A Spatial Awareness Framework for Enhancing Game Agent Behaviour

Simon Perkins[*]
University of Cape Town

David Jacka[†]
University of Cape Town

Patrick Marais[‡]
University of Cape Town

James Gain[§]
University of Cape Town

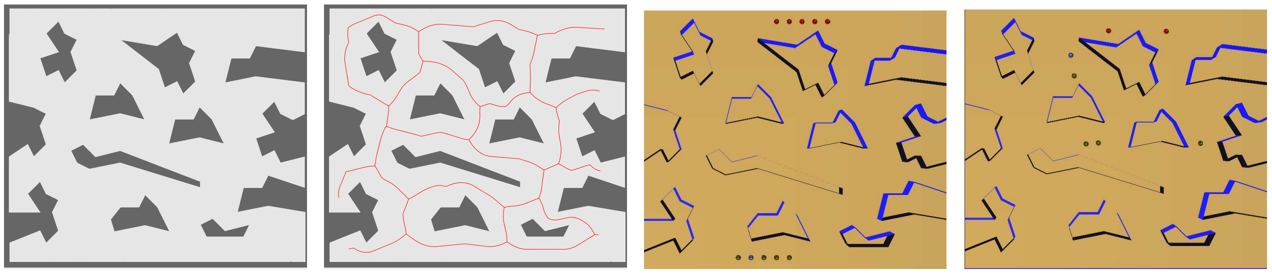


Figure 1: Starting with the geometry defining a virtual world in 1a, information on connectivity, width and curvature is extracted in 1b. This information is used by agents to enhance their behaviour within the virtual world. In 1b and 1c the beige and blue agents use this enhanced behaviour to defeat their red opponents.

Abstract

We describe a framework for providing game agents with awareness of the intrinsic spatial qualities of the virtual worlds that they inhabit. We develop a novel data structure based on a modified medial axis, which establishes a mapping between the medial axis and world structures. This data structure can be used to perform queries about the width, curvature and connectivity of a space within a virtual world. Additional information, such as sampled visibility can also be integrated with this framework. An agent-based crowd simulation is adapted to make use of the sensory information provided by this data structure and the success of using this information within two game scenarios is evaluated.

CR Categories: I.2.10 [Artificial Intelligence]: Vision and Scene Understanding—[Representations, data structures and transforms, Perceptual reasoning] I.4.8 [Image Processing and Computer Vision]: Scene Analysis— [I.4.7]: Image Processing and Computer Vision—Feature Measurement

Keywords: spatial analysis, agents, games

1 Introduction

Artificial Intelligence Agents are actors within a virtual world, such as a game environment or training simulation. They are important components in the creation of these worlds as they contribute to the realism of a virtual experience.

Humans modify their behaviour based on their environment. Straight and gently curving corridors of space are appropriate for running, while sharply curving spaces are not. Wide-open spaces provide good vantage points for trying to spot other objects, but are bad spaces to hide in. Dimly lit spaces, however, are good to lurk in.

Since humans strategise their behaviour based on their environment, designers of AI agents may wish to do the same to create compelling agent behaviour. It would be appropriate to design a bird agent to fly in wide-open spaces, but hop or walk in narrow, enclosed spaces. A designer trying to make an agent behave in a "sneaky" manner may wish the agent to favour dimly lit areas with poor visibility from other areas. To provide this spatial information to agents, it is necessary to perform spatial analysis on the virtual environment.

Pathfinding is a type of spatial analysis that is specifically designed to assist the navigation of agents within a world. However, other types of spatial information are extracted during the creation and preprocessing of a virtual world, and while such information may not be specifically extracted with agents in mind, it may still be useful to them. For example the visibility between different points in a world allows one to avoid rendering invisible world structures [Teller and Séquin 1991]. This information may also be useful to an agent wishing to strategise about the suitability of a vantage point.

Similarly, the lighting in various areas of a virtual world is calculated to improve the realism of a rendered image, but this information may also be useful if agents are designed to react to lighting information. Some agents may see better in the dark and should favour dark areas to gain an advantage

[*]e-mail: sperkins@cs.uct.ac.za
[†]e-mail: djacka@cs.uct.ac.za
[‡]e-mail: patrick@cs.uct.ac.za
[§]e-mail: jgain@cs.uct.ac.za

Sandbox Symposium 2008, Los Angeles, California, August 9–10, 2008.
© 2008 ACM 978-1-60558-173-6/08/0008 $5.00

over opponents with poor vision.

These fields have been well researched, but less work has been done on providing information about the intrinsic qualities of a space. For example, the way a space curves is useful to an agent when planning when to accelerate and decelerate, especially if its turning behaviour is physically modelled. The width or openess of a space is also a useful measure for evaluating proximity to world structures and strategising about wide or narrow spaces.

We present a novel data structure that provides useful data on higher-order connectivity, curvature and width. We also describe the process for automatically generating this structure from a world defined by 2D polyons. The data from this structure can be combined with data extracted from other sources, such as visibility and lighting. We implement simple agents and demonstrate how their effectiveness can be improved by utilising this data in two different scenarios. In the first, racing car agents use data about track curvature to improve their racing line. In the second, battle robots use data about path intersections to orientate themselves towards areas of high traffic. The agents use almost no planning capability since the intention of our work is not to improve agent design, but to assist it.

Our work has been developed in 2D as a preliminary to a full 3D implementation. It is worth noting that while current games are largely rendered in 3D, much of the strategy is based on 2D and 2.5D evaluation of worlds.

This paper is structured as follows: We describe previous work in Section 2 and discuss background material relevent to our work in Section 3. Section 4 describes our general approach to the problem and the creation of our data structure. Section 5 describes the agents that we use to test our work. Section 6 describes our testing and results and we conclude in Section 7.

2 Previous Work

Navigations Graphs are commonly used to enable agents to navigate within a virtual world by providing perception of paths within the world. They provide a simplified representation of the spatial areas that an agent may occupy. This reduces the time needed to search through positions in order to plan routes and make decisions. At first designers created navigation graphs by manually placing waypoints in the virtual world [Lidén 2000]. As virtual worlds increase in size and complexity, this task became more time-consuming and a candidate for automation.

The Quake III bot [van Waveren 2001] is an example of the automation of navigation graph creation. It extracts a navigation graph from a Binary Space Partition (BSP) Decomposition of a 3D world. The traversability of the convex BSP leaf nodes are examined and are linked together to create the graph. The *Navigation Mesh* [O'Neill 2002] simplifies navigation graph creation by taking advantage of the fact that 3D games are often set on earthlike terrain with humanoid agents. Since it is only possible for these agents to walk on the ground, the navigation mesh is constructed by connecting the agent-accessible polygons. As noted by [van der Sterren 2001], such navigation algorithms are useful for calculating the shortest path between two locations, but they do not assist agents in understanding the terrain around them.

Pottinger [Pottinger 2000] discusses *influence maps* which are created by applying terrain influences to a 2D array to determine the best position to site a game object. He also discusses grouping logical areas together for AI use via area decomposition and the importance of establishing connectivity between such areas for pathfinding purposes. However, this information is derived from features external to the terrain and not from the terrain itself. Morgan [Morgan 2003] evaluates a number of algorithms for determining suitable locations for a soldier to take a cover within a virtual world including using *Shadow BSP Trees* [Chrysanthou and Slater 1995] to detect regions of concealment. For reasons of efficiency, he decides to use a sensor grid which evaluates visibility around an agent at different heights to decide if a location may be used for cover.

Van der Sterren [van der Sterren 2001] discusses terrain reasoning by examining the relationship between waypoints in a navigation graph. The connectivity and visibility between waypoints is used to make estimates about the effectiveness of a waypoint as a firing position. This effectiveness is then modified by the actual performance of agents at the waypoint. Van Der Sterren recognises the need for annotating waypoints with higher-order terrain information such as visibility and lighting. However, the focus on waypoints inherently discards intrinsic geometric data that may be useful to agents and introduces resolution issues. For example, the curvature of a section of space would be difficult to reconstruct from waypoints and resolution issues would complicate the matter further. It also difficult to reason about higher-order connectivity because waypoints are distributed throughout the world in order to provide agents with sensory information. The proliferation of waypoints maybe obscure the fact that there is a single logical path that all the waypoints in a region may belong to. In contrast our method focuses on retaining such information since it may be useful to agent designers.

3 Background

Binary Space Partition (BSP) Trees [Fuchs et al. 1980] are commonly used in spatial analysis. It is a binary tree that recursively divides a space using half-planes. They are typically created from a set of polygons, using the polygon planes as splitting half-planes. BSP trees are commonly used in virtual worlds as they are useful in calculating visibility and performing collision detection.

Voronoi Diagrams [Voronoi 1907] are another useful tool for spatial analysis. Given a set of points S in a plane, a Voronoi diagram partitions the plane into convex polygons, each containing one point $p \in S$ and having the property that every point in the polygon is closer to p than any other point in S. Voronoi diagrams have various applications but the one that pertains to our work is their use in generating a *Medial Axis*.

In 2D, the medial axis of a shape can be defined as a set of curves defined as the locus of points have have two closest points on the boundary of a shape [Cornea and Min 2007]. It can be used for representing the shape of an object, since it constitutes a set of curves that run roughly down the centre of an object. Voronoi diagrams are frequently used to approximate the medial axis [Ogniewicz 1994a]. This process is used in path planning [Bhattacharya and Gavrilova 2007] and robotic motion planning [Guibas et al. 1999; Holleman and Kavraki 2000] to generate a medial axis, from which a navigation graph is derived.

Amenta et. al [Amenta et al. 1998] use Voronoi Diagrams to reconstruct surfaces from unorganised sample points. They use a process they call "Voronoi filtering" to choose faces of the Delauney simplices to remove. Of interest here is their use of the medial axis to construct a metric that relates sampling density to surface curvature, and from which they can prove the accuracy and topological validity

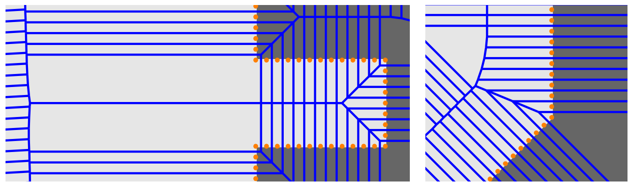

(a) Major Concavity (b) Minor Concavity

Figure 2: Major concavities introduce significant changes into the width and curvature of a space and therefore warrant a skeleton extension into the space. The changes introduced by minor concavities do not.

of their surface reconstructions.

4 Approach

We aim to develop a data structure representing the intrinsic qualities of a space, such as width, curvature and connectivity. A *medial axis* or *skeleton* is useful for representing these qualities since in 2D it is a set of curves that run through centre of a space. From the skeleton, logical paths through the space can be identified, providing connectivity information and allowing the path curvature to be used to identify the curvature of the surrounding space.

We first perform a BSP decomposition of the world. Since this structure subdivides the world using half-planes, convex regions representing the solid areas of a world can conveniently be determined and grouped together. The boundaries of these grouped regions can then be identified by traversing the outer boundary of the group. This approach means that a level designer can conveniently construct world structures out of separate polygons. It also allows for world structures with holes.

As we are dealing with geometric representations of virtual worlds, we adapt the popular geometric technique of extracting a medial axis from a Voronoi diagram [Ogniewicz 1994a]. The boundaries extracted from the BSP process are sampled and used as input to the voronoi tesselation process.

Medial axis transforms tend to overemphasise boundary details [Ogniewicz 1994b] and this leads to the creation of fine vestigial elements that represent complex boundary information. We do not wish to capture this structural data directly because it is too fine: we are interested in capturing the general qualities of a space.

Consider Figure 2a. The *major concavity* introduces significant changes to the width, curvature and connectivity of the space and should therefore be represented on the skeleton. The *minor concavity* in Figure 2b does not and should therefore be left off the skeleton. To this end, we prune sections of the skeleton that extend into minor concavities.

While the curvature of a space can be easily measured from the curvature of the skeleton, it is not necessarily trivial to arrive at a definition of the width of a space. In most cases the shortest distance between the skeleton and world boundary suffices. However, concavities again introduce problems, since the furthest point within a concavity can be the appropriate point to choose when measuring the width between the skeleton and the world. To deal with this issue, we develop a bi-directional mapping between the skeleton and world that ensures that sections of skeleton are associated with the correct sections of world boundary.

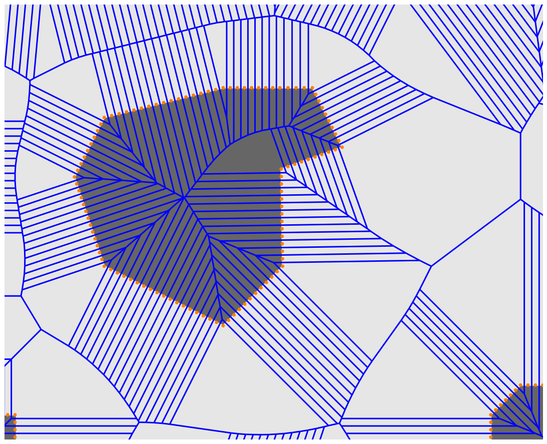


Figure 3: Voronoi Tesselation created from orange input points, sampled from the boundary of the gray world structures. The blue edges and vertices of this tesselation are linked together in a graph.

4.1 Skeleton Extraction

We create a modified medial axis from a Voronoi tesselation [Voronoi 1907] of virtual world objects. This medial axis is then pruned to fit our definiton of the skeleton.

Sampling polygons: We perform a Binary Space Partition (BSP) [Fuchs et al. 1980] of the polygons describing the world and extract the convex regions defined by the BSP tree half-plane intersections. Solid regions representing a distinct world structure are grouped together and the counter-clockwise boundaries of this structure are extracted. This boundary is stored as a closed, parameterised line.

Voronoi Tessellation and Initial Skeleton: Points are sampled on the the parameterised boundaries and are used as input points to the Voronoi tesselation process. We use *qhull* [Barber et al. 1996] to perform the tesselation. For each input point, a voronoi facet is generated, consisting of a number of voronoi vertices. The vertices and the edges between them are linked together in a graph as shown in Figure 3. Initially, a voronoi vertex is labelled as "off" the skeleton if it lies within a world structure, otherwise it is considered to be on the skeleton. A graph edge is considered to be on the skeleton if both it's starting and ending point are on the skeleton.

Pruning the skeleton: According to our definition of the skeleton, skeletal sections extending towards minor regions of concavity represent too much detail and should be pruned. Skeletal sections that extend into major regions of concavity are orthogonal to sections of the world. We therefore test the endpoints to see if the orthonality criterion is met. This is accomplished by examining the skeletal endpoints, e, that only have one neighbour n on the skeleton. An endpoint e is considered *strong* if there exists at least one adjacent non-skeletal neighbour a such that the angle between the vectors $\overrightarrow{en}$ and $\overrightarrow{ea}$ lies in the interval $[\frac{\pi}{2} - \epsilon, \frac{\pi}{2} + \epsilon]$ for some tolerance ϵ. Otherwise the endpoint is considered to be *weak*. *Weak* endpoints are pruned from the skeleton. The process continues until no endpoints remain or only *strongly* supported endpoints remain. Figure 4 shows two examples of this process.

Skeleton Parameterisation: Prior to creating a mapping between world boundaries and the skeleton, parameterisation must be performed to facilitate the mapping. To accomplish this, we need to parameterise both a world bound-

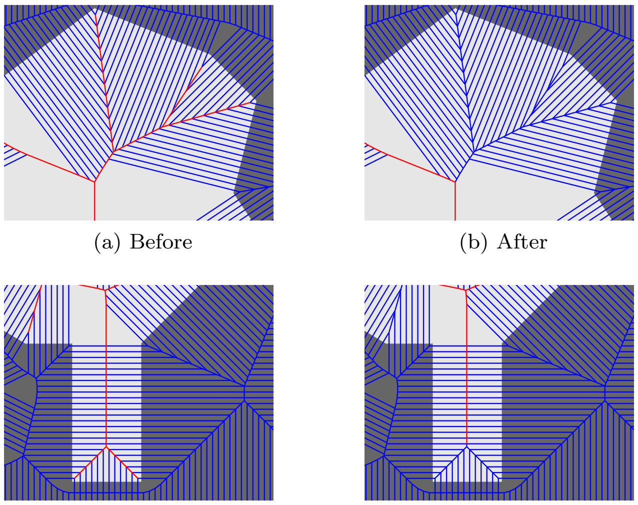

(a) Before (b) After (c) Before (d) After

Figure 4: Two examples of skeleton pruning: 4a shows a number of skeletal endpoints with *weak* supporting neighbours. This results in recursive removal until no endpoints remain as shown in 4b. In 4c the two endpoints are recursively pruned to the configuration in 4d, where only one endpoint remains supported by two *strong* neighbours.

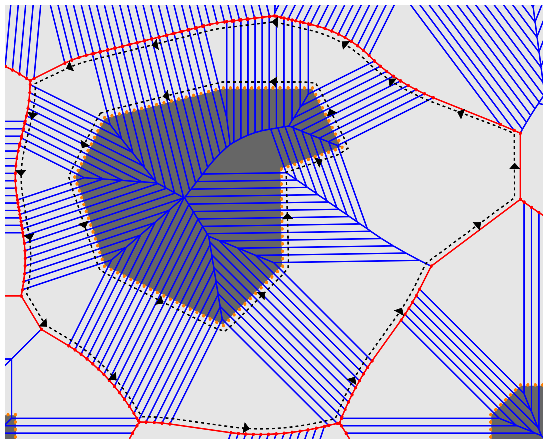


Figure 5: Skeleton Parameterisation: The orange input points to the Voronoi tesselation process produce the blue voronoi facets, which have edges and vertices that lie on the skeleton and are marked in red. The ordering of the input points produces a corresponding voronoi facet ordering, which allows us to establish an ordering of the skeleton points surrounding the world structure.

ary and a section of skeleton with the intent of creating a correspondence between the two. An intuitive way of visualising this is to realise that each world boundary is enclosed by a part of the skeleton. Parameterising a boundary is simple since we can extract a closed counter-clockwise sequence of points from the BSP tree to describe it. However, our skeleton at this point exists as a graph - a set of vertices connected by edges - with no implied direction and no way to choose which path to take at intersecting points.

To parameterise the section of skeleton surrounding a world boundary, we utilise the Voronoi facets derived from the tesselation process. By sampling the parameterised world boundaries, a sequence of points is extracted that is input to the voronoi tesselation process. The tesselation produces a facet for each input point and therefore produces a corresponding sequence of facets that intersect the parameterised boundary. Facets that do not have skeleton vertices can be safely ignored. Once the facet ordering has been established, the ordering of the skeleton points lying on the facets can also be established. This sequence of points is then parameterised. This relation between the voronoi input points and the skeleton points is shown in Figure 5.

At the end of the parameterisation process a parameterised world boundary $P_b(t_b)$, and a section of parameterised skeleton $P_s(t_s)$, that correspond to each other are produced, with $0.0 \leq t_b, t_s \leq 1.0$.

4.2 Mapping Generation

Once a skeleton has been derived from the world structure, and parameterisations for sections of skeleton and world boundaries have been established, we create a mapping between the skeleton and world boundaries. The aim of this mapping is to establish the best possible correspondence between points on the skeleton and points on the world boundary in order to accurately represent the width of the space. Once again, this is accomplished by examining the Voronoi facets that intersect the world boundaries and have an edge on the skeleton. We use these facets to categorise sections of world boundary and skeleton into three classes as shown in Figure 6.

Perpendicular Sections: Consist of one or more consecutively ordered Voronoi facets which have edges that all intersect the world boundary at the same angle. The intersecting edges are perpendicular to each other.
Folded Out Sections: Consist of one Voronoi facet that “fans outward” from the world boundary. The intersecting edges diverge from each other as they move from the world boundary towards the skeleton.
Folded In Sections: Consist of one or more consecutively ordered Voronoi facets which have had weak skeleton endpoints pruned away. These facets “fan in” towards skeleton endpoints that have been removed.

These sections are used to establish local mappings between world and skeleton boundaries and form the building block of the final mapping. Each section is assigned parameterised values for the starting and ending skeleton points (t_{ss} and t_{se}) and starting and ending boundary points (t_{bs} and t_{be}). Two section lists are maintained, a *skeleton ordered section list* ordered by the t_{ss} of each section and a *boundary ordered section list*, ordered by t_{bs}.

When performing a mapping from the parameterised skeleton onto the parameterised boundary for some parameterised skeleton value t_s, the *skeleton ordered section list* is used to look up a section such that $t_{ss} \leq t \leq t_{se}$. Linear interpolation is then performed to derive a corresponding parameterised value t_b for the boundary.

$$t_b = \frac{(t_s - t_{ss})(t_{bs} - t_{be})}{t_{ss} - t_{se}}$$

Folded in sections converge on a single skeleton point such that $t_{ss} = t_{se}$ and linear interpolation fails in this case. To deal with this we simply map t_{ss} and t_{se} to $\frac{t_{bs} - t_{be}}{2}$. Mapping from the boundary on to the skeleton can be performed by reversing the process, with no special cases needing to be dealt with, since sections never converge onto a single boundary point.

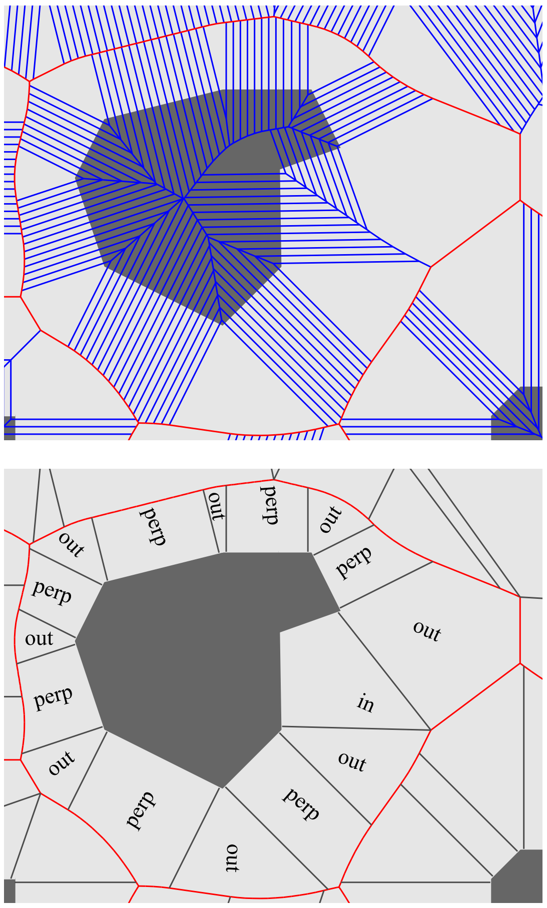


Figure 6: Classifying sections: Sections of space between the world boundary and skeleton are classified into *Folded Out*, *Folded In* and *Perpendicular* sections, based on the Voronoi facets found within the space.

5 Agent Implementation

In order to test the spatial awareness framework, we implemented an agent-based crowd simulation system. The simulation consists of world geometry represented as a set of polygons and a crowd of autonomous, embodied agents. Using forward Euler integration, the agents update their state at each time step based on their perception of the world as well as constraints imposed on them.

The autonomous agents interact with the world (as well as other simulated agents) through a set of senses as shown in Figure 7. These senses take information from the world (such as world geometry or enemy agent positions), as well as information about the local agent (such as turn rate or movement speed), and produce a two dimensional output according to the sense interface. This output is used by the agent to steer.

The senses are tailored for a particular type of agent. Examples include **distance_to_friends**, the distance to each friendly agent in the world, and **geom_vision_x**, the amount of area obscured by world geometry measured along the agent's x-axis. The senses may be customised to allow the agents a greater or lesser knowledge of the world or to allow a certain type of behaviour. For example, the **geom_vision_x** sense may be used for navigation purposes and the **angle_to_friend** sense for tactical decision purposes.

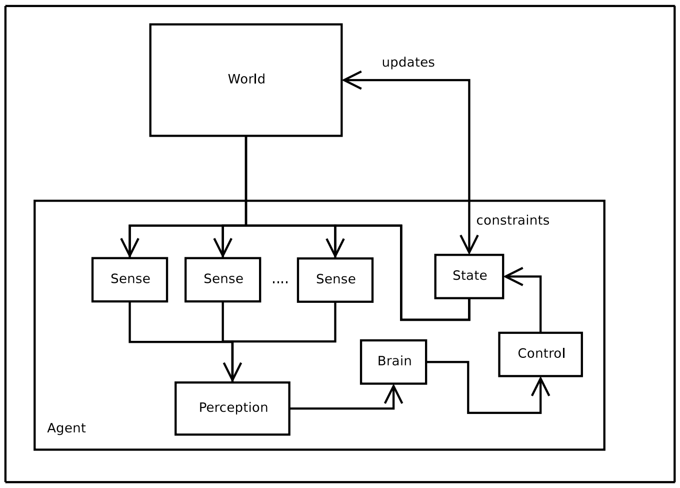


Figure 7: A visual overview how an agent's brain interacts with the world.

The input from the senses are grouped into a perception module and used as input to the agent's brain. This brain then alters control values for the agent which are in turn used to update the agent's internal state.

An agent's brain is a collection of fuzzy rules, well documented in the field of control systems [Zadeh 1996]. These rules may be visualised as a network of fuzzy logic nodes. Each rule, of a standard **if** *a* **then** *b* form, operates on fuzzy variables which, in contrast to the standard boolean variety, take on a value in the range [0, 1]. For a full explanation of fuzzy variable and fuzzy inference, the reader is referred to standard texts [Klir and Yuan 1996].

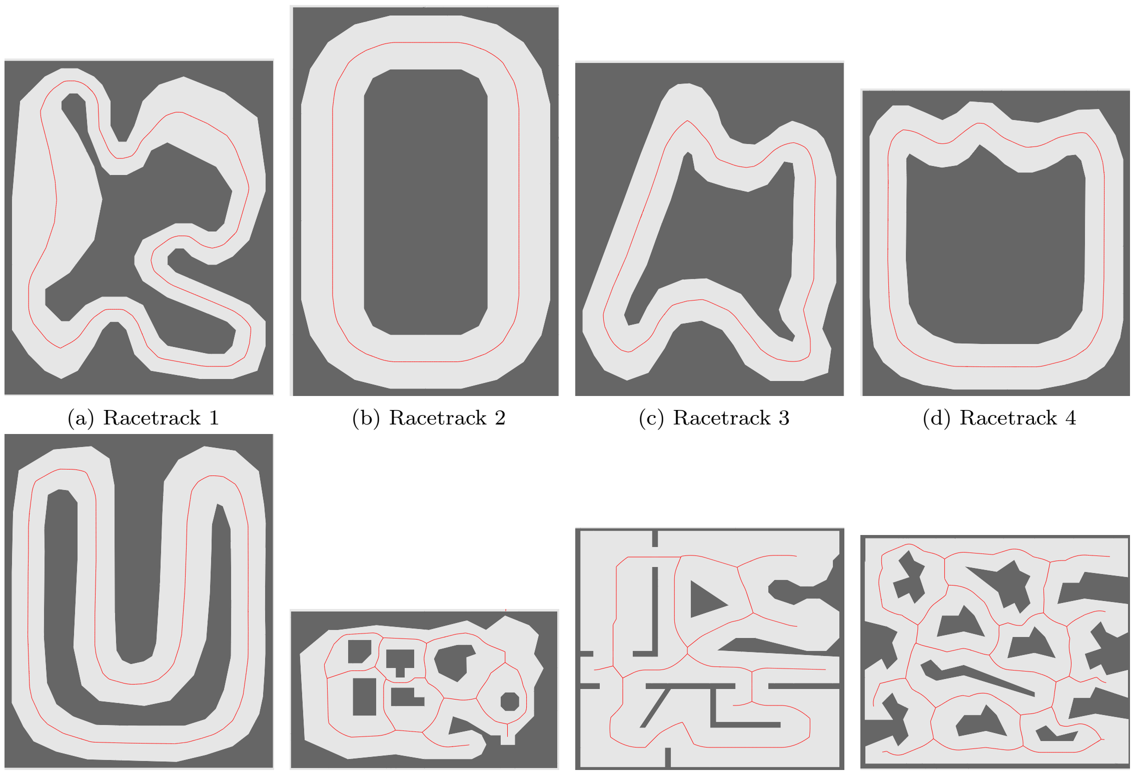
A fuzzification step takes the two dimensional input from the agent's senses and determines the values of the fuzzy variables used by the brain. This is done in a number of different ways, depending on the properties required, usually involving a scaling step followed by a summation or maximum operation. Once the fuzzy inference has been conducted, a defuzzification step is required in order to determine the real values for the agent's controls as well as combine rules that act upon the same control. We use the height method for this defuzzification [Mizumoto 1998] due its computational efficiency.

6 Results

To demonstrate the usefulness of the spatial awareness framework, we created two simple games using the crowd simulation agents designed to play the games at a basic level. We then created a new group of agents based on the original agents, but with additional rules making use of extra sensory information provided by the spatial awareness framework. By observing the performance of the modified agents, we evaluated whether or not the framework has enhanced their behaviour.

6.1 Racing Car Scenario

The racing car game involves the agents moving around a simple track as efficiently as possible while avoiding the walls. We observed that the basic agent, with inter-agent and wall avoidance behaviour, did not take an optimal line around corners. This is due to the agent being purely reactive with no knowledge or recollection of the way in which the track turns.


(a) Racetrack 1 (b) Racetrack 2 (c) Racetrack 3 (d) Racetrack 4

(e) Racetrack 5 (f) Robot War 1 (g) Robot War 2 (h) Robot War 3

Figure 8: The virtual worlds used in testing and their skeletons. The first five were used for testing the racetrack scenario, and the last three for testing the robot war scenario.

We created a sense which provided information on the curvature of the upcoming section of track by considering the angle changes along upcoming sections of skeleton. To accomplish this, the polygonal boundaries of the local mapping sections in the *skeleton ordered section list* were placed in a quadtree. During the game, the quadtree was used to look up the local mapping section containing the agent's position. A binary search was performed within the mapping to calculate the skeleton t value corresponding to the agent's position. The changes in skeleton curvature after the t value were then provided to the agent. Using this information, the agent was able to keep the inside wall of the track in view, hugging the walls and taking a better line around corners.

We created five racing tracks (See Figure 8a to 8e) to test the performance of the agents. Eight racing agents took part in each race, four of which were normal agents and the other four being enhanced with curvature awareness. The agents' starting positions were arranged in the traditional staggered, two column configuration, with the curvature aware agents placed at the back.

In all of the five tracks, the spatially aware agents overtook all the normal agents by the second lap. Taking the inside line on the track shortened the distance they travelled and gave them a better line making it more difficult for normal agents to pass.

The four spatially aware agents queried the framework in realtime. The complexity of this query is $O(\log N)$ since it involves a quadtree lookup followed by a binary search.

6.2 Robot War Scenario

To test the use of the spatial awareness framework in a setting somewhat similar to a first-person shooting game, we created a robot war simulation. Each agent was able to shoot in the direction that they are facing with some degree of randomness in their accuracy. The basic behaviour for a robot agent is the standard agent-and wall-avoidance, as well as a "targetting" behaviour in which an agent turns to face any enemy agent that it sees.

In this scenario, the improved agents were given a sense of how many skeleton intersection points - locations where three or more skeleton sections connected - were visible to them. The rationale for knowledge of these areas being advantageous is that places where paths intersect are likely to have a lot of traffic. This sense is therefore a combination of the intrinsic quality of connectivity provided by the framework and the secondary quality of visibility.

To this end, we generated a *strategy map* from the spatial awareness framework. The strategy map is generated by traversing the skeleton and sampling points on it and to either side of it. At each point, we compute the number of visible skeleton intersection points and two sense values, **best_vis_angle** and **position_goodness**. **best_vis_angle** is set to the angle at which the most skeleton intersection points can be seen in a 30° arc. **position_goodness** is set to the number of intersection points in the 30° divided by the number of visible skeleton intersection points. Thus, if a point has many intersection skeleton points visible in a single 30° arc, it will have a high **position_goodness**, representing the advantage of being able to see many areas of high traffic. The sample points were placed in a kd-tree

Contest	Smart Agent Kills	Normal Agent Kills
Map 1 Team	5	3
Map 2 Team	5	0
Map 3 Team	5	1
Map 1 Team Reversed	5	0
Map 2 Team Reversed	5	2
Map 3 Team Reversed	5	0
Map 1 1v1	1	0
Map 2 1v1	1	0
Map 3 1v1	1	0
Map 1 1v1 Reversed	1	0
Map 2 1v1 Reversed	1	0
Map 3 1v1 Reversed	1	0

Table 1: The outcomes of the agent contests. The number of kills for each type of agent are listed for each contest.

[Bentley 1975] to facilitate a fast lookup.

The first sense, **best_vis_angle**, was used by adding two rules to the brain which turn the agent toward the best angle if there are no enemies currently visible. The second sense, **position_goodness** was used to direct the agent to stop and wait for enemies in a location (also known as "camping") if it has a high **position_goodness** and is not too close to other friendly agents (to stop agents grouping together in one spot). The combination of these two behaviours results in the agents occasionally camping in a strategically valuable area while facing in the direction from which an enemy is most likely to come.

The contests between the agents took place in three virtual worlds (See Figure 8f to 8h). Since we wished to evaluate the effect of one environmental variable or sense at a time, we constructed worlds which did not give an undue advantage to the normal agents. For example, we broke up outer circuits on the edge of the world since normal agents tended to congregate on them and surprise the more spatially aware agents looking inward. While it would be easy enough to use additional environmental variables to eliminate this advantage, our intention was to assess the utility of the extra spatial information to agent behaviour, not to design an optimal agent.

For each world two different contests took place: A team contest were a group of five "smart" agents competed against a group of five "normal" agents and a one-on-one contest where one "smart" agent competed against one normal agent. Each contest was then repeated with the starting positions reversed in order to ensure that the virtual worlds did not unduly favour one side. The results of these contents are listed in Table 1.

In each case, the agents with awareness of path intersection points won the contest. Additionally, the more spatially aware agents "camped" near positions with a high **position_goodness** sense, managing to surprise normal agents who wandered across areas of high traffic. Since the agents react to sensory information, they have no higher-level planning behaviour besides "camping."

6.3 Complexity of Data Structure Queries

In each scenario, the more spatially aware agents queried the framework in realtime. To lookup curvature information for a racing agent, a quadtree lookup was performed followed by a binary search, yielding an $O(\log N)$ complexity. To lookup up a point sample for the warring agents, a nearest neighbour search was performed on a kd-tree, which again produces $O(\log N)$ complexity.

7 Conclusion

The spatial awareness framework presented in this paper introduces a new system which allowing agents to query the geometric qualities of the space that they are operating in, namely width, curvature and connectivity. To our knowledge, this is the first system which automatically extracts such qualitative geometric information from the environment. An agent crowd simulation system was implemented to test whether awareness of these qualities could improve the performance of agents within a virtual world.

Even though the agents were primarily designed to react to sensory information and only implemented the most basic of planning capabilities, their effectiveness was increased by providing geometric information. We also showed how the connectivity information derived from our spatial awareness framework could be combined with the commonly used visiblity information.

8 Acknowledgements

This research was funded by the National Research Foundation of South Africa under grants GUN2053402 and GUN66423.

References

AMENTA, N., BERN, M., AND KAMVYSSELIS, M. 1998. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 415–421.

BARBER, C., DOBKIN, D., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software 22*, 4 (Dec), 469–483.

BENTLEY, J. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM 18*, 509–517.

BHATTACHARYA, P., AND GAVRILOVA, M. L. 2007. Voronoi diagram in optimal path planning. In *ISVD '07: Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering*, IEEE Computer Society, Washington, DC, USA, 38–47.

CHRYSANTHOU, Y., AND SLATER, M. 1995. Shadow volume bsp trees for computation of shadows in dynamic scenes. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 45–50.

CORNEA, N. D., AND MIN, P. 2007. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on Visualization and Computer Graphics 13*, 3, 530–548. Member-Deborah Silver.

FUCHS, H., KEDEM, Z. M., AND NAYLOR, B. F. 1980. On visible surface generation by a priori tree structures. In *SIGGRAPH '80: Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 124–133.

GUIBAS, L., HOLLEMAN, C., AND KAVRAKI, L. 1999. A probabilistic roadmap planner for flexible objects with a workspace medial axis. In *Proceedings of the IEEE International Conference on Intelligent Robots.*

Holleman, C., and Kavraki, L. 2000. A framework for using the workspace medial axis in prm planners. In *Proceedings of the International Conference on Robotics and Automation*, 1408–1413.

Klir, G. J., and Yuan, B., Eds. 1996. *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh.* World Scientific Publishing Co., Inc., River Edge, NJ, USA.

Lidén, L. 2000. The integration of autonomous and scripted behaviour through task management. In *Artificial Intelligence and Interactive Entertainment, AAAI Spring Symposium.*

Mizumoto, M. 1998. Defuzzification. In *Handbook of Fuzzy Computation.* IOP Publishing Ltd.

Morgan, D. 2003. Algorithmic approaches to finding cover in three-dimensional, virtual environments. Masters thesis, MOVES Institute.

Ogniewicz, R. 1994. A multiscale mat from voronoi diagrams: The skeleton-space and its application to shape description and decomposition. *Aspects of Visual Form Processing*, 430–439.

Ogniewicz, R. 1994. Skeleton-space: a multiscale shape description combining region and boundary information. In *1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 746–751.

O'Neill, J. 2002. Efficient navigation mesh implementation. *Journal of Game Development*.

Pottinger, D. 2000. Terrain analysis in realtime strategy games. In *Proceedings of Computer Game Developer Conference.*

Teller, S. J., and Séquin, C. H. 1991. Visibility preprocessing for interactive walkthroughs. *Computer Graphics 25*, 4, 61–68.

van der Sterren, W. 2001. Terrain reasoning for 3d action games. *Game Programming Gems 2*, 307–316.

van Waveren, J.-P. 2001. The quake iii arena bot. Masters thesis, Delft University of Technology.

Voronoi, G. 1907. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. *Journal für die Reine und Angewandte Mathematik 133*, 97–178.

Zadeh, A. 1996. A rationale for fuzzy control. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A. Zadeh.* World Scientific Publishing Co., Inc., River Edge, NJ, USA, 123–126.