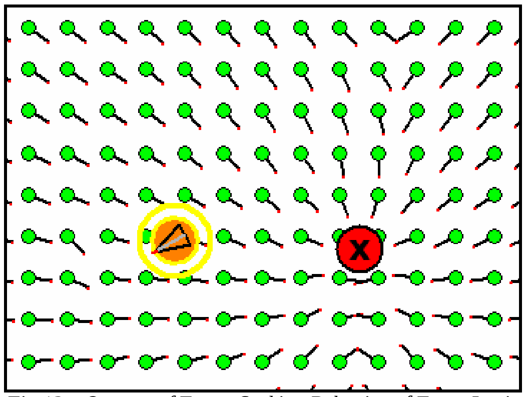
Fig 12 – Outputs of Target Seeking Behavior of Fuzzy Logic

Figure 13 illustrates outputs from a combined system, where the dark coloured robot is trying to navigate from the bottom of the figure to the top, avoiding all of the other robots. The six small dot objects indicate the path created by the A* algorithm (which have been connected by arrows for clarity). We can see that as the robot moves through the third, fourth, and fifth waypoints, it will need the fuzzy system to refine its path of movement.

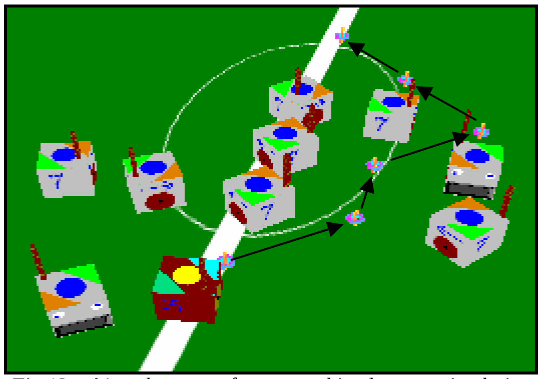
Fig 13 – A path outputs from a combined system simulation*

It is worth noting that some refinement of the fuzzy obstacle avoidance system is necessary to prevent disagreements between the fuzzy layer and the A* layer. For example, it was discovered during development of the simulation pictured in *figure 13* that although the A* algorithm calculated a path of movement diagonally between obstacles (such as the path between the third and fourth points in *figure 13*), the fuzzy system detected the nearby obstacles and would try to avoid them, overcompensating and preventing movement through the gap. Some tweaking of fuzzy set membership functions is necessary to remedy such disagreements.

V. FUTURE WORKS

A. Multiple Agent Coordination Using A Nodes*

When planning paths for a cooperative team of robots, it is possible to re-enforce the concept of a *dominating robot* [3], that is, where one robot has priority of movement over the others by recording the waypoints of this robot and the estimated time of arrival at those points, in the nodes of the conceptual grid used to calculate them. Subsequent robots to plan paths can then avoid choosing waypoints that coincide with those of the dominant robot, and thereby avoid collision.

To extend this concept, each robot in the team can be given a priority, the robots with higher priority calculate their paths before those with lower priority, and store the waypoints in the conceptual grid, thereby allowing the entire team to plan paths that will not result in collisions or competition with robots from the same team.

B. Applicability to an All-Terrain System

Provided with enough information about the environment, the architecture outlined in this paper is suitable for an all-terrain autonomous vehicle.

The *g** scores calculated for different cells could easily be adjusted to reflect estimated costs for moving across various identifiable types of terrain, thereby allowing a robot to avoid rough areas in favour of more quickly passable terrain.

*G** scores can also reflect the increased or decreased cost of moving up or down sloped terrain.

Major challenges to applying this framework to an all-terrain system will not come from expanding the framework, but in making sure the different layers are provided with accurate and simple information about the environment.

Environmental information could come in the form of specially prepared or interpreted maps, or from improved machine vision techniques to identify impassable objects. GPS information may also be useful for accurately determining robot location.

VI. CONCLUSION

This paper successfully integrates Fuzzy Logic and the A* algorithm for target pursuit and obstacle avoidance in a real-time 3-D simulation test-bed for the robot soccer game. Empirical results demonstrate that the Hybrid Fuzzy A* navigation system is able to take advantage of A*'s forward planning feature, and Fuzzy Logic's very fine reactive robot movement, allowing the robots to seek its target while avoiding the obstacles simultaneously.

REFERENCES

[1] J. Baltes, N. Hildreth, and Y. M. Lin: *The all botz robocup team. In Proceedings of the PRICAI Workshop on RoboCup, Singapore*, November 1998.

[2] A.Bonari, G.Invernizzi, T.Halva Labella, M.Matteucci, An architecture to coordinate fuzzy behaviors to control an autonomous robot, *Fuzzy Sets and Systems*, (134): 101-1 15,2003

[3] Q..Meng, X. Zhuang, C. Zhou, J. Xiong, Y. Wang, T. Wang, and B. Yin, Game Strategy Based on Fuzzy Logic for Soccer Robots, *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*,vol.5, pp.3758-3763, 2000.

[4] Oller, A., de la Rosa J. Ll., García, R., Ramon, J.A., and Figueras, A., “Micro-robots playing soccer games: a real implementation based on a multi-agent decision-making structure” *Intelligent Automation and Soft Computing*, in press, 1999.

[5] C.C.Wong, M.F.Chou, C.P.Hwang, C.H.Tsai, and S.R.Shyu, A method for obstacle avoidance and shooting action of the robot soccer, *Proceedings of the 2001 IEEE International Conference on Robotics & Automation*, 2001, pp. 3778-3782.

[6] W.S. Newman, and N. Hogan, "High Speed Robot Control and Obstacle Avoidance Using Dynamic Potential Functions." Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, March 31-April 3, 1987, pp. 14-24.

[7] Koren, Y. and Borenstein, J., "Critical Analysis of Potential Field Methods for Mobile Robot Obstacle Avoidance." Submitted for publication in the IEEE Journal of Robotics and Automation, July 1990.

[8] Y. Koren and J. Borenstein, “Potential field methods and their inherent limitations for mobile robot navigation,” in Proc. IEEE Conf. Robotics and Automation, Sacramento, CA, Apr. 7–12, 1991, pp. 1398–1404.

[9] S. S. Ge and Y. J. Cui, “New potential functions for mobile robot path planning,” IEEE Trans. Robotics Automat., vol. 16, pp. 615–620, Oct. 2000.

[10] Peter Yap, "Grid-Based Path-Finding", Advances in Artificial Intelligence: 15th Conference of the Canadian Society for Computational Studies of Intelligence, AI 2002 Calgary, Canada, May 27-29, 2002. Proceedings Volume 2338 / 2002 Chapter: p. 44

[11] C. Reynolds, “Steering Behaviors for Autonomous Characters,” Proc. Game Developers Conf., 1999.

[12] Botea, A., M¨uller, M., & Schaeffer, J. (2004). Near Optimal Hierarchical Path-Finding. Journal of Game Development, 1(1), 7–28.

[13] C.L.Hwang, N.W.Chang, S.Y.Han, A Fuzzy Decentralized Sliding-Mode Control for Car-Like Mobile Robots in a Distributed Sensor-Network Space, Presented, Third International Conference on Computational Intelligence, Robotics and Autonomous Systems 13 - 16 December 2005.