

# Driving Intelligence: A New Approach for Smart, Efficient Vehicle Simulation

A.P. Gerdelan

Complex Systems and Simulations Group (CSSG)  
Institute of Information and Mathematical Sciences  
Massey University, North Shore 102-904, Auckland, New Zealand  
Email: gerdelan@gmail.com  
Tel: +64 9 414 0800 Fax: +64 9 441 8181

April 2009

## Abstract

The new wave of computer-driven entertainment technology throws audiences and game players into massive virtual worlds where entire cities are rendered in real time. Computer animated characters run through inner-city streets teeming with pedestrians, get into and out of cars, and drive through rush-hour traffic; all fully rendered with 3D graphics, animations, particle effects and linked to 3D sound effects to produce more realistic and immersive computer-hosted entertainment experiences than ever before. No modern urban environment is complete without realistic simulated road traffic, however traffic simulation has not kept pace with the entertainment industry - modern traffic simulations simply do not reproduce convincing individual driver intelligence to the level required by interactive entertainment. We present a new paradigm for agent-driven traffic simulation, specifically designed for such applications. Our agent architecture utilises a new hybrid algorithm for dynamic road navigation, implicitly facilitates rational overtaking behaviour, and produces realistic smooth-motion vehicle control.

**Keywords:** 3D graphics; traffic simulation; agents; virtual reality; video game AI; fuzzy logic.

## Acknowledgements

Thanks to the Graphics, Vision, and Visualisation Group (GV2) of Computer Science at Trinity College Dublin with the Science Foundation Ireland for hosting and supporting this work, Jonathan Ruttle who detailed for me his work in

this area, and to IIMS at Massey University in New Zealand for supporting my PhD research.



Figure 1: The system in operation: real-time congested traffic simulation for Dame Street, Dublin, Ireland.

## 1 Introduction

In our most recent work we outlined a new system for designing and visually *laying* a road network within an existing model or simulation [1]. The work described in this paper is Part 2 - the architecture and navigation system for the agents that will operate on such a road network.

In our previous work, we introduced a rapid system for manually laying road networks within existing simulations. The roads created in this system had to support vehicle-driving agents that could integrate into the complex environments of modern visualisations and simulations. Typical challenges of integration into these simulations are:

- To-scale models of urban landscapes already exist
- Geometrically large urban areas are simulated
- Must run in real-time
- Interaction with existing 3D systems (world models, pedestrians)
- CPU and GPU constraints with existing systems
- Simulated 3D vehicles, with real performance characteristics must be supported (see Figure 2)
- Must be realistic at a range of camera angles and vantage points (car-level, pedestrian-level, birds-eye, etc.)

To tackle all of these problems we build a simplified digital representation of the road network; free of noise and complexity, such that our vehicle-controlling agents can use it as a guide in the same way that one might refer to a street map when driving through a new city. Because the 3D environment already exists, we need to create our digital road map as a post-process, and our technique is to marry it to the existing environment in such a manner so that it does not interfere with the existing work.

In our previous paper [1] we have analysed the major traffic simulation models [2–10] and find that they are based on a *car-following* formula as the primary system input to each vehicle-driving agent. Equations 1, 2, and 3 give us Gipps’ [6] traffic model - which dictates the *velocity* of a following vehicle  $v^f$  as a combination of an *acceleration rate*  $v^a$  and a *braking rate*  $v^b$  based on a *distance of separation* behind the lead vehicle  $x_t^l - x_t^f$ .



Figure 2: A test vehicle that we have created for our traffic simulation - an Enviro 400 Dublin Bus.

$$v_{t+T}^f = \min [v_{t+T}^a, v_{t+T}^b] \quad (1)$$

$$v_{t+T}^a = v_T^f + 2.5aT \left[ 1 - \frac{v_t^f}{V_n} \right] \sqrt{0.025 + \frac{v_t^f}{V_n}} \quad (2)$$

$$v_{t+T}^b = b * T + \sqrt{b^2 T^2 - b \left\{ 2 \left[ x_t^l - x_t^f - S_{jam} \right] - v_t^f T - \frac{(v_t^l)^2}{b^*} \right\}} \quad (3)$$

We have taken a different approach, and conduct a graph search of the road network; primarily using a list of way-points as input. We take other traffic into account as dynamic obstacles - car following in our model is a secondary, **emergent** property. An advantage of this approach is that lane-changing behaviour is dictated by a rational agent decision, rather than as the product of a pseudo-random formula, as it is in the major models. As our simulation is to be viewed at a range of levels of detail we aim to produce a more convincing behaviour model for individual vehicles first, and a traffic flow simulation second.

## 2 Agent Paradigm

The agents that we have designed operate with a *Belief-Desire-Intention* [11] intelligent agent model, and are based on a simplified version of our previous stack architecture for robot-controlling agents [12].

### 2.1 Belief

Whilst other simulation developers have endeavoured to produce realistic models of human audio-visual perception, for the sake of simplicity we have opted to cheat; providing the agents’ *belief* or perceptual input with hard, generalised data. Our agents are aware of all of the other vehicles and static and dynamic obstacles in their proximity. We also broadly assume that our agents have knowledge of local road congestion heuristics; representing knowledge of which roads to avoid at peak hour. We simply provide all of our agents with the same network-weight information, and have not investigated a driver route-planning model that incorporates a psychology model. Adding error to this information, or utilising real congestion and driver-behaviour data from city planners may be a subject for focus in future works.

We regularly ascribe the following specific information to each simulated agent, constituting its crude perception of the world:

- The relative location of the nearest moving or stationary obstacle (whichever is closer)
- The relative location of the next point along the path that the agent has planned for itself
- Local road network linkage information (a city map)
- Local road congestion heuristics (an estimate of congestion on each road)
- Occupancy of nearby road lanes

## 2.2 Desire

All of our agents are given a generalised goal, or list of goals upon entry into the simulation. The first case that we consider is that of city buses operating set routes through the city. Our bus-driving agents are given a list of way-points, stops, and time-table information, and then the agent is left with the task of driving through traffic between these points, making the appropriate stops, and adhering to a time-table. Private cars have even more generalised goals - they simply have a destination - the agent must contrive its own plan for navigation through traffic in a dynamic fashion.

## 2.3 Intention

The *intention* component of the agent model comprises several key functions;

1. Using *belief* information to form a multi-stage plan to achieve *desired* goals.
2. Working out specific actions to take immediately to move to the next stage of the plan.
3. Attempting to affect those actions

In the case of our car-driving agents, a specific example of this process is:

1. Using road-network information to find a short path to our destination.
2. Decide if we need to change our steering and acceleration such that we follow the road lane and avoid crashing into anything.
3. Adjust steering and braking/acceleration.

## 3 Agent Architecture

Our architecture stack is illustrated in Figure 3. The architecture is broken into several layers of processing. This is a typical *stack* BDI agent architecture, that we have shown

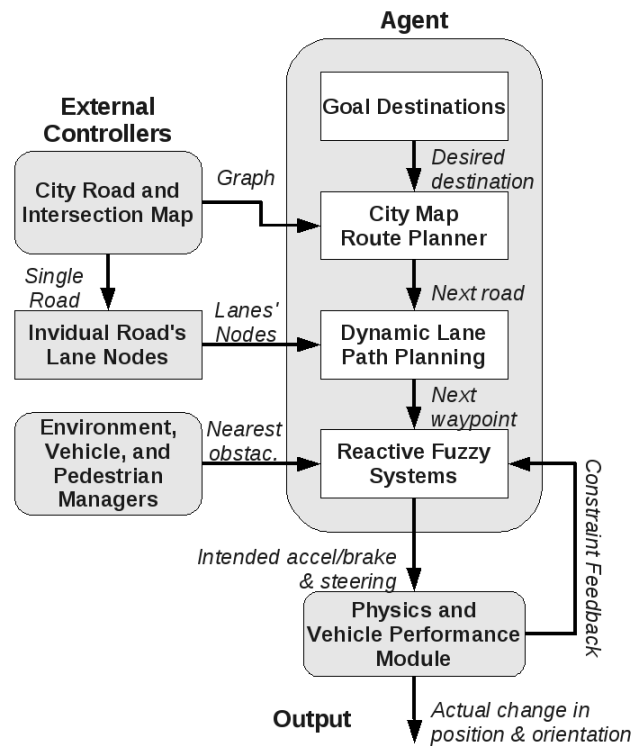


Figure 3: Architecture of the agents used for driving vehicles. We see the main modules representing belief, desire, and intention - input from external controllers, “Goal destinations”, and output from fuzzy systems, respectively.

is both efficient and successful for real-time agents using hybrid architectures in previous works [13–16]. Important to note about our design, is that the *desire* component of the agent, in this case the Reactive Fuzzy Systems output does not necessarily translate 1 : 1 into the vehicle’s actual movement in the world. Because we want to use this architecture as a generalised *agent middleware*, supporting a large range of vehicles with different performance characteristics, we provide an additional output module, which governs the actual output based on the physics of the environment, and the performance characteristics of a particular vehicle.

Any *intelligent* agent worthy of its name has the ability to evaluate the output of its own actions [17]. In order to facilitate this behaviour, we provide a feedback loop - this lets the Fuzzy Controller module know how much its intended outputs have been constrained by physical limitations, and it can scale back its instructions to suit. The entire stack is generally updated with every frame, however to support a very large number of agents operating on one CPU simultaneously, our initial testing indicates that throttling back the dynamic path-planning module to  $6Hz$ , and the reactive module to no less than  $30Hz$  provides an effective minimum threshold of updates. Any less frequent updates and approximated curved path-following behaviour becomes highly erratic. A study to confirm the ef-

fects of time-slice update throttling on artificial intelligence and movement by vehicle-controlling agents is a study of an ongoing work [18].

### 3.1 Path Planning

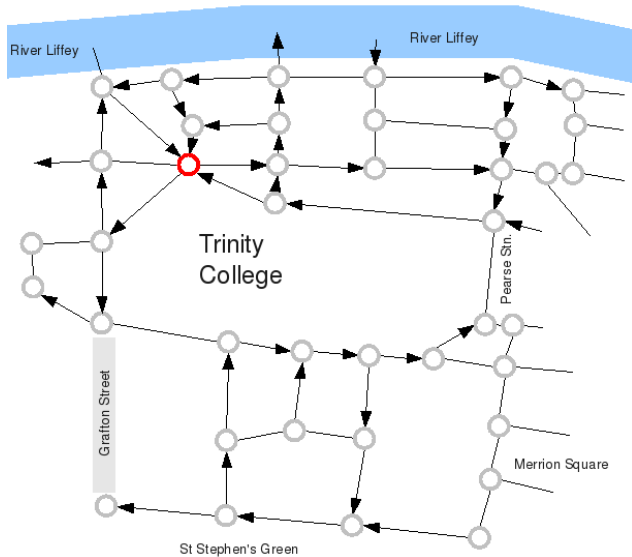


Figure 4: Representation of an automatically generated intersection “city map” for the area around Trinity College Dublin, Ireland. Only a very small number of nodes are required for the search domain of a city route plan.

The two layers of path-planning in our system both conduct fast searches of roads represented as nodular networks. The higher-level module takes an agent’s desired destination and determines a path through a city, making use of a *pre-computed graph* that we have generated from our rapid road-laying tool (see for details). This graph is purely made up of road intersection points, and as such is a very small graph to search (see Figure 4). There are several search method options for this graph:

1. a static route look-up, using pre-computed paths
2. pre-defining routes to every destination, updated dynamically by *the Road Manager* using a routing algorithm
3. a dynamic heuristic-based search (during driving) to reflect changing conditions such as congestion.

In extremely large simulations, or those where the time period simulated does not allow for changing conditions, then a pre-computed route model may suffice. Propagating congestion information around a city map using a routing algorithm would require a powerful central server, but may make for an interesting case study for urban planning. We have chosen, however, to use a standard approach and do a

dynamic heuristic search of the road map at the commencement of travel (any time that an agent’s goal destination changes). To reflect changing congestion conditions it is also possible to recompute this route during travel, but we have not investigated progressive traffic pattern change simulations at this stage, although a possible subject of future work may be to demonstrate changing patterns of traffic at peak hours - where cars take alternative routes. We have used the ubiquitous  $A^*$  algorithm [19] for this task.

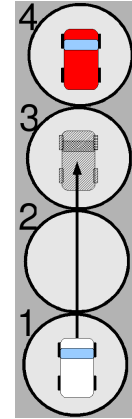


Figure 5: Lane occupancy and following distance model; the white car maintains following distance behind a leading car on a section of continuous road. Circles represent our nodular break-down of the road lane. In inner-city operation following cars aim to come to a complete halt at the last unoccupied node along their route. In high-speed operation vehicles store time-stamp each node as they exit it to emulate human following time estimation.

The lower-layer path finding is dynamic, and operates *within individual road lanes*. Our road lane laying system automatically breaks lanes into vehicle-sized *nodes*. These are both searchable, and also aid car following separation and traffic queueing. Single lanes are simply list-traversal operations, where nodes that are *occupied* temporarily halt the generated path (see Figure 5 for illustration). Multi-lane roads facilitate emergent lane-changing behaviour. Given a small heuristic cost for changing lanes, we can stimulate our agents to spread evenly over multiple road lanes, or for faster vehicles to decide to overtake slower leading vehicles. In this way we completely avoid designing or employing a complex, and clockwork mathematical formula for representing lane-changing behaviour as is widely employed for traffic simulation models.

### 3.2 Reactive Vehicle Control

We are using a fuzzy logic model for controlling the steering and acceleration behaviour of our vehicles. Fuzzy logic has been employed by other traffic simulations for its flexible nature - easily expanding to accommodate more complex input variables into the mix [20]. Initial studies have

Real Term	Rough Set Value Range	Fuzzy Term
Wide Arc	$> 22.5^\circ$	WID
Mid-range Arc	$0 - 33.75^\circ$	MID
Narrow Arc	$0 - 22.5^\circ$	NAR
Far Distance	$> 20m$	FAR
Medium Distance	$0 - 28.28m$	MED
Near Distance	$0 - 20m$	NEA

Table 1: Fuzzy Input Term Definitions (Route-Following). We are using a 3x3 input set model as based on our previous works. This table shows what range of input values are accepted as part of each set. The third column gives us the fuzzy shorthand name for each set. Note that the ranges for the sets overlap; this helps us smoothly transition between rules rather than hard-step as in traditional logic.

Real Term	Rough Set Value Range	Fuzzy Term
Wide Arc	$> 45^\circ$	WID
Mid-range Arc	$0 - 67.5^\circ$	MID
Narrow Arc	$0 - 45^\circ$	NAR
Far Distance	$> 20m$	FAR
Medium Distance	$0 - 28.28m$	MED
Near Distance	$0 - 14.14m$	NEA

Table 2: Fuzzy Input Term Definitions (Obstacle Avoidance). An interesting feature of the obstacle avoidance input sets is that, whilst route-following behaviour can operate alone (when there are no obstacles), obstacle-avoidance is actually a modifier to route-following. The input angles here are double the range for route following so that the opposite extreme turn can be applied to any route-following turn if so required.

shown that fuzzy-logic based models can represent real traffic flow more accurately than traditional GM traffic simulation models, and take more detailed simulation variables into account - such as representing behaviour based on different levels of driver experience, and the more accurately modelling the affect that varying weather conditions have on driver behaviour across different experience levels [21].

Fuzzy Logic allows us to represent a *partial truth*, or imprecise values between *completely true* and *completely false*. This gives us a mechanism for discriminating imprecise or changing data into a small group of overlapping *fuzzy sets*. From this foundation we can create very simple judgement-based reasoning, or *fuzzy inference*, to deal with complex real-world data; mimicking human decision-making. Fuzzy Logic systems require very little computational overhead, and can also produce smooth transitional outputs. Fuzzy Logic is therefore an ideal candidate for modelling human driver behaviour in large-scale, real time simulations.

In our current driver agent model we are concerned with two fuzzy systems. Both operate in real-time, and both are

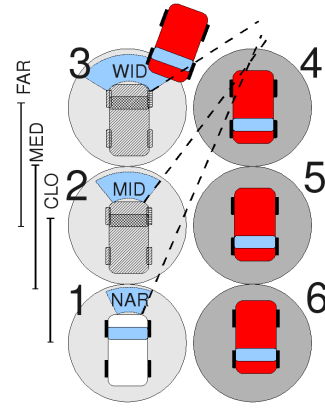


Figure 6: A scenario illustrating the design of input distances and angles to the Fuzzy Obstacle Avoidance systems for vehicles in urban traffic. Here the fuzzy input angles are designed such that, over the range of distances, oncoming traffic can be ignored unless it crosses over the centre line. An oncoming vehicle that has crossed lane into the path of the white car in position 1. Input distances and angles to the Fuzzy Obstacle Avoidance systems are shown. Situations where the white car had been in positions 2 and 3 are considered.

blended together to form a combined result for steering and acceleration behaviour. These two complimentary systems are:

- Reactive Obstacle Avoidance (OA)
- Route-Following Behaviour (RF)

The Route-Following system takes as input the *next waypoint* fed down from the path-planning layer. Specific inputs used are the *change in heading angle* and *distance* to this waypoint. Conversely, the Obstacle-Avoidance system considers the *change in heading angle* and *distance* to the nearest obstacle. Both of these systems classify the real inputs into overlapping *rough sets*, where they can be classified in human terms, for a quick look-up-table type decision. The mapping of these inputs is designed so that, for a vehicle moving up a lane, oncoming traffic can be ignored unless it crosses into the path of the subject vehicle, and if so - a series of rules will progressively move the subject vehicle away from its route, avoid the obstacle, then finally return to its route. Figure 6 illustrates this scenario, and shows the range of distances and angles used as input. Tables 1, and 2 provide our design for fuzzy input set mappings. We produce fuzzy sets for classifying (or *fuzzifying*) real inputs into our rough sets, these are illustrated in Figures 7-10.

We have also mapped our vehicles' output performance specifications to fuzzy output sets (Tables 3 and 4). Because we want to represent a range of different vehicles with the same fuzzy system, we have expressed our outputs in terms of a vehicle's *maximum velocity* ( $v_{max}$ ). The steering ad-

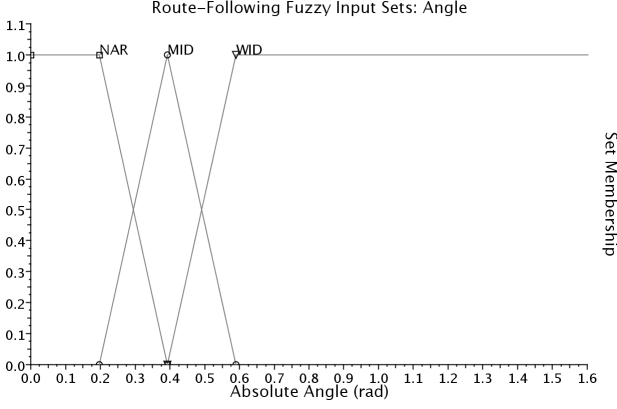


Figure 7: Fuzzy Input Set Membership Functions for classifying the *angle* to the nearest obstacle or next waypoint in Fuzzy terms. Angles here are absolute radians to the left or right of the current heading of a vehicle, so that waypoints on the left hand side of a vehicle are treated the same as waypoints to the right.

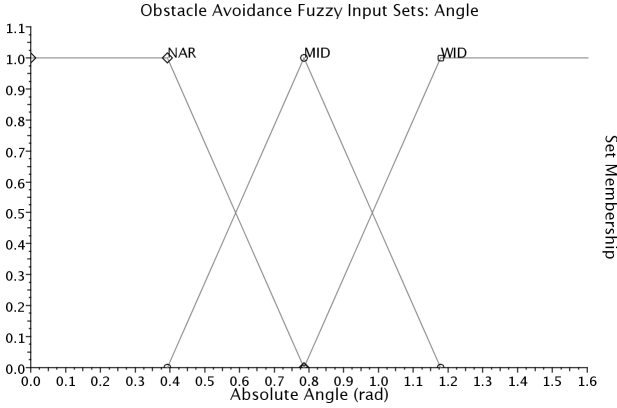


Figure 8: Fuzzy Input Set Membership Functions for classifying the *angle* to the nearest obstacle or next waypoint in Fuzzy terms. Angles here are absolute radians to the left or right of the current heading of a vehicle, so that obstacles on the left hand side of a vehicle are treated the same as obstacles to the right.

justment outputs are also expressed relative to the vehicle's current speed; such that resultant movement vector of the vehicle is to some extent scalable, and we can expect similar rates of vehicle turn at different speeds. A further feedback system is then used to scale back the output if current physical performances limits are reached.

Fuzzy Output set mapping functions are illustrated in Figures 11-13.

The labour-intensive task is then to design fuzzy rules. These map all of the different combinations of fuzzy input values to fuzzy outputs. A sample of our rule design is given in Algorithm 1. Whilst we can manually tune these rules to produce a satisfying output for one particular vehi-

Real Term	Centre Value	Fuzzy Term
Top Speed	$1.0 \cdot v_{max}$	TOP
Fast Speed	$0.8 \cdot v_{max}$	FAS
Medium Speed	$0.6 \cdot v_{max}$	MED
Slow Speed	$0.4 \cdot v_{max}$	SLO
Very Slow Speed	$0.2 \cdot v_{max}$	VSL
Stop	$0.0 \cdot v_{max}$	ZER
Full turn	$v_{defuzz} \cdot 0.08rad \cdot s^{-1}$	FUL
Very Sharp Turn	$v_{defuzz} \cdot 0.06rad \cdot s^{-1}$	VSH
Sharp Turn	$v_{defuzz} \cdot 0.04rad \cdot s^{-1}$	SHA
Medium Turn	$v_{defuzz} \cdot 0.02rad \cdot s^{-1}$	MED
Light turn	$v_{defuzz} \cdot 0.01rad \cdot s^{-1}$	LIG
Very Light turn	$v_{defuzz} \cdot 0.005rad \cdot s^{-1}$	VLI
No turn	$v_{defuzz} \cdot 0.0rad \cdot s^{-1}$	ZER

Table 3: Fuzzy Output Term Definitions (Route-Following Component). These fuzzy sets are quite different to fuzzy input sets, as their purpose is to define centre-points for a centre-of-gravity function. The process of combining fuzzy outputs into a final combined crisp output is called *defuzzification*. We have defined a large range of output sets to give our rules more flexibility. Note that not all outputs are used in rules. We have left them in the system as we intend to develop an auto-training system. This then gives the auto-training system flexibility with a full range of values.

Real Term	Centre Value	Fuzzy Term
Top Speed	$1.0 \cdot v_{max}$	TOP
Fast Speed	$0.8 \cdot v_{max}$	FAS
Medium Speed	$0.6 \cdot v_{max}$	MED
Slow Speed	$0.4 \cdot v_{max}$	SLO
Very Slow Speed	$0.2 \cdot v_{max}$	VSL
Stop	$0.0 \cdot v_{max}$	ZER
Full turn	$v_{defuzz} \cdot 0.16rad \cdot s^{-1}$	FUL
Very Sharp Turn	$v_{defuzz} \cdot 0.12rad \cdot s^{-1}$	VSH
Sharp Turn	$v_{defuzz} \cdot 0.08rad \cdot s^{-1}$	SHA
Medium Turn	$v_{defuzz} \cdot 0.04rad \cdot s^{-1}$	MED
Light turn	$v_{defuzz} \cdot 0.02rad \cdot s^{-1}$	LIG
Very Light turn	$v_{defuzz} \cdot 0.01rad \cdot s^{-1}$	VLI
No turn	$v_{defuzz} \cdot 0.0rad \cdot s^{-1}$	ZER

Table 4: Fuzzy Output Term Definitions (Obstacle Avoidance Component). The values expressed in our output sets are not actual velocities, as one would expect, but rather a portion of a vehicle's maximum allowable speed. This allows us to use the same output set definitions for a range of different vehicles, and also for vehicles in different speed limit zones. The output steering is relative to a vehicle's output speed - this helps us to maintain similar steering rates (velocity vectors) at a range of different speeds.

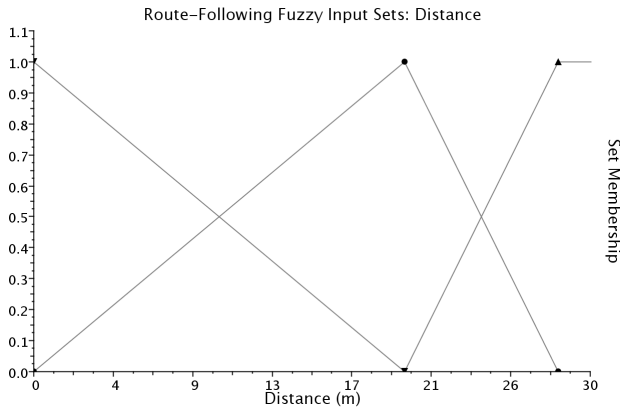


Figure 9: Fuzzy Input Set Membership Functions for classifying the *distance* to the next waypoint in Fuzzy terms.

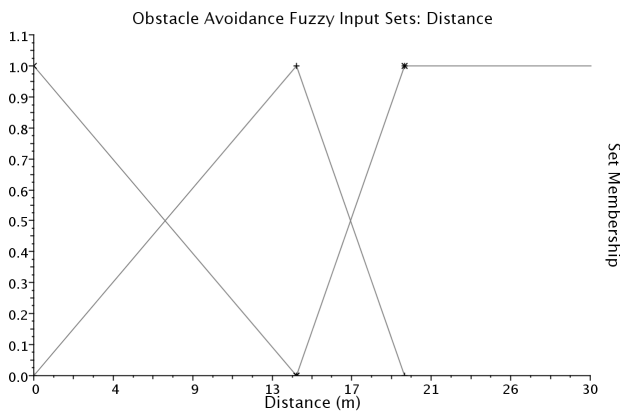


Figure 10: Fuzzy Input Set Membership Functions for classifying the *distance* to the nearest obstacle in Fuzzy terms.

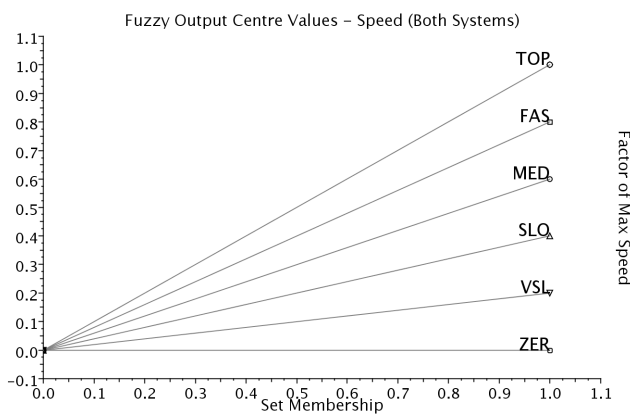


Figure 11: Fuzzy Output Value centres for desired *speed* factor. The defuzzified output speed factor will be multiplied with the vehicle's top allowable speed to produce a desired speed as a real number.

cle, the fuzzy systems need to be recalibrated for every vehicle with different performance characteristics or physical dimensions. As we wish to simulate a large variety of urban

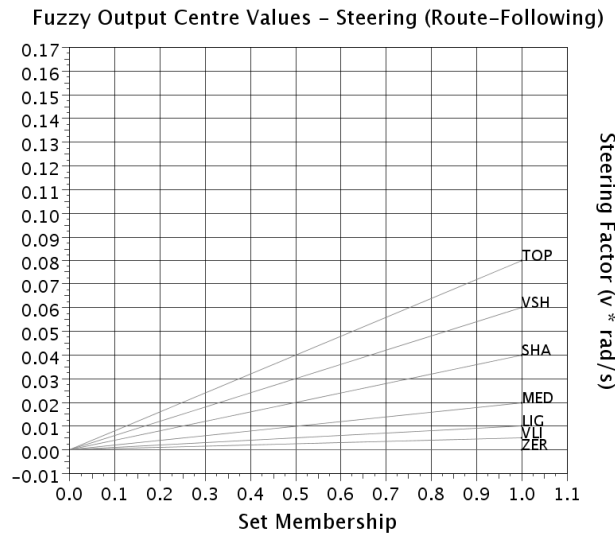


Figure 12: Fuzzy Output Value centres for base route-following desired *steering adjustment* factor. This factor is directly proportional to the vehicle's base route-following speed factor so that the vehicle will steer remain én route over a range of speeds.

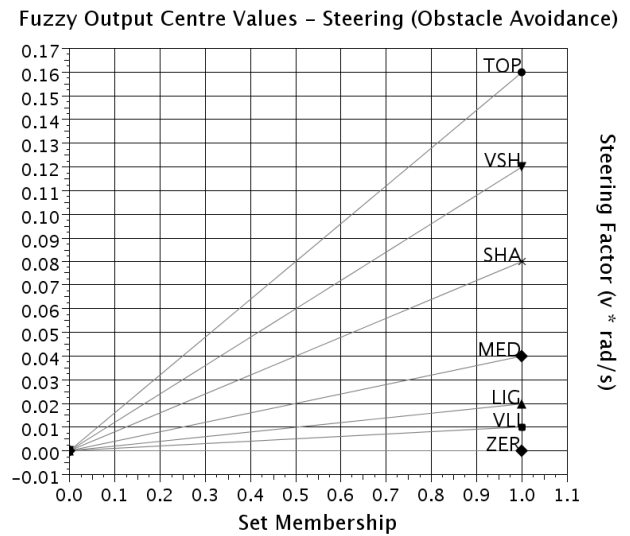


Figure 13: Fuzzy Output Value centres for obstacle avoidance desired *steering adjustment* modifier. This factor is subtracted from the route-following steering factor. To enable the same range of steering adjustment these values are exactly double those used for route-following.

vehicles, then this task becomes a serious constraint-based problem; and we intend to tackle this in our next work by way of an automatic-calibration and self-training system. Our initial rules are provided in Tables 5-8. We note that not all of our output sets have been used in these rules, and that there is certainly scope for designing more balanced rule-sets.

**Algorithm 1** A subset of the fuzzy rule design for Obstacle Avoidance. Rules map each possible combination of fuzzy inputs to valid fuzzy outputs. These rules are based on scenario diagrams as illustrated in Figure 6. After designing the fuzzy inference engine in human terms like this we create look-up tables for quick operation as expressed in Tables 5 and Tables 6.

---

**if** the obstacle is a *near away* **and**  
the obstacle is a *narrow* angle from our heading **then**  
change speed to *zero*, **and**  
change steering to a *sharp turn*.  
**else if** the obstacle is a *medium* distance away **and**  
the obstacle is a *narrow* angle from our heading **then**  
change speed to *very slow*, **and**  
change steering to a *medium turn*.  
**else if** the obstacle is a *far* distance away **and**  
the obstacle is a *narrow* angle from our heading **then**  
change speed to *zero*, **and**  
change steering to a *very light turn*  
**end if**

---

	NEA	MED	FAR
NAR	ZER	VSL	ZER
MID	VSL	SLO	MED
WID	SLO	MED	FAS

Table 5: 3x3 FAMM for Desired Speed (Obstacle Avoidance Component). Output fuzzy speeds are given for all fuzzy input distances and angles.

	NEA	MED	FAR
NAR	SHA	MED	VLI
MID	MED	VLI	ZER
WID	VLI	ZER	ZER

Table 6: 3x3 FAMM for Desired Steering (Obstacle Avoidance Component). Output fuzzy steering adjustments are given for all fuzzy input distances and angles.

note: not all outputs used - scope for using these for adjustment of rules

	NEA	MED	FAR
NAR	VSL	FAS	TOP
MID	VSL	MED	FAS
WID	VSL	SLO	MED

Table 7: 3x3 FAMM for Desired Speed (Route-Following Component). Output fuzzy speeds are given for all fuzzy input distances and angles.

	NEA	MED	FAR
NAR	ZER	VLI	LIG
MID	VLI	LIG	MED
WID	LIG	MED	SHA

Table 8: 3x3 FAMM for Desired Steering (Obstacle Avoidance Component). Output fuzzy steering adjustments are given for all fuzzy input distances and angles.

Once all of the rules have been evaluated we use an aggregation *centre-of-gravity* function to merge the outputs for each system (Equation 4). The outputs of both systems are then blended together; the steering of the obstacle-avoidance component is exactly double that of the route-following system in order to reflect this. The rules, therefore, need to be designed to reflect the operation of the system in combination; a difficult task to accomplish manually, and which further underlines the advantage that would be gained from an auto-calibrating system.

$$v_{defuzz} = \frac{\sum_{i=top}^{zer} mem_i * v_i}{\sum_{i=top}^{zer} v_i} \quad (4)$$

Talk about obstacle avoidance - static versus dynamic. Most static and dynamic handled at layer above, but still need to have this information on hand in case move near obstacle 2 while avoiding obstacle 1 etc etc. Can either scan in automatically (if regular shapes or small) or for irregularly shaped buildings - manually add strips of points so that along edges of buildings it can be considered a series of smaller obstacles (so that we have a v quick method for picking a 'nearest' obstacle without using some expensive trig function). Also can do a lookup of this meta-data based on grid reference that we are in as each obstacle associated with a 2D array reference for v quick lookup.

## 4 Discussion and Conclusions

We have implemented a prototype of the complete system. This was demonstrated at the *Metropolis* exhibition at the Science Gallery, Trinity College Dublin. Figure 14 demonstrates the complete system in action; a range of vehicles have been *spawned* to replicate high-congestion traffic around the Trinity College Dublin campus. A range of vehicles, road lanes, and mergers are represented.

Figure 15 presents this same scene, but with road lane demarcations visible. Each one of the arrows in Figure 15 corresponds to one of our road lane *nodes* - a roughly vehicle-sized tag used by the agents to facilitate traffic queueing and vehicle-following higher-level behaviours.



Figure 14: Prototype system in operation: real-time congested traffic simulation for College Street, Dublin City.

In a large city scene, with over 2000 vehicles simulated in real-time simultaneously, we were able to maintain processing of over  $200Hz$  (frames rendered per second) using a single thread programme architecture on an Intel Core(TM)2 Quad  $2.4GHz$  CPU with a GeForce 8800 GTX card, which more than satisfied our requirement that the traffic was unintrusive in terms of the overall programme.

hicle through a range of *obstacle courses* representing typical simulation operating environments and key problems (tight corners, intersections, merging traffic, static and dynamic obstacles etc). Batches of these runs will generate a large amount of comparative performance data, which gives us grounds for full agent improvement, and should provide us with enough information to show how effective this approach is in tuning fuzzy sets and rules for best effect. The best *selection* method for self-improvement is a matter of study in future work, with options ranging from:

- brute-force methods
- evolutionary selection
- genetic algorithms

Genetic algorithms have been designed for the similar problem of soccer robots [22], and as our hybrid system is an extension of robot navigation algorithms this warrants investigation also.

Our prototype simulation already has this infrastructure in place, initially using an architecture as illustrated in Figure 17; where we are using the *time taken* and a *collision heuristic* as a fitness (evaluation) function. Whether or not the resulting vehicle controllers will produce human-like behaviour is in question, as is the requirement of additional smoothing of outputs as by PID controllers.



Figure 16: Street-level scene of system in operation: real-time congested traffic simulation for Dame Street, Dublin City.

Our main intention is to extend this work Figure 17 with a self-training module that operates by running a new ve-



Figure 15: Prototype system in operation: lane demarcations for College Street, Dublin City.

## References

- [1] A. P. Gerdelan. A solution for streamlining intelligent agent-based traffic into 3d simulations and games. Technical Report CSTN-072, Complex Systems and Simulations Group (CSSG), Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand, January 2009.
- [2] Louis A. Pipes. An operational analysis of traffic dynamics. *Journal of Applied Physics*, 24(3):274–281, 1953.
- [3] S. R. Perkins and J.I. Harris. Report gmr632, criteria for traffic conflict characteristics. Technical report, General Motors Corporation, Warren, MI, 1967.
- [4] S. R. Perkins and J. I. Harris. Traffic conflict characteristics: Accident potential at intersections. *Highway Research Record, Highway Research Board, Washington DC*, 225:45–143, 1968.
- [5] W. Leutzbach and R. Wiedemann. Development and applications of traffic simulation models at the karlsruhe institut für verkehrswesen. *Traffic Eng. Control*, 27:pp. 270278, 1986.
- [6] P. G. Gipps. A behavioural car following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):pp. 105111, 1981. CSIRO Division of Building Research, Melbourne, Australia.
- [7] S. Krauß, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597 – 5602, May 1997.
- [8] S. Krauß. *Microscopic Modeling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics*. PhD thesis, Mathematisches Institut, Universität zu Köln, 1998.
- [9] R. Wiedemann. Simulation des straßenverkehrsflusses. Technical report, Schriftenreihe des Instituts für Verkehrswesen, Heft 8, Universität (TH) Karlsruhe, Deutschland, 1974.
- [10] L. Bloomberg and J. Dale. Comparison of vissim and corsim traffic simulation models on a congested network. *Transportation Research Record: Journal of the Transportation Research Board*, 1727 / 2000:52–60, January 2007.
- [11] M. E. Bratman. *Intentions, Plans, and Practical Reason, Agents*. Harvard University Press, Cambridge, MA, 1987.
- [12] K. A. Hawick and A. P. Gerdelan. Software integration architectures for agents. Technical Report CSTN-054, Complex Systems and Simulations Group

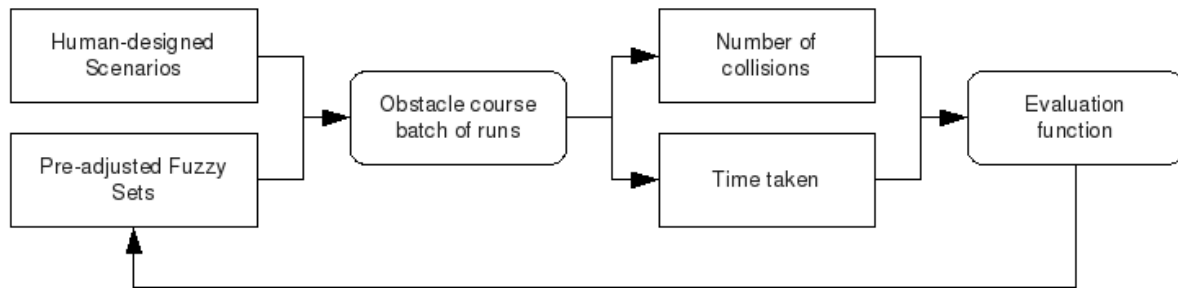


Figure 17: Proposed self-training system for fuzzy navigation set membership functions.

(CSSG), Institute of Information and Mathematical Sciences (IIMS), Massey University, New Zealand, May 2008.

Institute of Information and Mathematical Sciences, Massey University, North Shore 102-904, Auckland, New Zealand, January 2009.

- [13] A. P. Gerdelan and N. H. Reyes. Towards a generalised hybrid path-planning and motion control system with auto-calibration for animated characters in 3d environments. *Springer-Verlag, Lecture Notes in Computer Science*, Proceedings of the 15th International Conference on Neural Information Processing (ICONIP2007):25–28, November 2008.
- [14] A. P. Gerdelan, D. Iskandar, A. F. Djohar, and N. H. Reyes. 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 (NCEI06) and 6th International Conference on Hybrid Intelligent Systems (HIS '06)*, 2006.
- [15] A. P. Gerdelan and N. H. Reyes. *Advances in Soft Computing: Computational Intelligence: Theory and Applications*, chapter Synthesizing Adaptive Navigational Robot Behaviours using a Hybrid Fuzzy A\* Approach, pages pp. 699–710. Springer, (Berlin & Heidelberg), 2006.
- [16] A. P. Gerdelan and N. H. Reyes. 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*, volume vol.1, pages pp. 75–79, Auckland, New Zealand, 2006.
- [17] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ., second edition edition, 2003.
- [18] K. A. Hawick, D. P. Playne, A. Leist, A. P. Gerdelan, and C.J.Scogings. Its about time: the role of time in simulations. Technical Report CSTN-072, Complex Systems and Simulations Group (CSSG),
- [19] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*, chapter H.2 A\* Algorithm, pages 527–536. MIT Press, 2005. ISBN 0262033275, 9780262033275.
- [20] Mark Dougherty, Ken Fox, Michael Cullip, and Marco Boero. Technological advances that impact on microsimulation modelling. *Transport Reviews*, 20(2):145–171, 2000.
- [21] Jaimu Won, Sooil Lee, Soobeom Lee, and Tae Ho Kim. *Advances in Multimedia Modeling*, volume 4352/2006 of *Lecture Notes in Computer Science*, chapter Establishment of Car Following Theory Based on Fuzzy-Based Sensitivity Parameters, pages 613–619. Springer Berlin / Heidelberg, 2006.
- [22] C. H. Messom. Genetic algorithms for autotuning mobile robot motion control. *Res. Lett. Inf. Math. Sci.*, (3):pp 129–134, 2002. ISSN 1175-2777.