Real-time Global Pedestrian Navigation with Graphics Hardware using FLAME GPU

Robert Chisholm Daniela M. Romano Lewis Gill

Department of Computer Science
Regent Court
211 Portobello
S1 4DP
Sheffield, UK
{R.Chisholm,D.Romano,L.Gill}@sheffield.ac.uk

ABSTRACT

A crowd is a large group of people gathered together in a disorganised way that appears to act as one organism. Agent-Based modelling is well suited for simulating the actions and interactions of large number of individuals and supporting crowd normative understanding. Considering a large population increases the validity of the emergent characteristic observed. FLAME GPU is an agent-based modelling framework well suited for simulating massive populations on a single machine exploiting the parallel power of the Graphic Processing Unit. Here we presents a review of the most known agents' navigation techniques and their implementation on the graphic processing unit with the agent-based framework FLAME GPU. Two known techniques: Roadmaps and Vector Field navigation, are compared with a new technique: Attractors and Repulsors, inspired by robotics research. The findings show that navigation using Roadmaps is versatile, with lower memory usage, able to handle three dimensions, although needs a full recalculation as the environment layout changes, and performs slower than Vector Field navigation when considering the average kernel execution time. Vector Field navigation allows dynamic changes in the layout at run-time, and has the fastest average kernel execution time that remains stable regardless of map complexity and only minimally increases at the increase of the populations size. The new technique, Attractor and Repulsors, although well suited for very large and dynamic environment, sometimes traps pedestrians in local minima and has the worst average kernel execution time.

Categories and Subject Descriptors

I.6.8 [**Types of Simulation**]: Parallel; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Performance

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

Keywords

Agent-based Simulation, Simulation Techniques, Tools and Environments, Navigation Techniques, Crowd Simulation, Roadmap, Vector Field, Attractors and Repulsors, Comparison

1. INTRODUCTION

Simulating realistic pedestrians is a challenge to both the fields of computer graphics and social science, and also has important applications in the design and testing of layouts and architectural plans in Engineering. Many pedestrian models also share similarities with path planning in Robotics research. The majority of goal-orientated pedestrian and crowd modelling approaches delegate the global navigation and local collision avoidance to separate algorithms, only a small number choose to merge the navigation functionality with the agent's behaviour. There has been significant research observing the interaction of pedestrians within crowds, and attempting to reproduce the visible behaviour with artificial models, however more work is needed to validate and compare the various solutions. One of the most used validation techniques is the use of video data to 'calibrate' the pedestrian model, however most research concerning pedestrian modelling is qualitative in nature and unable to provide side by side quantitative comparisons to the alternative approaches developed by other institutions.

Agent-Based Modelling (ABM) is well suited to the modelling of crowds. A crowd is a large number of things or people considered together as one entity. ABM is also very well suited to study complex systems considering a bottom-up approach .[1 3 2 32 34] Whether concerned with a crowd of boids or pedestrians, the autonomous nature of the individual agents provides the basis for an Agent-Based Modelling approach.[33] Graphics processing hardware is particularly suited to the field of ABM.[31] Whilst their architecture was initially designed to render thousands of polygons simultaneously, the rise in general purpose graphics processing unit (GPGPU) programming allows their parallel architecture to be redirected towards new avenues. Simultaneously performing thousands of graphics operations can be replaced with the simultaneous processing of thousands of agents. Researchers have shown that ABM algorithms transferred from non-GPU implementations to GPU allow the simulation of much larger agent populations, and consequently

permits the consideration of a larger areas amount of inter-agent interaction.[28] With agent data already on the GPU, rendering the agent populations also benefits, reducing the data transfer necessary to render each frame. This is to a great advantage when considering the computation and rendering in real-time of Agent-Based Models, in a decision support system fed by real-time data, cyber-physical systems,[5] where minimising execution time is paramount.

Global navigation is concerned with providing pedestrian agents, the means to reach a goal location within the simulated area. There are two common approaches which global navigation techniques can be classified to; Vector Field whereby pre-computed forces representing the path to each goal are stored for recall at runtime;[13] and Roadmap navigation whereby the map's layout is represented as a cell portal graph that agents use to navigate via way-points to reach their goal at runtime.[37] A model from each of these approaches has been implemented for quantitative evaluation as part of this study. A new model, inspired by the potential field path planning models used in robotics, Attractor and Repulsors, is introduced and compared to the other two well known algorithms.

The organization of the rest of the paper is as follows: Section 2 provides a review of related work; Section 3 explains a qualitative evaluation of the three approaches to global navigation and how this lead to the implementation of the three chosen navigation models; Section 4 presents the results used for a quantitative evaluation of the implemented navigation models; Section 5 discusses the findings. Finally, Section 6 provides the conclusion and a direction for further research.

2. RELATED WORK

2.1 The Need for a GPU implementation in Real-time Computing

Modellers and decision makers are increasingly looking towards larger simulations. The motivation is clear, more realistic simulations requires larger populations, often of multiple types, as the validity of emergent characteristics is dependent on both: the accuracy of the behaviour modelled and the population sizes. In addition there is a need to forecast behaviours faster than the wall-clock time in computer-based control systems, or cyber-physical systems.[5] Real-time computing and processing of large data is the next step forward in computing.[19] Run-time costs are presently inhibiting the effective use of ABM as forecasting tool. Since the introduction of the first behavioural model by Reynolds (1987),[26] it has been clear that large behavioural models require high levels of computation. Since several hardware based solutions have been investigated to date to provide real-time computations. For example Uhrmacher (2000)[40] proposes a simulation to optimise strategies or interactions amongst multi-agent systems in a serial implementations. Zhou & Zhou (2004)[42] report a parallel algorithm to simulate flocking of 512 individuals in a more efficient manner than similar serial implementations, using a parallel cluster of 16 processors. Quinn et al. (2003)[24] have simulated 10,000 pedestrians at 50fps using a parallel algorithm on a multi-computer architecture. Another approach used to improve simulation speed is the adoption of spatial data structures, the allow only neighbouring individuals to be computed (Shao & Terzopoulos, 2007)[35]; also Reynolds (2006)[25] use of a spatial structure has facilitated the simulation of 15,000 individuals on PS3 hardware. Passos et al. (2008)[21] have simulated 1 million flocking boids at nearly 30fps on a with spatial partitioning on the GPU using CUDA technology, which was state of the art technology in 2007. Erra, De Chiara et al. (2004)[6] present a massive simulation of 32,768 agents on the GPU at 3.45fps, and Erra et al. (2009)[7] have simulated 65,536 individuals at 60fps USING a GeForce 8800GTX. Richmond & Romano (2008)[30], on a GeForce 8800GT using FLAME GPU, also simulated 65,536 agents at 60fps, however they simultaneously rendered the behaviour as a 3D visualization; where Erra et al. (2009)[7] had the same number of entities, without 3D visualisation. The rendering of the population has an effect on the performance of the system with on average a 5-10% drop in frame rate. As hardware improves so size of the population that can be simulated increases. The advantage of using the FLAME GPU framework rather than an bespoke GPU simulation approach, beside the 3D graphical visualization of results, is that the framework is able to abstract much of the complexity of GPU programming, providing an easy to use XML based interface to build families of simulations quickly.

2.2 Models

The majority of global navigation models can be placed into three techniques: Naive Potential Field, Vector Field and Roadmap. There are also a small number of hybrid models, which combine the operations of navigation and collision avoidance into a single model.

2.2.1 Naive Potential Field

The term Potential Field, is used in robotics research, whereby a robot knows the direction it must travel to reach it's goal, and is able to detect obstacles in it's path. Using this knowledge of goal and obstacle locations, a velocity is produced combining the virtual repulsive forces applied from each obstacle, and the attractive force of the goal [14]. Such a model is suitable for implementation in an agent-based system, as shown in Section 3, and very similar models are the basic idea behind collision avoidance models often used in pedestrian simulations.[10] Separating it from collision avoidance of dynamic obstacles (e.g. pedestrians), better enables its assessment as a global navigation model. This form of navigation is naive, without heuristics to perform higher level path planning, and has been observed to allow the navigating robot to become trapped at local minima's[15].

2.2.2 Vector Field

Known by various names, these models are largely an evolution of the potential-field models, utilising preprocessing to shift the navigation process from runtime, to the map compilation stage. This preprocessing allows the models to carry out a greater level of force calculation and path planning, than would be feasible to execute at runtime. These forces are usually stored in cells within multiple discrete grids, each representing a different goal or grouping of collisions, which can be recalled at runtime by converting an agents continuous location to a discrete location. These maps can require a large amount of storage when representing large or detailed areas.

Wang & Chirkjian (2000)[41] describe an artificial poten-

tial field model whereby steady-state heat transfer is modelled to provide path planning for a robot. Obstacles within their workspace give covered cells a low conductivity to heat, whereas free cells have a much higher conductivity. The model then works by treating the goal as a sink that pulls in heat, creating heat flux lines that fill the workspace directed towards the goal.

Tecchia et al. [38] detail a model assembled from four discrete layers, each layer providing a different behaviour; inter-collision detection, collision detection, behaviour and callback. Whilst not providing explicit navigation within the model, the behaviour layer encodes actions such as reaching a bus stop. The collision and inter-collision layers are used for providing information to prevent collisions between static and dynamic obstacles respectively.

Mannicam (2003)[18] presents a hexagonal lattice grid traffic model, and Jun et al. (2009)[12] a rotated hexagonal lattice mode for pedestrian flow, whilst these models perform in a discrete space whereby agents move between hexagons each tick. Their research suggests that a hexagonal lattice better represents the arrangement of pedestrians within crowd than a square lattice.

Chenny (2004) describes a technique, flow tiles,[4] for designing and representing velocity fields through combining a discrete set of tiles. Each flow tile contains a small field, combinations of these can be combined to produce larger flows. Tiles are constructed so that corners and edges combine to ensure continuity of flows. Their use of a discrete set of tiles allows reduced memory consumption. They demonstrated both static and dynamic fields created through flow tiles.

Shao et al. (2005) [35] present a model utilising a hierarchical data structure, that takes advantage of quad-tree's, to spatially partition the space to be navigated. This allows differing levels of detail to be represented across the navigable area, whilst reducing memory consumption. Such a method is greatly beneficial to maps combining large open areas and smaller detailed areas, that would otherwise contain many identical tiles repeated across the large open areas.

Jin et al. (2008)[11] use radial based functions in their model to generate continuous vector fields. Anchor points are placed to redirect the field which can then be recalculated in real-time. Whilst not demonstrated for goal orientated navigation, their model presents an example of a continuous vector field approach which is unique from the other vector field models surveyed.

Kharmakharm et al (2010)[13] demonstrate a GPU implementation of a discrete vector field model. Their model utilised a single static obstacle collision layer, and a single navigation layer per exit. Dynamic collisions were modelled using a velocity obstacle method. A propagation algorithm is used to automatically generate the navigation fields. Their use of GPGPU allowed the simulation of over 250,000 pedestrians concurrently whilst retaining over 30 simulation ticks per second.

Patil et al. (2011)[22] describe a model for combining navigation fields collected from different interfaces; sketch, extracted from crowd video footage. Their model ensures smooth, least effort paths, free from local minima that might otherwise cause agents to become trapped. Dynamic collisions in their model were handled with a reciprocal velocity obstacle method.

2.2.3 Roadmap

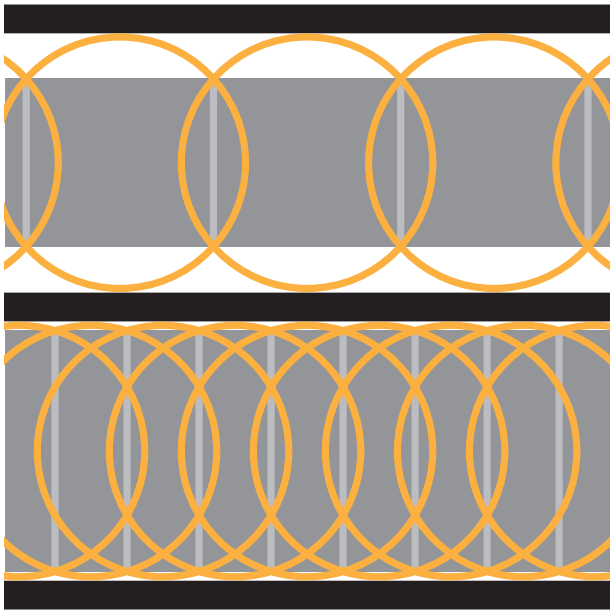


Figure 1: The difference in vertex density affecting coverage within the model described by Pettré et al.

These models consist of representing the map to be navigated as a roadmap, cell-portal-graph or navigation mesh, either searching the graph at runtime, treating it as a routing table or assigning agents preselected paths from the graph.

Lamarche & Donikian (2004) [16] describe a two dimensional spatial partitioning technique. Firstly the environment is partitioned using constrained delaunay triangulation, then again using the computed shortest distance between walls and corners. The now triangular cells are sorted so that they may be iterated according to the length of their free segments in ascending order. From here two cells are only merged if they form a convex cell and the shared segment is a greater length than each individual segment of the resulting cells. The use of convex cells, ensures there is always a linear path traversing each cell.

Pettré et al. (2005)[23] present a model, that uses navigation corridors to generate a graph able to cover multi-layered and uneven terrain. Geometrically they represent vertices as cylindrical areas, whereby the intersection of two vertices produces a rectangular gate which represent the edges of the graph. This method however does create a trade-off of graph complexity and coverage quality, whereby a straight corridor must be represented by numerous cylindrical vertices. As vertices are packed more tightly they increase the coverage while adding redundant edges to the graph, this is illustrated in Figure 1. For their simulation they use Dijkstra's algorithm to generate paths during a preprocessing stage, whereby they also remove a small number of edges from the selected paths.

Lerner et al. (2006)[17] use a simple two pass heuristic algorithm to compute cells-and-portals partitions of two dimensional architectural interiors. They show their algorithm produces more efficient partitions than a standard BSP approach. They did however identify some misaligned cells which would not have been created by a manual designer, these imperfections should not affect the efficiency of the graph though as they do not increase the number of portals.

Guy et al. (2010)[9] utilise a dynamic energy roadmap, whereby a least energy function is utilised to minimise energy expended by agents. Each iteration the edges of the roadmap are updated, using the average velocity across the edge to calculate the energy expended crossing the edge. This then allows agents to select a route which minimises the energy expended. Specifically agents attempt to travel the shortest route to their destination and to maintain their preferred speed. The local collision avoidance model used (reciprocal velocity obstacle) returns a set of permissible velocities that will not result in a collision, these are then used in combination with the roadmap to select the velocity expected to expend the least energy. This allows agents to adapt to congestion in a method similar to that observed in crowds. They provide no details of the roadmap used by their model.

Olivia & Pelechano (2013)[20] present a novel GPU based implementation for automatically producing efficient navigation meshes from complex 3D and multi-layered scenes. The scene is first voxelised to extract walk-able layers, a high resolution render is then performed using a fragment shader to obtain a 2D floor plan of each layer. Finally a convex decomposition of each layer is calculated, these are then linked to produce the completed navigation mesh.

2.2.4 *Hybrid Models*

A small subset of models combine global navigation and collision avoidance into a single model.

Treuille et al. (2006)[39] manage this using a fluid dynamics approach, it is assumed that every agent within a crowd with the same intention has similar desires towards reducing their travel time, discomfort and other factors. This assumption allows them to calculate the potential fields for a single agent within a crowd and apply the result to all of the agents which have the same intention. To compensate for diversity in walking speed, large crowds are split into smaller groups, each with differing speeds and behaviours. Additionally a minimum distance between agents has to be enforced, otherwise it is possible for agents to intersect.

Sud et al. (2008)[36] use voronoi diagrams to combine global navigation and local collision avoidance requirements into a single data structure they call the multi-agent navigation graph (MaNG). A MaNG is produced by performing a union of the first and second order voronoi diagrams. As MaNG's capture pairwise proximity information, computed paths directly result in collision avoidance. Extracting each agents path from a MaNG is more efficient than computing them individually. Paths generated by a MaNG do however require smoothing and the application of a right hand collision rule.[1] This work is likely a continuation of the earlier paper by Sud et al[37] which also uses voronoi diagrams towards building a roadmap inclusive of local collision avoidance.

3. MODEL SELECTION

To evaluate the potential of real-time pedestrian navigation models when coupled with a GPGPU implementation, one algorithm from each of the three main groups described in Section 2.2 was implemented using FLAME GPU[29 27]. Each model was implemented as a crowd of homogeneous pedestrians, configured to use an identical local collision avoidance model, velocity obstacle,[8] to navigate agents across two dimensional maps.

[1]If two agents are to collide, each should go to their right hand side to avoid the collision.

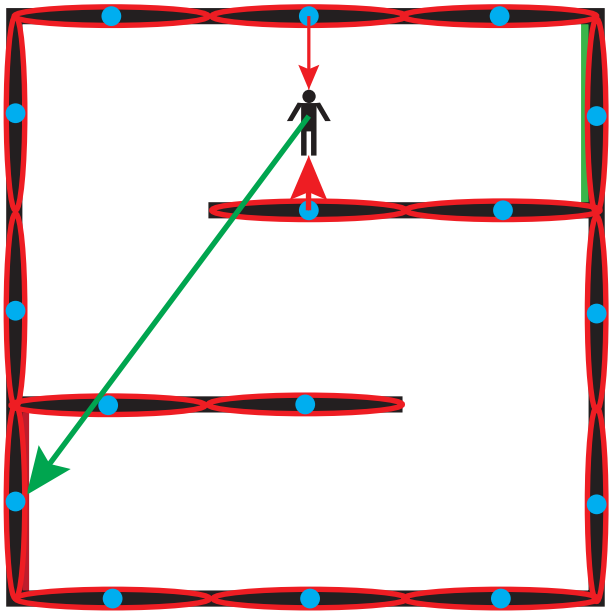


Figure 2: An illustration of the forces that may be applied to an agent by the Attractor and Repulsors model.

3.1 Attractor and Repulsors

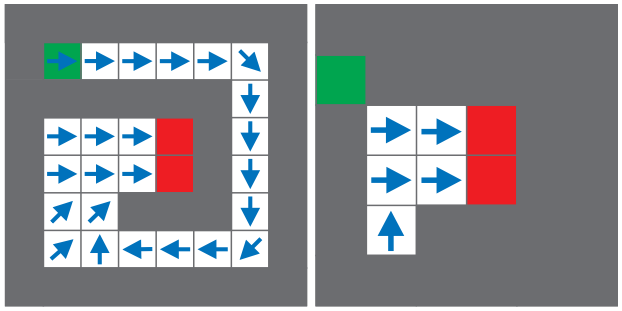
The Naive Potential Field approach has been used to created a new navigation method for agents, named Attractor and Repulsors. This navigation model trades the high memory usage common within the discrete structures of Vector Field approaches, for a high amount of runtime processing.

Each obstacle is represented as a Repulsor, using the inverse square law to modulate the strength of the force applied to the agent. At each iteration the agent considers all visible obstacles, that is those with which the agent might interact for example within 5 metres, and their force vectors is calculated using the inverse-square law of their separation. These forces are then summed to calculate the static collision force. The goal force (Attractor) on the agent is then calculated by subtracting the position of the goal from the position of the agent. These two force vectors are then each multiplied by an additional weight, which may be adjusted at runtime to balance their effects, before summing with the force produced by the dynamic collision model, to calculate the force to act on the agent. Figure 2 provides an illustration of this model.

The nature of this models navigation can cause agents to become trapped at singularities[15] whereby to reach the goal the agent must be given an additional force to walk away from it temporarily. Although of easy implementation on large scale maps, it is only suitable for a small subset of maps unless manual intervention is used, to include additional forces allowing agents to better negotiate areas where they would become trapped.

3.2 Vector Field

The Vector Field model implemented is based on the GPGPU model described by Karmakharm et al (2010)[13].



(a) (b)

Figure 3: (a)A Vector Field map with suitable grid dimensions. (b)A Vector Field where the low grid dimensions cause necessary walk-able cells to be treated as obstacles.

The model is implemented such that each goal has it's own navigation layer and there is also a single static collision layer. These layers are each discrete grids, Karmakharm et al show how performance decreases as the grid size increases, however this performance difference becomes less significant as the pedestrian population increases. Using a larger grid allows greater detail to be modelled at the cost of a larger memory footprint[2], this implementation uses a 256x256 grid. Shao and Terzopoulo's (2005)[35] quad-tree vector field mentioned in Section 2.2.2 could alleviate this issue of a single level of detail over the entire navigable area. However access to vectors would require tree traversal, as opposed to the faster direct memory access used by other Vector Field models.

To fill these discrete grid with vectors that can be used by agents, a wavefront propagation algorithm is used. All obstacle or goal cells are added to a primary list, of the remaining empty cells, any adjacent to the cells in the primary list that have not been assigned a vector are added to a secondary list. The vector for each of the cells in the secondary list is calculated as the normalised sum of the vectors pointing away from any adjacent cells in the primary list (or in the case of goal layers towards their adjacent cells). In the case of the static collision layer, this value is then multiplied by the iteration count, so that the force weakens for agents further from the obstacles. The secondary list then replaces the primary list so that the algorithm can be repeated until every cell has been assigned a vector. The created goal layers have diagonal convergence issues, whereby paths converge tightly. This is reduced by applying a back-flow smoothing technique, where vectors are propagated backwards, setting their value to a normalized sum of their neighbours to be visited the next iteration. This back-flow smoothing operation must be repeated several times to achieve the best results, however over application can also be detrimental to the navigation vectors produced.

During execution agents are then able to convert their continuous location to a discrete coordinate with which to extract their relevant goal and static collision vectors from the previously generated layers. This can then be combined with the output from the dynamic collision model to produce the final force to be applied to the agent.

The use of a discrete grid to represent a potentially continuous environment can lead to coverage issues, especially if the dimensions of the discrete grid align badly with the environment, causing walls to be misrepresented and possibly losing finer paths. Figure 3 provides a simplified view of how resolution issues may affect a Vector Field model.

3.3 Roadmap

The model used for roadmap generation is an implementation of the navigation corridors model described by Pettré et al (2005)[23]. This model may provide reduced coverage compared to the model by Lamarche and Donkian (2004)[16], however it is more viable for use modelling real world environments due to its support of multi-layered terrain, whereas both Lamarche and Lerner's algorithms are constrained to modelling flat areas. Olivia and Pelechano's (2013) algorithm was not selected, as the complexity of it's implementation far exceeded that of other algorithms considered and would have required significant additional commitment of time to implement and thoroughly test.

To construct the navigation graph within a plane, first the medial axis of the walk-able area must be calculated, this is the set of all points which have more than one closest point on the boundary of the walk-able area. A simple brute force approach was used to find the first point, after which a flood-fill technique was used to identify remaining points. This is possible as all points along the medial axis should form a single cohesive body. Each point from the medial axis can then be used to create a vertex within the graph, whereby the radius of each vertex is it's clearance to the walk-able area. This creates a graph of overlapping circles, whereby the intersection of two neighbouring circles marks an edge an edge. Such a graph contains many redundant vertices which must then be removed to ensure more efficient graph traversal, however it is also important to consider that the further the circles are separated, the more that the coverage quality of the graph is reduced. To reduce redundant vertices within the graph the following algorithm is used to select vertices;

1. Select the point along the medial axis with the greatest clearance from the set of possible points
2. Create a vertex from the selected point
3. Remove all points encapsulated by the vertex created in step 2 from the set of possible points
4. Return to step 1 until the set of possible points is empty

Now that only the desired vertices remain, they are simply tested for intersection to build the connectivity of the graph and assign costs of the distance between vertices to each edge. To maximise performance in navigation of a static map (the cost of routes was not required to adapt to congestion or similar), a routing table was calculated and stored onto the edges of the graph, enabling agents to more directly lookup their path as opposed to performing graph searches at each vertex.[3] The bellman-ford algorithm was used for building the routing table.

[2]If each cell stores two floats as the direction vector for each of 10 exits (80 Bytes). 5.2MB will be used to store this information in a 256x256 grid, this increases to 83MB for a 1024x1024 grid.

[3]Placing the routing table on to the edges of the graph, allows them to become the vertices of the graph as is more

At runtime an agent is then able to be assigned a vertex from which they are birthed and a goal vertex which they are to target. All edges are then considered which connect the agents current vertex, keeping track of only the edge which has the lowest cost between the current vertex and the goal vertex in the correct direction. The selected edge is then used to identify the next vertex to be targeted as a waypoint along the agents path to their goal. The agent may simply target this until it has been reached it, whereby it performs another routing table lookup to find it's next waypoint.

4. EVALUATION

As emergent phenomena is primarily attributed to local navigation models[10], evaluation of the global navigation is largely restricted to performance metrics.

All benchmarks were performed using an GeForce 560ti GTX graphics card which has 384 CUDA cores and the Nvidia NSight CUDA profiler. Use of a CUDA specific profiler allows the timing of only the kernel launches used in global navigation to be measured, providing measurements independent of unrelated overheads such as rendering or collision avoidance.

4.1 Pedestrian Population Size

As the population size increases, the navigation function is called more frequently, this benefits algorithms that perform pre-processing, allowing them reduced processing per pedestrian agent.

The growing demand for real-time simulations for example of large urban spaces at times of peak traffic and pedestrians rush hours, provides a need for global navigation models that can scale effectively to meet pedestrian agent population demands.

Each of the three navigation algorithms being evaluated has a complexity of $O(1)$ for their navigation lookup functions relative to the size of the pedestrian population. Therefore an increase in pedestrian population can be observed to linearly affect the speed of execution.

To visualize this effect, each model was executed with varying population sizes, to observe the linear increase in execution times. For the purposes of this evaluation it is important to only consider the gradient of increase, rather than the comparative times between each model, as map details affect the complexity of each models representation independently. This data is shown in Table 1 and Figure 4.

4.2 Map Complexity

Each model represents maps in a significantly different manner, as such providing a truly unbiased evaluation of their performance across maps of varied size and design is infeasible.

The Vector Field model represents maps as discrete grids. The conversion of an agents continuous position to a cell within this discrete grid is all that is required before performing each agents velocity lookups (Figure 3a). Therefore map complexity should have minimal effect on the speed of execution unless the size of the discrete grid is adjusted.

The Attractor and Repulsors model represents maps as static obstacles, delineating the borders of the objects to be avoided with sparse static emitters of force each applying a force to any agent in range (Figure 2).

This allows representing a map with only a sparse number of static force emitters, far less than the static agents of a grid as in Vector Field Navigation; however as the number of obstacles increases, the number of static force emitters that the agent must consider increases.

The Navigation Corridors model represents maps as a graph produced from the connectivity of circles placed within the walk-able area. This means that as an area of the map deviates further from a circular shape more vertices must be placed to model the area. This becomes particularly evident in narrow corridors which contain many redundant vertices. Increasing the number of vertices within the map increases the number of edges which each agent must search when finding its next waypoint.

To illustrate the difference in execution speed for varying maps across each model, the table and maps presented in Table 2 contains the average navigation kernel times for simulations containing 1000 pedestrians on the four maps shown in Figure 5. These times show the Attractor and Repulsors model consistently requiring the highest processing time, as expected some agents became trapped in the maps b, c and d (Figure 5) as maps were automatically generated with no manual intervention. The navigation corridors model was

suitable from a pedestrian navigation stance. (Agents walk between the intersections of the circles, rather than the circles)

Average Kernel Time (μs)		Model		
		Vector Field	Navigation Corridor	Attractor and Repulsors
Population Size	125	120.454	1,046.852	5,536.107
	250	126.724	1,051.795	5,572.018
	500	132.779	1,057.848	5,791.169
	1000	141.062	1,096.028	6,796.612
	2000	159.818	1,095.419	7,852.519
	4000	275.378	1,156.800	10,914.817
	8000	391.330	1,456.328	16,706.929

Table 1: Average kernel times for each models global navigation kernel(s).

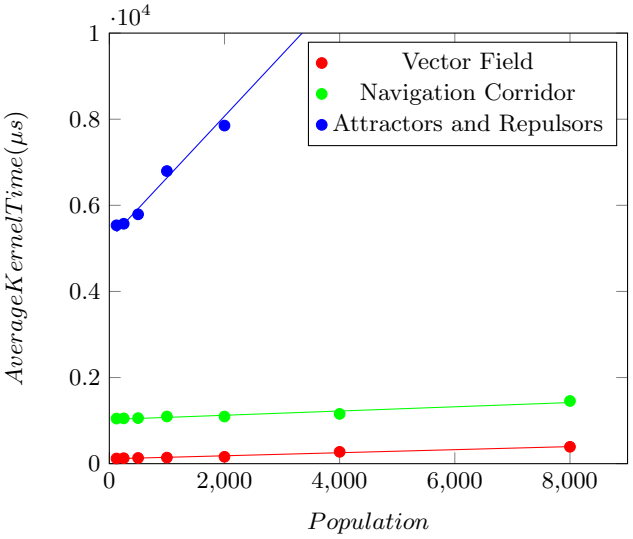


Figure 4: Graph plotting average kernel time against pedestrian population for each model.

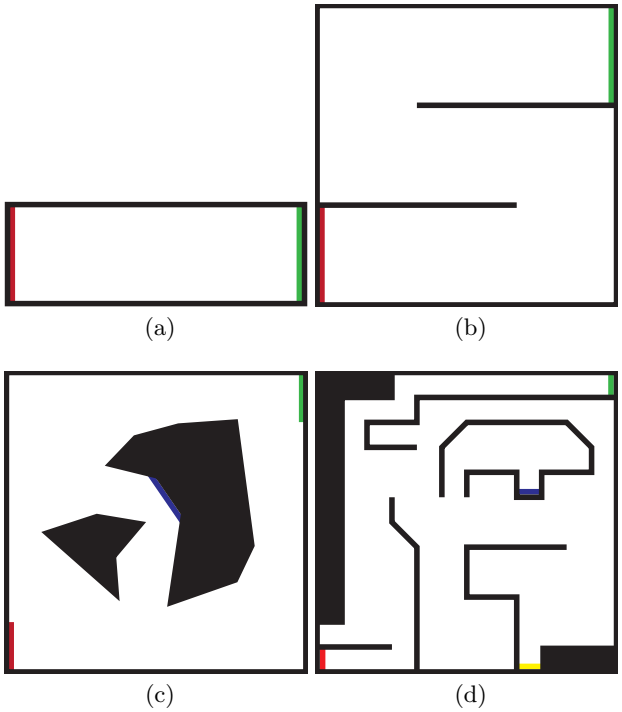


Figure 5: The maps used for the benchmarks in Table 2.

Average Kernel Time (μs)		Model		
		Vector Field	Navigation Corridor	Attractor and Repulsors
Map Reference	a	137.793	184.319	2,293.084
	b	138.135	432.023	3,257.583
	c	139.620	643.977	4363.023
	d	139.463	3,966.286	10,296.624

Table 2: Average kernel times for each models global navigation kernel(s).

only ˜4 times slower than the Vector Field model in map c, however this grew to 28 when map d was executed. All the narrow corridors within map d, created many redundant edges within the graph, this increase in edges and vertices thereby increases the number which must be checked by each agent per iteration. The Vector Field model kept a fairly stable time throughout the maps as was expected.

5. DISCUSSION

Naive potential field models, e.g. Attractors and Repulsors implementation, were shown to have an heavy processing overhead, which is far outdone by Roadmap and Vector Field techniques, both which provide a heuristic form of navigation. Whilst the evaluated Attractor and Repulsors model may provide a memory-light alternative to Vector Field, it is quickly outperformed by the other models, due to its naive path planning, when pitted against more complex maps, Roadmap techniques are best suited in these cases.

Vector Field models have been shown to far outperform other techniques in terms of average kernel time, except in extremely simple maps. However in maps whereby the level of detail varies significantly, a redundant repetition of Vector Fields can consume significant amounts of memory. Existing Vector Field models only provide support for two dimensional maps. The lack of research towards multi-layered Vector Field techniques, suggests that it would be challenging to apply current Vector Field techniques to a three dimensional terrain, whereby multi-layered walk-able areas are present. Roadmap models instead have already been shown to work with three dimensional terrains.

Roadmap techniques have proven the most versatile, with the potential to provide efficient coverage in both two and three dimensions. Whilst graphs generated via the algorithm described by Olivia & Pelechano (2013)[20] were not compared directly to the performance of Vector Field approaches, the state of the Navigation Corridors algorithm evaluated leaves much potential for further performance enhancement. By replacing the used navigation portal graphs with Olivia & Pelechano's technique, the size of graph used can only be reduced, which benefits graph searches. Additionally, further preprocessed information could be stored within the routing table, this would further reduce runtime processing costs if there is no cause to update costs across the graph.

Roadmap techniques are the most versatile in terms of quality and efficiency of coverage, memory usage and ability to handle a three dimensions, whilst only performing slightly slower (varying from 1.5-28 times to the speed of Vector Field) than the Vector Field model, this difference should be negligible until incredibly high agent numbers are simulated. The downfall of Roadmap model is the need to re-calculate the whole navigation graph if the map layout changes in real-time (e.g. a door is opened or closed, a wall falls down), a case that is instead well handled by the other two navigation techniques.

6. CONCLUSION

This study has presented a review of pedestrian global navigation techniques, with quantitative evaluation across an exemplar model from each of three general approaches. Also considerations with regards to the map's layout have been made, showing how specific layouts are better at minimizing the potential pitfalls of a model. In completing this, the benefits of each of the common approaches to global navigation modelling have been identified. Further research would be well spent exploring the most efficient techniques for concurrently searching and updating cell-portal-graphs, to provide a basis for adaptive navigation whereby pedestrian agents actively avoid visible congestion.

7. ACKNOWLEDGEMENTS

The research presented here is a summary of the work carried out in several projects including: DSTL CDE CAPIRE, Royal Academy Secondment to Costain, EPSRC ITaUU secondment to Costain. The authors thank all sponsors for their support.

8. REFERENCES

[1] M. Burkitt, D. M. Romano, D. C. Walker, and A. Fazeli. 3d modelling of complex biological structures: the oviduct. In *Theory and Practice of Computer Graphics*, pages 255–262. The Eurographics Association, 2010.

[2] M. Burkitt, D. Walker, D. Romano, and A. Fazeli. Using computational modeling to investigate sperm

navigation and behavior in the female reproductive tract. *Theriogenology*, 77(4):703–716, 2012.

[3] M. Burkitt, D. Walker, D. M. Romano, and A. Fazeli. Modelling sperm behaviour in a 3d environment. In *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*, pages 141–149. ACM, 2011.

[4] S. Chenney. Flow tiles. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 233–242. Eurographics Association, 2004.

[5] E. COMPUTING. Cyber-physical systems. 2009.

[6] U. Erra, R. De Chiara, V. Scarano, and M. Tatafiore. Massive simulation using gpu of a distributed behavioral model of a flock with obstacle avoidance. In *Proceedings of vision, modeling and visualization*, volume 2004, 2004.

[7] U. Erra, B. Frola, V. Scarano, and I. Couzin. An efficient gpu implementation for large scale individual-based simulation of collective behavior. *High Performance Computational Systems Biology, 2009. HIBI'09. IEEE International Workshop on*, pages 51–58, 2009.

[8] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[9] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, and D. Manocha. Pledestrians: A least-effort approach to crowd simulation. *Symposium on Computer Animation, 2010 Eurographics/ ACM SIGGRAPH*, pages 119–228, 2010.

[10] D. Helbing and P. Molnár. Social force model for pedestrian dynamics. *Physical review E*, 51(5), May 1995.

[11] X. Jin, J. Xu, C. C. Wang, S. Huang, and J. Zhang. Interactive control of large crowd navigation in virtual environment using vector field. *Computer Graphics and Applications, IEEE*, 28(6):37–46, 2008.

[12] L. Jun, H. Mingook, P. Ilsub, and J.-H. Chung. The rotated hexagonal lattice model for pedestrian flows. In *Proceedings of the Eastern Asia Society for Transportation Studies*, volume 7, 2009.

[13] T. Karmakharm, P. Richmond, and D. M. Romano. Agent-based large scale simulation of pedestrians with adaptive realistic navigation vector fields. *Theory and Practice of Computer Graphics, The Eurographics Association 2010*, pages 66–74, 2010.

[14] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[15] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1398–1404, 1991.

[16] F. Lamarche and S. Donikian. Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. In *Computer Graphics Forum*, volume 23, pages 509–518. Wiley Online Library, 2004.

[17] A. Lerner, Y. Chrysanthou, and D. Cohen-Or. Efficient cells-and-portals partitioning. *Computer Animation and Virtual Worlds*, 17(1):21–40, 2006.

[18] S. Maniccam. Traffic jamming on hexagonal lattice. *Physica A: Statistical Mechanics and its Applications*, 321(3):653–664, 2003.

[19] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.

[20] R. Oliva and N. Pelechano. Neogen: Near optimal generator of navigation meshes for 3d multi-layered environments. *Computers & Graphics*, 37(5):403–412, 2013.

[21] E. Passos, M. Joselli, M. Zamith, J. Rocha, A. Montenegro, E. Clua, A. Conci, and B. Feijó. Supermassive crowd simulation on gpu based on emergent behavior. In *Proceedings of the VII Brazilian Symposium on Computer Games and Digital Entertainment*, pages 81–86, 2008.

[22] S. Patil, J. Van Den Berg, S. Curtis, M. C. Lin, and D. Manocha. Directing crowd simulations using navigation fields. *Transactions on Visualization and Computer Graphics, IEEE*, 17(2):244–254, February 2011.

[23] J. Pettré, J.-P. Laumond, and D. Thalmann. A navigation graph for real-time crowd animation on multilayered and uneven terrain. *First International Workshop on Crowd Simulation*, pages 81–90, 2005.

[24] M. J. Quinn, R. A. Metoyer, and K. Hunter-Zaworski. Parallel implementation of the social forces model. In *Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics*, pages 63–74. Citeseer, 2003.

[25] C. Reynolds. Big fast crowds on ps3. In *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, pages 113–121. ACM, 2006.

[26] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

[27] P. Richmond. *FLAME GPU Technical Report and User Guide*. The University of Sheffield, Department of Computer Science, 1.0 edition, 2011.

[28] P. Richmond, S. Coakley, and D. Romano. Cellular level agent based modelling on the graphics processing unit. In *High Performance Computational Systems Biology, 2009. HIBI'09. International Workshop on*, pages 43–50. IEEE, 2009.

[29] P. Richmond and D. Romano. Agent based gpu, a real-time 3d simulation and interactive visualisation framework for massive agent based modelling on the gpu. In *Proceedings International Workshop on Supervisualisation*, 2008.

[30] P. Richmond and D. Romano. A high performance framework for agent based pedestrian dynamics on gpu hardware. *Proceedings of EUROSIS ESM*, 2008.

[31] P. Richmond and D. Romano. Template-driven agent-based modeling and simulation with cu da. *GPU Computing Gems Emerald Edition*, page 313, 2011.

[32] D. M. Romano, L. Lomax, and P. Richmond. Narcsim an agent-based illegal drug market simulation. In *Games Innovations Conference, 2009. ICE-GIC 2009. International IEEE Consumer Electronics Society's*, pages 101–108. IEEE, 2009.

[33] I. Sakellariou, O. Kurdi, M. Gheorghe, D. Romano, P. Kefalas, F. Ipate, and I. Niculescu. Crowd formal modelling and simulation: The sa'yee ritual. In *Computational Intelligence (UKCI), 2014 14th UK Workshop on*, pages 1–8. IEEE, 2014.

[34] M. Schwarz, D. Romano, and M. Gheorghe. Visualizing bacteria quorum sensing. In *AISB 2008 Convention Communication, Interaction and Social Intelligence*, volume 1, page 1, 2008.

[35] W. Shao and D. Terzopoulos. Autonomous pedestrians. *Graphical models*, 69(5):246–274, 2007.

[36] A. Sud, E. Andersen, S. Curtis, M. C. Lin, and D. Manocha. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):526–538, 2008.

[37] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 99–106. ACM, 2007.

[38] F. Tecchia, C. Loscos, R. Dalton, and Y. Chrysanthou. Agent behaviour simulator (abs): A platform for urban behaviour development. 2001.

[39] A. Treuille, S. Cooper, and Z. Popović. Continuum crowds. *ACM Transactions on Graphics (TOG)*, 25(3):1160–1168, 2006.

[40] A. Uhrmacher and K. Gugler. Distributed, parallel simulation of multiple, deliberative agents. In *Parallel and Distributed Simulation, 2000. PADS 2000. Proceedings. Fourteenth Workshop on*, pages 101–108. IEEE, 2000.

[41] Y. Wang and G. S. Chirikjian. A new potential field method for robot path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 977–982. IEEE, 2000.

[42] B. Zhou and S. Zhou. Parallel simulation of group behaviors. In *Proceedings of the 36th conference on Winter simulation*, pages 364–370. Winter Simulation Conference, 2004.