EG UK Theory and Practice of Computer Graphics (2010)
John Collomosse, Ian Grimstead (Editors)

Real-Time Traffic Simulation Using Cellular Automata

C. S. Applegate, S. D. Laycock and A. M. Day

Crowd Simulation Group, School of Computing Sciences, University of East Anglia, Norwich, United Kingdom. NR4 7TJ
c.applegate@uea.ac.uk, s.laycock@uea.ac.uk, a.day@uea.ac.uk

Abstract

In this paper, we present a method to simulate large-scale traffic networks, at real-time frame-rates. Our novel contributions include a method to automatically generate a road graph from real-life data, and our extension to a discrete traffic model, which we use to simulate traffic, demonstrating continuous vehicle motion between discrete locations. Given Ordnance Survey data, we automatically generate a road graph, identifying roads, junctions, and their connections. We distribute cells at regular intervals throughout the graph, which are used as discrete vehicle locations in our traffic model. Vehicle positions are then interpolated between cells to obtain continuous animation. We test the performance of our model using a 500 x 500m^2 area of a real city, and demonstrate that our model can simulate over 600 vehicles at real-time frame-rates ($>$ 80% network density).

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: traffic simulation, mesoscopic simulation, cellular automata, road generation, vehicle animation

1. Introduction

For more than 50 years, researchers from a diverse set of disciplines have been studying the behaviour of traffic flow to better understand the causes of traffic congestion, accidents and related phenomena. As the world's population continues to increase, there is high demand for effective and efficient transport infrastructures that alleviate these problems. Typically, cost-effective solutions are designed and developed using simulation packages, which allow transportation engineers to accurately model road networks and evaluate their designs though visual simulation. These packages provide three-dimensional representations of different traffic scenarios prior to their actual construction, allowing professionals, local authorities and the general public to instantly understand the expected impact on the surrounding environment. However, to better portray this information, methodologies to model larger-scale networks at higher visual fidelity are desirable.

One of the major challenges faced by researchers in the computer graphics industry, involves the design and implementation of algorithms that allow large-scale simulations to run at real-time frame-rates. This is a typical requirement of traffic simulators, as it enables designers to modify and interact with an environment in real-time, reducing the development time and overall cost of a project. However, to obtain the required performance, many applications will compromise behavioural accuracy or visual quality. These challenges also translate to the entertainment industry, as real-time traffic simulation is required in computer games, films and virtual tourism applications.

Over the years, there has been extensive research into the development of realistic traffic simulations, with the majority of literature either focusing on agent-based *microscopic models*, where individual vehicles have their own detailed behaviour, or continuum-based *macroscopic models*, which focus on the collective behaviour of vehicles, modelled through density conservation equations. A third type of traffic simulation, *mesoscopic models*, combines microscopic and macroscopic model properties, simulating individual vehicles that show collective behaviour. However, there has been relatively little literature that considers extending mesoscopic traffic models to produce traffic simulations, which exhibit both accurate behaviour and continuous animation.

DOI: 10.2312/LocalChapterEvents/TPCG/TPCG10/091-098

In this paper, we propose a system to simulate large-scale traffic networks in real-time. Our novel contributions include a method to automatically create a road graph from real-life data, and our extension to Nagel and Schreckenberg's original mesoscopic traffic model [NS92], which we use to simulate a large-scale traffic network, demonstrating continuous animation between discrete locations. We test the performance of our model using a 500 x 500m^2 area of a real city, and demonstrate that our model can simulate over 600 vehicles at real-time frame-rates ($>$ 80% network density).

2. Related Work

The realistic behaviour of vehicles has been studied extensively in literature by researchers from a variety of disciplines. The pioneering work of Lighthill and Whitham [LW55] and Richards [Ric56], investigated the behaviour of traffic flow as a density continuum, modelling the macroscopic or 'continuous' behaviour of traffic using theory from fluid-dynamics. The Lighthill-Whitham-Richards Model (LWR Model) uses aggregate variables representing the average velocity of vehicles, traffic density and traffic flow, to model the collective behaviour of vehicles from the defined relationship between these attributes. The main disadvantage of macroscopic traffic models is that they do not differentiate between individual vehicles, and it is therefore difficult to accurately simulate complex urban networks where there are a diverse range of vehicle types and driver behaviours. An advantage of modelling the collective behaviour of traffic is that simulations operate at high-speeds. For these reasons, macroscopic models are often used to simulate large-scale networks where detail is less important. Recently, Sewall et al. [SWML10] extended a macroscopic traffic model to produce detailed three-dimensional animations of traffic flow on large-scale networks. Their system uses continuum dynamics to model traffic, but maintains individual vehicle information to display each vehicle. Their method is scalable with multi-core/multi-processor architectures and runs at real-time frame-rates.

Microscopic or 'discrete' traffic simulations are the most popular form of traffic simulation, and model the behaviour of individual vehicles with detailed behaviours. Microscopic models are often preferred to macroscopic alternatives, as these systems can accurately simulate traffic on both urban and rural road networks, using 'car-following' and 'lane-changing' rules, which model individual vehicle attributes such as speed, acceleration, braking forces, reaction times, etc. In these rules, the movement of each vehicle is calculated based on the position or velocity of the vehicle in front. A disadvantage of microscopic models is that they typically require more computation time when compared with macroscopic models. Significant contributions in this area can be found in the works of Gerlough [Ger55], Newell [New61] and Gipps [Gip81]. Popular microscopic traffic simulators include, Quadstone Paramics and PTV AG's VISSIM.

Mesoscopic traffic simulators combine microscopic and macroscopic model properties, simulating individual vehicles that show aggregated behaviour at high computation speeds. In 1992, Nagel and Schreckenberg [NS92] proposed a mesoscopic model for traffic simulation using cellular automata. The authors used a one-dimensional array to represent a single-lane carriageway, which was segmented into a finite number of cells. Each cell contained a maximum of one vehicle, and vehicles moved between consecutive cells using simple rules. Updates were performed per-cell and in parallel, thus the performance of this model is scalable with multi-core/multi-processor architectures. Rickert et al. [RNSL96] extended Nagel and Schreckenberg's model to incorporate two lanes of traffic, where vehicles were permitted to change lanes, based on their distance to neighbouring vehicles. Nishinari and Takahashi [NT99] use Burgers Cellular Automaton (BCA), an ultra discrete version of the Burgers equation to model traffic flow. In their model, each cell can contain a number of vehicles, and movement between cells is defined by the evolution of the BCA. In Nishinari and Takahashi's model, individual lane-changing rules are neglected, but the macroscopic effects of lane changes are preserved. More recently, cellular approaches have been used to model the interactions in mixed bicycle flow [JJW04], the interactions between pedestrians and low-speed vehicles (e.g. bicycles, tricycles and barrows) [JW06b, JW06a] and between different types of vehicle [ZJG07, MDDZ07, CYZ*08, LGJZ09].

Although cellular approaches combine the advantages of macroscopic and microscopic models, their discrete nature is not directly suitable for animating the continuous movement of traffic, as vehicles appear to 'pop' between disjoint cells. Animation quality could be improved using finer discretisations, but at the cost of reducing performance. Alternative techniques for simulating the behavioural animation of vehicles have been investigated in the literature; Go et al. [GVK06], introduce a framework for animating autonomous agents in interactive applications, simulating the behaviour of vehicles and spacecraft, by combining Reynold's steering behaviours for autonomous agents [Rey99] with an online path-planning algorithm. The authors increase the run-time performance of their model through an adaptive sampling technique, which reduces the number of paths to search. More recently, van de Berg et al. [vdBSLM09] present a technique to reconstruct a three-dimensional visualisation of traffic from road-sensor data, in real-time. Their method records vehicle information at two discrete locations on a stretch of road, and computes vehicle trajectories between these two points. To produce realistic results, the algorithm minimalises the number of lane changes and vehicle acceleration and deceleration, whilst maximising the distance between individual vehicles.

These systems demonstrate high-quality vehicle animations. However, there is little literature which considers extending highly-scalable mesoscopic traffic models to pro-

duce similar results at high speeds. We identify that this is an area that requires further research, and therefore focus on mesoscopic models in this paper.

To model traffic, we require a road graph that details the layout of the road network. This could be created manually, but can become a time-consuming and laborious process when modelling large-scale networks. A variety of techniques for automatically generating roads have been studied in the literature. Chen et al. [CEW*08] proposed a method to interactively and intuitively generate large-scale road networks, based on tensor fields, which guide the automatic generation of roads. In their work, the authors present a variety of techniques to manipulate the tensor fields to obtain typical urban street networks. More recently, Galin et al. [GPMG10] proposed an automated technique for generating roads in complex environments that contain rivers, lakes, mountains and forests. A weighted shortest-path algorithm is performed between an initial and final location on the landscape, which is discretised into a regular grid. Weightings can be adjusted interactively to create roads which exhibit certain characteristics.

The above works have presented methodologies to generate realistic but fictitious roads from scratch, and also techniques to edit existing road networks. However, there has been little research with focus on the automatic identification of road structures, i.e. extracting roads, junctions, and their connections, from road centre line data. The ability to automatically generate road graphs from road centre lines is highly desirable, as it would enable networks to be intuitively designed with a few brush strokes in an editor, or created from Ordnance Survey data. We therefore focus on such techniques in this paper.

3. Generation of the Road Graph

To begin our process, we require a road graph that encodes the structure of the road network, detailing roads, junctions, and their connections. In this section, we present our novel technique for automatically generating a road graph from real-life data. It is important to note, that in our current implementation, all roads are considered as uni-directional roads. However, we plan to extend our technique to include bi-directional roads, in future work.

To generate the road graph, we use road centre lines obtained from Ordnance Survey LandLine.Plus data, represented as a set of vertices and connecting edges (Figure 1). We then identify and classify junctions, such that, vertices with three or more connecting edges are stored as *road junctions*, and vertices with only one connecting edge are stored as *terminal junctions*. These are junctions that provide either entry into or exit from the road network (Figure 2).

We consider a road to be a set of connecting edges that start and end at a road or terminal junction. However, we realise that there may be several junctions along a single road,

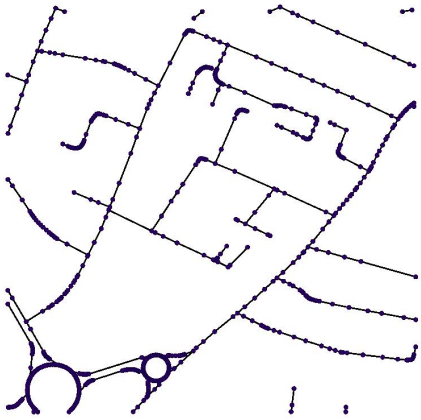

Figure 1: *Road centre lines of a 500 x 500m^2 tile area of a real city. The data is represented as a set of vertices and connecting edges. At the boundaries of the tile, there are some isolated edges, created from the tile segmentation.*

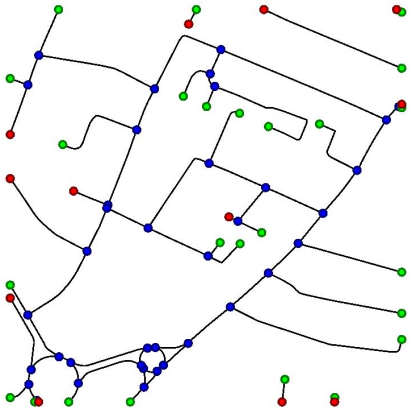

Figure 2: *Identified junctions on the road graph. Blue points indicate junctions with three or more connecting edges, green points indicate entry junctions, and red points indicate exit junctions.*

and therefore suggest that the end of a road could be identified by comparing the angles between connected edges at a junction. In our implementation, if the smallest of these angles is greater than 10°, we have reached the end of the road. This value was selected arbitrarily, as it provided the most accurate classification of roads in our test data (Figure 3).

Using this criteria, individual roads are identified. This is an iterative process, which visits each terminal junction and traverses through the string of unvisited edges, until the end of the road is identified. When it is possible to follow multiple edges, the edge with the smallest angle incident to

the road is selected. After identifying all roads that emanate from terminal junctions, we repeat the algorithm following edges that emanate from junctions with only one unvisited edge. This is then repeated until all edges have been visited.

Throughout the remainder of this paper, we refer to all junctions that lie between the start and end junctions of a road as *sub-junctions*, and the set of connected road edges between sub-junctions as *road links*. This forms a hierarchical structure, where a road edge connects two vertices, a road link is an ordered set of road edges, and a road is an ordered set of road links. We make a distinction between junctions and sub-junctions to encode road precedence, as vehicles joining a road from a sub-junction must yield to other traffic, e.g. traffic joining a main road from a t-junction.

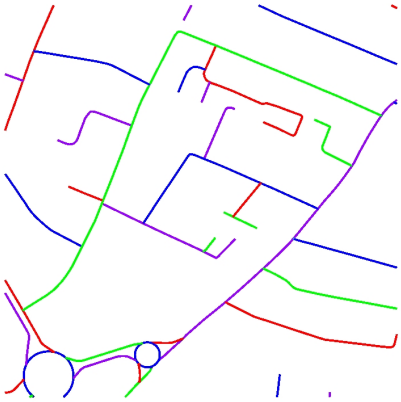

Figure 3: *Identified roads on the road graph. Coloured lines indicate individual roads.*

To simulate traffic, we apply Nagel and Schreckenberg's original traffic model [NS92] to our road graph. In this model, vehicles exist at discrete positions at discrete times. We define these positions as cells, which are created at regular intervals along each road. Ideally, the size of each cell would be constant throughout the road graph. However, as the road graph can be derived from real data, there is no guarantee that the length of a road will be a multiple of the desired cell length. To resolve this issue, we adopt an approach used by Sewall et al. [SWML10], where a target cell length Δx is defined. Cells are then positioned uniformly along each road link, rather than over an entire road, as this ensures that a cell is placed at every sub-junction. For a link i, of length L_i, the number of cells N_i and the cell length Δx_i, is defined in Equation 1. In our implementation, we have selected a target cell length, which is approximately the length of an average car, including safe distances. Figure 4 shows the distribution of cells over a section of the road graph.

$$N_i = \left\lfloor \frac{L_i}{\Delta x} \right\rfloor, \quad \Delta x_i = \frac{L_i}{N_i} \tag{1}$$

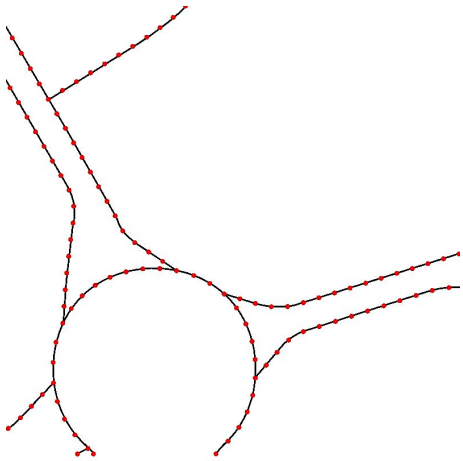

Figure 4: *Cells (red points) are positioned at regular intervals throughout the road graph.*

4. Behaviour Model

It is essential that the behaviour model enables the update of a large number of vehicles, with minimal computations, in order to obtain the desired performance. Therefore we implement Nagel and Schreckenberg's mesoscopic traffic model [NS92], which uses cellular automata to simulate individual vehicles with collective behaviour. Cellular automata are dynamic systems that operate on a regular lattice (commonly a grid of cells), where each lattice site (cell) has one of a finite number of states. States are defined by local interaction rules and are updated at discrete time-intervals, often in parallel. In Nagel and Schreckenberg's traffic model, each update consists of four consecutive steps, listed below:

1. Acceleration: a vehicle will increase its velocity v by 1, providing that it does not exceed its maximum velocity v_{max}, and the number of cells to the vehicle in front, is greater than $v+1$. The value of v_{max} is a user-defined constant integer.
2. Deceleration: if the distance to the vehicle in front, i, is less than or equal to v cells away, a vehicle will decrease its velocity to $i-1$.
3. Randomisation: a probability factor is used to randomly decrease vehicle velocities by 1, simulating natural alterations in velocity, due to human behaviour and external parameters. Without this step, the model would be deterministic and a pattern of motion would emerge. It is important to note that randomisation is only considered when the vehicle is in motion, i.e. $v > 0$.
4. Update: Each vehicle is advanced v cells along the road.

This model has been extended over the years, but the fundamental principles remain the same, i.e., a vehicle is permitted to 'hop' to an adjacent cell, provided that the cell is empty. However, conflicts arise when two vehicles attempt to move to the same unoccupied cell. For example, this could

occur at a t-junction, when vehicles on different roads attempt to move into the intersection cell at the junction. This issue could be resolved by performing all updates in series, but at the cost of reducing scalability and therefore performance.

Nagel et al. [NWWS98] encountered a similar problem when modelling lane-changing behaviour on highways with symmetric rules. In these systems, there is no default lane for traffic, and vehicles are permitted to overtake on either side of the highway. Conflicts occur when a vehicle in the left-most lane and a vehicle in the right-most lane, both attempt to change to the middle lane during the same update cycle. Nagel et al. propose scheduling the updates to resolve this issue, updating the behaviour of all vehicles turning from left-to-right, on *even* time steps, and updating all of the vehicles turning from right-to-left, on *odd* time steps.

We adopt a similar technique to resolve conflicts at road junctions, where vehicles attempting to join a road at a junction are forced to yield to other vehicles. This schedule prevents conflicts on our road graph, as no junction has more than three connected links, and therefore, a maximum of two vehicles can compete for the same cell. However, if there are junctions with four or more connected links (e.g. at crossroads), then further scheduling rules would be required. We suggest that priority could be either assigned randomly, or through scheduling rules that reflect the priorities of the specific junction.

5. Route Planning

In real environments, vehicles will enter and leave the road network. To simulate this, each vehicle that enters the graph, computes a route from its entry point to a random exit point, ensuring a continuous flow of vehicles through the network.

In our model, routes are calculated using Dijkstra's path-planning algorithm, where junctions on the road graph are represented as nodes in the path-planning algorithm. It is important to understand that the roads in our model are uni-directional, and therefore, there is no guarantee that a path exists between any two nodes. We incorporate this uni-directional structure into the path-planning algorithm, by only evaluating the cost of road links that emanate from the search node.

Our algorithm returns an ordered set of junctions to visit (Figure 5), which is then condensed by removing redundant information, leaving only the junctions where the vehicle must turn (Figure 6). This simplified list is analogous to the list of directions provided by a satellite navigation unit, as it details only the junctions that are important to the driver. We simplify the list of junctions to save memory, but more importantly, so that our model could incorporate variable vehicle speeds. This representation will enable us to determine the number of cells before the next turn, and therefore gradually reduce vehicle speed.

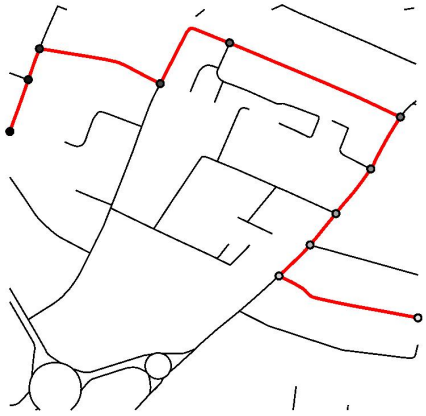

Figure 5: *An example route between an entry and an exit junction. The route contains an ordered list of junctions that the vehicle must visit.*

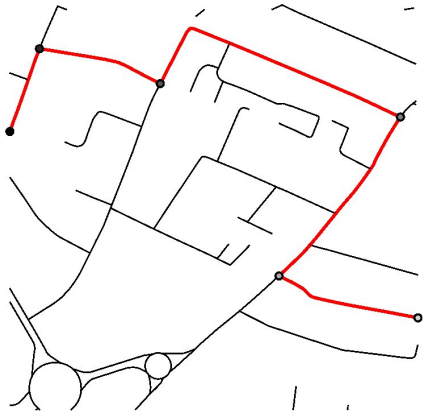

Figure 6: *The route plan is condensed by removing redundant information, leaving only the junctions, where the vehicle must turn.*

To maintain a constant vehicle density on the road graph, at each update cycle, we compare the total number of vehicles n_V, against an assigned network density $\rho' \in [0,1]$. The value of ρ' could be estimated using theory from space-syntax analysis [HH84], and maintained by modifying edge costs to favour under-populated routes. Road density $\rho \in [0,1]$, is defined as the total number of vehicles divided by the total number of cells n_C:

$$\rho = n_V / n_C \tag{2}$$

When ρ is less than the required density ρ', we compute

the number of vehicles that need to be inserted onto the network to maintain the required density, N_V:

$$N_V = \max(0, \lfloor \rho' * n_C \rfloor - n_V) \quad (3)$$

As there are a limited number of entry points on the road graph, the maximum number of vehicles that can be inserted onto the network, in one update cycle, I_V, will be the minimum of the number of vehicles that we wish to insert N_V, and the number of unoccupied entry points on the graph n_E:

$$I_V = \min(N_V, n_E) \quad (4)$$

To insert a vehicle, we randomly select an unoccupied entry junction and an exit junction on the road graph, and find the shortest route between these two points. This step is performed I_V times, to enter the required number of vehicles on the road graph to maintain ρ'.

6. Real-Time Simulation

The Nagel and Schreckenberg traffic model [NS92] can simulate large-scale networks with minimal computations, as it operates at discrete positions on the road graph, at discrete time-intervals, and it is highly parallelisable. Despite its discrete nature, the model is able to produce realistic behaviour, as it exploits time coherence, i.e., we assume the behaviour of a vehicle will remain constant in-between update intervals. This assumption holds, providing that the time-interval between consecutive updates is less than a driver's reaction time. One benefit of this model is that performance can be increased through the use of a more coarse discretisation, but at the cost of reducing behavioural realism. However, even at fine discretisations the simulation can appear visually disturbing, as vehicles appear to 'pop' from one cell to the next, in synchronisation. To resolve this issue, we propose interpolating the movement of vehicles between consecutive update steps, so that vehicles appear to move on a continuous path. This technique could also be used for simulations that do not require high behavioural realism, as vehicles will exhibit continuous motion, even if, coarse discretisations are used. In the remainder of this section, we present our novel extension to the Nagel and Schreckenberg traffic model, which we use to simulate large-scale traffic networks, with continuous vehicle movement between discrete cells.

To produce smooth animation, we use time-based animation techniques to compute the distance d that each vehicle has travelled since the last frame, assuming constant speed. A simple linear interpolation between the centres of the cells could be used to update the position of the vehicles. However, as the road graph can be generated from real-life data, we cannot guarantee that the road edges between cells will form a linear trajectory. For example, areas of high curvature will consist of multiple straight segments within a small area. If a simple linear interpolation was used, vehicles would not follow the curvature of the road, which would appear unrealistic, especially with coarse discretisations.

To resolve this issue, we compute the vector from the position of the vehicle, to the end point of the road edge that the vehicle is traversing. If the length of this vector is less than or equal to the distance the vehicle has travelled, d, then the vehicle is moved a distance d along the vector. However, if d is greater than the length of this vector, we traverse through subsequent road edges, recording the distance traversed t, until $t \geq d$. We then take the last traversed road edge and position the vehicle at a distance $t - d$ from its end.

We are able to increase the real-time performance of our simulation by pre-processing commonly executed queries. For example, we store a separate list of pointers to entry and exit junctions, the connected links at a junction, and component properties such as the length, unit vector and angle of rotation of road edges.

7. Results

We test our simulation on a 500 x 500m^2 area of a real city, with a coordinate resolution of 0.01m. The system has been tested with 50, 100, 200, 400 and 600 vehicles, on a computer with an Intel Core 2 Duo 3.06GHz processor, 2GB RAM and a NVIDIA GeForce 9800 GT graphics card, with 1GB of dedicated RAM. The simulation was written using the C++ programming language and utilises the OpenGL graphics library. Throughout our tests, all vehicles use the same geometric model, which contains approximately 1000 triangles, and the camera remains in a fixed position above the scene, with a bird's eye view of the road network. All simulations were performed over 420 time-steps (7 minutes), and the first 120 time-steps have been discarded to allow for the initial transient stage, when vehicles are entering the network. The road graph consists of 38 roads and 74 junctions, requiring 66.37ms to compute. A video demonstrating our simulation can be found at: http://www.crowdsimulationgroup.co.uk/TrafficSimulation.wmv.

We analyse the real-time performance of the system by measuring the frame-rates of the simulation, using three different behaviour models A, B and C. The first behaviour, A, tests the rendering performance of the scene, displaying all vehicles without any behaviour model. Behaviour B is representative of the original Nagel and Schreckenberg traffic model [NS92] using our junction rules, and Behaviour C is our extension of the Nagel and Schreckenberg model, which interpolates vehicle positions between cells. The results are presented in Table 1.

The results demonstrate that our extension to the Nagel and Schreckenberg traffic model (Behaviour C) is suitable for real-time applications, as we are able to simulate 600

No. of Vehicles	ρ'	Behaviour A			Behaviour B			Behaviour C		
		Min	Avr	Max	Min	Avr	Max	Min	Avr	Max
50	0.07	939.00	948.08	949.00	887.00	957.52	1012.00	390.00	441.36	509.00
100	0.14	523.00	526.41	527.00	484.00	534.10	548.00	190.00	256.03	319.00
200	0.28	273.00	278.45	279.00	278.00	281.30	285.00	197.00	255.11	284.00
400	0.55	142.00	143.40	144.00	138.00	144.32	147.00	40.00	115.22	146.00
600	0.83	96.00	96.57	97.00	109.00	119.64	147.00	45.00	101.72	123.00

Table 1: *The frames-per-second (FPS) of our simulation with no behaviour (A), the Nagel and Schreckenberg behaviour model using our junction rules (B), and our extension to the Nagel and Schreckenberg model (C).*

vehicles, at frames-rates above the requirements for real-time performance. Furthermore, the performance difference between the original model (Behaviour B) and our extension (Behaviour C) decreases, as the number of vehicles increases. We suspect that the performance of these two behaviours are similar when simulating large numbers of vehicles, as network density is higher and traffic jams form more easily. With the majority of the vehicles remaining stationary, little interpolation is required and therefore the difference in performance is negligible.

It is also evident, that the performance of the original model (Behaviour B) is limited by the rendering process. However, in some tests, Behaviour B outperforms Behaviour A. We believe this is because all vehicles are rendered throughout the entire simulation during Behaviour A, but this does not always occur during Behaviour B, as vehicles continually enter and exit the road network. Therefore, the actual network density ρ is often less than the desired network density ρ', and fewer vehicles are rendered, increasing performance. This effect is magnified at higher values of ρ' and when the ratio of entry to exit junctions is low.

To better evaluate the performance of behaviours B and C, we decouple the rendering process from the behaviour system and provide the computation times required to update the original Nagel and Schreckenberg traffic model (Behaviour B), and the times required to interpolate all vehicles in our extension to the Nagel and Schreckenberg model (Behaviour C). The results are presented in Table 2 and show that the average computation time required by the interpolation (Behaviour C) is less than 5ms and remains constant as the number of vehicles increases. As Behaviour B is updated at discrete time-intervals (1Hz), and the interpolation is performed per-frame between intervals, the additional computation required to exhibit continuous vehicle movement is bound by the larger of the two computation times.

8. Conclusions and Future Work

In this paper, we have presented a novel technique to automatically extract road information from Ordnance Survey data and create a road graph, suitable for simulating traffic at real-time frame-rates. In addition, we have extended Nagel and Schreckenberg's original behaviour model [NS92], by interpolating vehicle movement between discrete cell positions, enabling us to simulate traffic across a large area of a real city, with continuous movement at coarse cell discretisations. Figure 7 shows a screenshot of our real-time simulation.

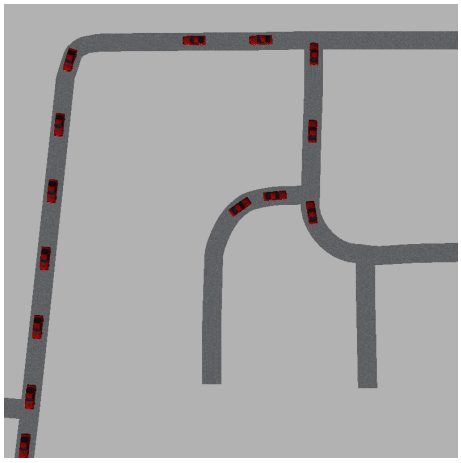

Figure 7: *A screenshot of our traffic simulation.*

In future work, we aim to improve the performance of our model, so that this technique can be used to simulate a diverse range of vehicles on a much larger scale. To achieve this, we intend to exploit the mass-scalability of our model, by performing updates in parallel and utilising acceleration techniques on the GPU. Furthermore, we aim to investigate different levels of simulation, where vehicles further from the view frustum have an increasing coarse interpolation. This would enable us to increase system performance, without compromising visual quality. We also intend to incorporate bi-directional roads and variable vehicle speeds.

References

[CEW*08] CHEN G., ESCH G., WONKA P., MÜLLER P., ZHANG E.: Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3 (2008), 1–10.

[CYZ*08] CHENG S., YAO D., ZHANG Y., SU Y., XU

No. of Vehicles	ρ'	Behaviour B			Behaviour C		
		Min	Avr	Max	Min	Avr	Max
50	0.07	0.31	0.85	5.16	1.62	2.27	5.07
100	0.14	0.32	0.98	4.25	2.25	3.02	4.55
200	0.28	0.33	1.55	13.05	2.33	3.19	7.23
400	0.55	0.36	5.19	36.69	1.20	1.86	5.42
600	0.83	1.12	8.69	49.17	1.54	2.63	13.76

Table 2: *Computation times required to update the original Nagel and Schreckenberg traffic model (B), and the times required to interpolate all vehicles in our extension to the Nagel and Schreckenberg model (C). Times are shown in milliseconds.*

W.: A CA model for intrusion conflicts simulation in vehicles-bicycles laminar traffic flow. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on* (2008), pp. 633–638.

[Ger55] GERLOUGH D.: *Simulation of freeway traffic on a general-purpose discrete variable computer*. PhD thesis, University of California, Los Angeles, 1955.

[Gip81] GIPPS P.: A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological 15*, 2 (1981), 105–111.

[GPMG10] GALIN E., PEYTAVIE A., MARÉCHAL N., GUÉRIN E.: Procedural Generation of Roads. *Computer Graphics Forum (Proceedings of Eurographics) 29*, 2 (2010).

[GVK06] GO J., VU T., KUFFNER J.: Autonomous behaviors for interactive vehicle animations. *Graphical models 68*, 2 (2006), 90–112.

[HH84] HILLIER B., HANSON J.: *The social logic of space*. Cambridge University Press Cambridge, 1984.

[JJW04] JIANG R., JIA B., WU Q.: Stochastic multi-value cellular automata models for bicycle flow. *Journal of Physics A: Mathematical and General 37* (2004), 2063–2072.

[JW06a] JIANG R., WU Q.: Interaction between vehicle and pedestrians in a narrow channel. *Physica A: Statistical Mechanics and its Applications 368*, 1 (2006), 239–246.

[JW06b] JIANG R., WU Q.: The moving behavior of a large object in the crowds in a narrow channel. *Physica A: Statistical Mechanics and its Applications 364* (2006), 457–463.

[LGJZ09] LI X., GAO Z., JIA B., ZHAO X.: Modeling the Interaction Between Motorized Vehicle and Bicycle by Using Cellular Automata Model. *International Journal of Modern Physics C 20* (2009), 209–222.

[LW55] LIGHTHILL M., WHITHAM G.: On kinematic waves. II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences* (1955), 317–345.

[MDDZ07] MENG J., DAI S., DONG L., ZHANG J.: Cellular automaton model for mixed traffic flow with motorcycles. *Physica A: Statistical Mechanics and its Applications 380* (2007), 470–480.

[New61] NEWELL G.: Nonlinear effects in the dynamics of car following. *Operations Research 9*, 2 (1961), 209–229.

[NS92] NAGEL K., SCHRECKENBERG M.: A cellular automaton model for freeway traffic. *J. Phys. I France 2* (1992), 2221–2229.

[NT99] NISHINARI K., TAKAHASHI D.: A new deterministic CA model for traffic flow with multiple states. *Journal of Physics A: Mathematical and General 32* (1999), 93–104.

[NWWS98] NAGEL K., WOLF D., WAGNER P., SIMON P.: Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E 58*, 2 (1998), 1425–1437.

[Rey99] REYNOLDS C.: Steering behaviors for autonomous characters. In *Game Developers Conference. http://www.red3d.com/cwr/steer/gdc99* (1999), Citeseer.

[Ric56] RICHARDS P.: Shock waves on the highway. *Operations research 4*, 1 (1956), 42–51.

[RNSL96] RICKERT M., NAGEL K., SCHRECKENBERG M., LATOUR A.: Two lane traffic simulations using cellular automata. *Physica A: Statistical and Theoretical Physics 231*, 4 (1996), 534–550.

[SWML10] SEWALL J., WILKIE D., MERRELL P., LIN M.: Continuum traffic simulation. In *Computer Graphics Forum* (2010), vol. 29, John Wiley & Sons, pp. 439–448.

[vdBSLM09] VAN DEN BERG J., SEWALL J., LIN M., MANOCHA D.: Virtualized Traffic: Reconstructing Traffic Flows from Discrete Spatio-Temporal Data. In *Proceedings of the 2009 IEEE Virtual Reality Conference* (2009), Citeseer, pp. 183–190.

[ZJG07] ZHAO X., JIA B., GAO Z.: A new approach for modelling mixed traffic flow with motorized vehicles and non-motorized vehicles based on cellular automaton model. *Arxiv preprint arXiv:0707.1169* (2007).