Towards a Generalised Hybrid Path-Planning and Motion Control System with Auto-calibration for Animated Characters in 3D Environments

Antony P. Gerdelan and Napoleon H. Reyes

Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand
gerdelan@massey.ac.nz, n.h.reyes@massey.ac.nz

Abstract. Intelligent navigation and path-finding for computer-animated characters in graphical 3D environments is a major design challenge facing programmers of simulations, games, and cinematic productions. Designing agents for computer-animated characters that are required to both move intelligently around obstacles in the environment, and do so in a psycho-visually realistic way with smooth motion is often a too-difficult challenge - designers generally sacrifice intelligent navigation for realistic movement or vice-versa. We present here a specially adapted hybrid fuzzy A* algorithm as a viable solution to meet both of these challenges simultaneously. We discuss the application of this algorithm to animated characters and outline our proposed architecture for automatic tuning of this system.

1 Introduction

Our previous works have focused on the development of next generation navigation algorithms for soccer robots by combining existing navigation and motion control algorithms into hybrids which exploit the benefits of the individual algorithms but also negate their limitations. To this end we have developed the Hybrid Fuzzy-A* [1] algorithm which can perform both efficient and dynamic route planning in environments with static and moving obstacles, and also move using smooth-path motion and avoid obstacles at very high speeds, requiring only a very small allocation of CPU resources as robot soccer teams are typically controlled by one commodity desktop computer which also has to process computer vision and direct 3-11 other robots every 1/30th of a second. Several variants of this hybrid were created which enabled us to incorporate other properties into robot behaviour [2] [3] - in particular, an ability for the ball-carrying robot to *evade* hostile robots on its path to the opposition goal and pre-empt their path of movement as well [4]. The success of these works, and also the impressive appearance of this algorithm operating in 3D graphical simulation (see Figure 1), lead us to speculate whether the algorithm could be applied to a

M. Köppen et al. (Eds.): ICONIP 2008, Part I, LNCS 5506, pp. 1079–1086, 2009.

Fig. 1. Our 3D robot soccer simulator with robots controlled by the Fuzzy A* algorithm. The darker coloured robot at the top of the figure plans a path through moving robots. Blobs indicate the points along the *current* plan for the robot's path.

Fig. 2. An animated character - in this case a to-scale model of a Soviet T-28 circa 1932 with simulated real physical operating characteristics - travels through a 3D-environment in our simulation engine complete with hills and a range of obstacles (trees and houses)

range of much more demanding applications beyond the mathematically simple 2d realm of soccer robots.

We have constructed a real-time simulation engine using the OGRE3D graphics library [5], which has been used successfully for military visualisation and simulation projects [6] with success, to mimic a typical computer game-type environment as a proving ground for the application of hybrid robot soccer control algorithms to much more complex 3D environments. This type of environment also typically has limited CPU available for artificial intelligence computation as 3D graphics processing tends to occupy the bulk of resources [7] so the need for efficient algorithms is paramount, and can process sensory information with large amounts of environment noise or complexity [8] - the kind of environments where Fuzzy Logic is at a premium.

The Hybrid Fuzzy A* algorithm has been adapted so that it is scalable and can be applied to a range of animated characters with their own specific motion characteristics. Our initial implementation has been successful with vehicle-type animated characters - see Figure 2. As depicted in the figure, the vehicle has planned a prospective path around obstacles using the A* component of the Hybrid Fuzzy A* algorithm (as indicated by the segmented line visibile ahead of the vehicle in the figure) and has detected, and is driving around the *nearest obstacle* on its right hand side (as indicated by the second line visible in the Figure) - in this case a house.

Due to the complexity of manually tuning the many algorithm parameters for optimal results we have also designed an architecture for automatic parameter calibration so that the system can self-train for application to a range of new characters and environments by operating on a series of *obstacle courses* - typical operating environments created by a designer for training - and self-evaluating.

We also detail here some new visualisation techniques, which we have found necessary to develop, for visibility of complex multi-level artificial intelligence systems in real-time applications; as post-run analysis of animated character motion is an almost impossible task.

2 Hybrid Algorithm

The Hybrid Fuzzy A* algorithm is arranged in a cascade of systems. At the top level, we give the character some information about the obstacles environment and it builds an *environment obstacle map*. The environments that we are considering so far can be simplified and expressed as a 2-dimensional map and are stored as a 2D array of values. More complex path-planning might require a 3D array of values. We are using a character array to store our obstacle map, as this allows us to express a range of obstacle types with single letters ('*s*' for shrubs, '*b*' for buildings). A requirement of the system is that the character has some ability to divide the environment into a graph or searchable area. The character then translates known obstacle positions into graph indexes and marks the locations. In an environment with many dynamic (moving) obstacles we recompute this map with every frame of calculation. Larger obstacles can occupy multiple graph cells. The character will ultimately make use of several types of *environment map* as weighted search domains for a *depth-limited dynamic A* algorithm* (with depth limited to scale to available CPU window per application) to compute a a near-optimal, partially complete path to its target destination. Graph nodes marked as containing obstacles will be either excluded from the search domain (impassable obstacles) or given a weight based on their resistance to the character's movement (for example shrubs may be given a moderate weight modifier for heavy vehicles in a military simulation or game, as they can be driven over, but buildings may be impassable and excluded from the search domain). This path is broken into *waypoints* and the first or an early-stage waypoint (if the graph resolution is very high) is used as the *current waypoint* and given to the Fuzzy navigation system as *target location crisp input*. Our experience with robot soccer path-planning [4] has shown us that the A* algorithm's heuristic can be modified to incorporate many different weighted inputs into the path-planning decision process.

$$f^* = h^* + u^* + g^* \qquad \text{scoring function with undesirability feature (u*)} \quad (1)$$

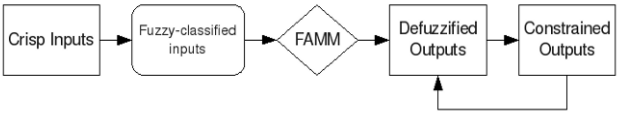

Fig. 3. Scalable Fuzzy Navigation cascade with feedback loop to adjust instructions for a character's known physical limitations

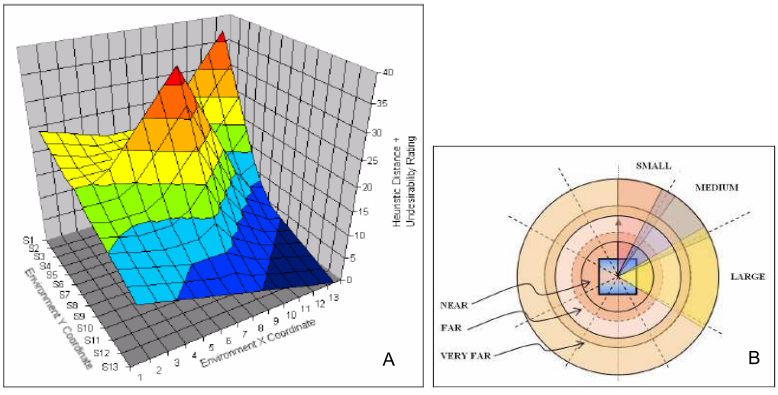

Fig. 4. (A) Representing combined heuristic weights of cells in the environment graph. The A* algorithm will choose the most overall down-sloping path (minimising overall cost + balanced heuristic). (B) Classifying environment elements into overlapping fuzzy sets representing angles and distances.

Eq.(1) depicts the modified scoring function. The ranges of u* (undesirability) must be carefully balanced with h* values (heuristic distance from starting node to goal node). On the other hand, the g* component (actual cumulative distance) ensures an overall steepest downhill (shortest) path. This generates the *heat environment map* (Figure 4, A). It influences the selection of paths that are slightly longer but safer; inculcating an evasive path planning property. Factors such as slope of terrain, and known condition of ground surface can be weighted as proportions of the heuristic. Prudent designer decisions are required here to choose heuristics that are meaningful in a particular 3D environment, although the best ratio of weights of each input could be determined by self-training simulation runs. These heuristics may be tailored to a particular vehicle or re-evaluated periodically to adapt to changing conditions; reflecting the value of a true dynamic A*, as opposed to a less flexible variant such as D* [9]. The balancing of heuristic weights is deemed delicate.

The Fuzzy navigation layer is a rapid calculation system that takes the *current waypoint* on the path created by the A* layer as a rough guide (Figure 6). The systems can disagree - for example if the A*-generated path dictates to move left around a tree, but the vehicle is actually already moving around it to the right - then the path will be recalculated in the next frame of calculation. The fuzzy system consider two groups of crisp input - the *distance and angle to the nearest obstacle* and the *distance and angle to the current waypoint* see Figure 4 (B) for an illustration of this classification. These two groups of inputs are fed into distinct fuzzy set membership functions (only 3 sets per input) and fuzzy associative memory matrices. We are taking advantage of angle symmetry, and therefore, the rules are generalised - the antecedents do not explicitly state whether the obstacle/target is on the left or right [4]. In effect, using 3 fuzzy sets (9 rules) for half of the angle space (right-half) (Fig. 4, B) gives the same accuracy as using 6 fuzzy sets. The fuzzified outputs for obstacle avoidance and

target seeking behaviour are then be blended together using a centre of gravity function to produce smooth transitions between the two behaviours. Take note that this reactionary layers are controlling the steering angle and speed for both target pursuit and obstacle avoidance(Figure 6). On the other hand, the evasive behaviour is inculcated by the modifed A* scoring function Eq.(1).

Navigation outputs are, in our adapted hybrid algorithm, expressed as *ratios* of character properties. Steering is also expressed as ratio of the speed control outputs. In this way the algorithm should produce motion that is both scalable to a character's motion limitations and applicable to a range of characters. We have also extended the Fuzzy A* algorithm with a feedback loop (see Figure 3). This allows us quickly re-scale our defuzzified output ratios should the indended output ratios (for example a vehicle can not steer as sharply as desired at its current speed) exceed a character's physical limitations.

3 Visualisation

Figure 5 is a screenshot from the 3D graphical simulation that we have created as a proving ground for our algorithms. Obstacles of various sizes are strewn about the 3D landscape (houses and trees) and several monitoring panels overlay the display. Because of the level of complexity and number of overlapping systems involved in real-time hybrid algorithms, it is a great aid to have a system in place for visualising these different processes in real-time so that the designer can monitor the system as it happens with a full range of information. When an interesting case does seem to occur in the behaviour of the character - for instance, we observed that our character was speeding up and slowing down even when travelling in a straight line - as illustrated by the sawtooth pattern in the 3rd graph down on the left in Figure 5. When this behaviour was observable we were able to pause the simulation time so that the character stopped moving mid-sequence. We observed the saw-tooth velocity graph and have a system in place whereby we were able to export the dataplots for all of these graphs to a range of formats for closer inspection in external programmes, where we could see that the *current waypoint* being assigned to the Fuzzy control system was always too close to the vehicle due to the high resolution of the path-finding system.

4 Proposed Self-training Architecture

Given the large number of parameters that need to be calibrated for animated character control, and in particular for hybrid algorithms, it has been proposed that various automatic training systems be employed for this task; genetic algorithms have been explored for robots soccer agent training [10], and evolving neural networks have been used to improve and even generate entire behaviours for animated characters [11] [12] with some success. We intend to adopt a subset of these ideas to self-train our characters. We propose an obstacle-course based *survival of the fittest* paradigm.

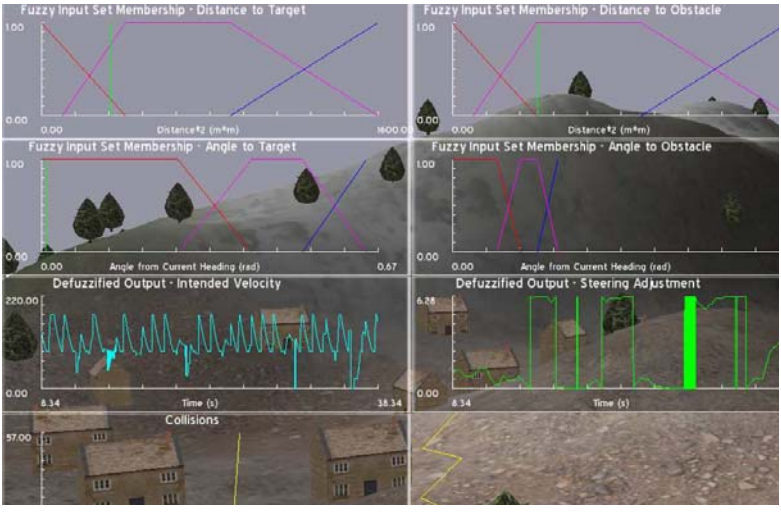

Fig. 5. Graphing navigation data in real time

A range of obstacle courses can be quickly created with scenario designers, such as the real-time designer that we have built into our simulation engine. These courses can be quickly designed as test cases for the sort of 3D environments that the characters would be expected to operate when finally deployed; for example we might design a course or scenario consisting of flat landscape and large streets of house-like obstacles, and another course consisting of hills and valleys with sporadic trees to avoid for training an animated soldier character for a war scene in a film production.

The advantages of this kind of automatic training system would be:

- Initial characters start from some adequate basis, rather than from a zero-skill base
- Can train a large range of simulated characters or even simulations for real vehicles.
- Can adapt itself to new environments - e.g. we can hand-craft a new scenario that the character should have to be able to cope with and it can improve it's system to deal with this new environment as well.
- Could be set to self-improve in real-time during real execution over a time-slice basis, rather than a per-individual character basis.

If we arrange a batch of simulation runs where we randomly adjust the parameters of a character by some threshold, then start the character at a range of pseudo-random start locations and orientations, and ask for them to move to some fixed destination over a number of runs, then we can also ask the character to evaluate its performance based on a fitness function that generates a fitness score based on *completion time* and a metric determining number and severity of *collisions* (based on seconds of simulation time * distance inside obstacle bounds). You can see in Figure 5 that we are already extracting these parameters

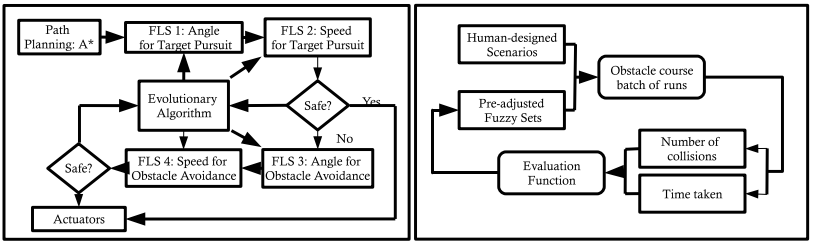

Fig. 6. The Hybrid Fuzzy A* - Evolutionary Algorithm (left) and the proposed algorithm for self-improving animated character navigation (right)

per character evaluation. Using a *generation* of our character, each using slightly different modifications of parameters, we can *select* the best two members of the generation (with the most minimal fitness scores) to create the base parameters for the next generation. This process is illustrated in Fig.6. We could also distribute this system over a network of commodity-level machines for very rapid mass training. We have infrastructure in our simulation engine in place for this sort of distributed computing an intend to carry out a full investigation of this training architecture in works in the near future.

5 Conclusions

We have presented a snapshot of our progress adapting hybrid robot navigation algorithms for animated characters in 3D environments, introducing adaptations that we have had to make to the system for application to more complex 3D environments, our work towards completing a generalised version of this system that can be applied to a range of autonomous animated characters, and illustrated some visualisation techniques that we have found particularly relevant to developing these sorts of real-time systems. We seek to expand on this work by automating the calibration procedures for the hybrid algorithms dictating path-planning and motion control for these characters, and have in this paper introduced our proposed architecture for this system.

References

1. Gerdelan, A.P., Reyes, N.H.: A novel hybrid fuzzy a* robot navigation system for target pursuit and obstacle avoidance. In: Proceedings of the First Korean-New Zealand Joint Workshop on Advance of Computational Intelligence Methods and Applications, Auckland, New Zealand, vol. 1, pp. 75–79 (2006)
2. Gerdelan, A.P.: Artificial Intelligence in Robot Soccer. Bachelor of engineering honours year project report, Institute of Information and Mathematical Sciences, Massey University, Albany, New Zealand (October 2006)
3. Gerdelan, A.P., Iskandar, D., Djohar, A.F., Reyes, N.H.: Utilising the hybrid fuzzy a* algorithm in a cooperative multi-agent system. In: Conference Program and Abstracts of the 4th Conference on Neuro-Computing and Evolving Intelligence (NCEI 2006) and 6th International Conference on Hybrid Intelligent Systems, HIS 2006 (2006)

4. Gerdelan, A.P., Reyes, N.H.: Synthesizing Adaptive Navigational Robot Behaviours using a Hybrid Fuzzy A* Approach. In: Advances in Soft Computing: Computational Intelligence: Theory and Applications, pp. 699–710. Springer, Heidelberg (2006)
5. Junker, G.: Pro OGRE 3D Programming. APress (2006); ISBN 1590597109
6. Lawes, G., Barlow, M.: Visual realism and decision making: A novel approach to real-time maritime battlespace visualisation. In: SimTecT 2007 Conference Proceedings. Virtual Environment and Simulation Laboratory (VESL), University of New South Wales at the Australian Defence Force Academy, Simulation Industry Association of Australia (2007); ISBN:0 9775257 2 4
7. Barron, T.: Strategy Game Programming With DirectX 9.0. Wordware (2003); ISBN 1-55622-922-4
8. Funge, J.D.: Artifcial Intelligence for Computer Games. A K Peters, Ltd., Wellesley (2004)
9. Choset, H.M., Hutchinson, S., Lynch, K.M., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion: Theory, Algorithms, and Implementation. MIT Press, Cambridge (2005)
10. Messom, C.H.: Genetic algorithms for autotuning mobile robot motion control. Res. Lett. Inf. Math. Sci. (3), 129–134 (2002); ISSN 1175-2777
11. Stanley, K.O., Bryant, B.D., Miikkulainen, R.: Real-time neuroevolution in the nero video game. IEEE Transactions on Evolutionary Computation 9(6), 653–668 (2005)
12. Miikkulainen, R.: Creating intelligent agents in games. The Bridge 36(4), 5–13 (2006)