GPU-based Real-Time Execution of Vehicular Mobility Models in Large-Scale Road Network Scenarios

Kalyan S. Perumalla, Brandon G. Aaby, Srikanth B. Yoginath, Sudip K. Seal
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
perumallaks@ornl.gov, aabybg@ornl.gov, yoginathsb@ornl.gov, sealsk@ornl.gov

Abstract

A methodology and its associated algorithms are presented for mapping a novel, field-based vehicular mobility model onto graphical processing unit computational platform for simulating mobility in large-scale road networks. Of particular focus is the achievement of real-time execution, on desktop platforms, of vehicular mobility on road networks comprised of millions of nodes and links, and multi-million counts of simultaneously active vehicles. The methodology is realized in a system called GARFIELD, whose implementation details and performance study are described. The runtime characteristics of a prototype implementation are presented that show real-time performance in simulations of networks at the scale of a few states of the US road networks.

1. Introduction

1.1. Motivation

Simulations are routinely used in emergency planning and management in order to make decisions such as whether to order an evacuation or not [1, 2]. The quality of decisions can greatly depend on the quality of insights into simulation results. When larger geographical regions are considered in such decisions, simulations become highly computationally intensive. Improving the speed of large-scale simulation can help evaluate an increased number of alternatives, and improve confidence bounds, all within the short amount of decision time available.

Large-scale scenarios of vehicular traffic simulation problems are characterized by long-range queuing effects, control mechanisms and other phenomena. While small-sized scenarios are relatively easy to analyze, larger scenarios need specialized treatment for efficient execution, especially for very large network sizes (millions of road intersections) and/or for heavy loads of vehicular traffic load. An appealing computational platform in this context is a graphical processing unit (GPU).

1.2. Background

Graphical processing units have been subjected to general-purpose use over the past decade. Literature on general-purpose computation over GPUs is extensive [3, 4]. However, newer methodologies and implementation approaches are still being discovered to exploit GPUs in different areas. Although GPU-based execution is not new, and the computational potential of GPUs has been known, no specific method has been proposed to map vehicular mobility models to GPUs. Traditional CPU-based (time-stepped or event-driven) models have remained elusive for straightforward application to the GPU domain.

1.3. Contributions

To the best of our knowledge, ours is the first work to apply GPU-based model execution to transportation network simulations. Also, we are not aware of any other system or approach that has been shown to support queuing effects in either aggregate or semi-aggregate models of vehicular mobility at the level of millions of road network nodes and links. Ours is also the first to provide a novel field-based formulation of a vehicular traffic mobility model in a large road network that can be executed on a GPU. Modeling dynamic re-routing is another distinguishing aspect of our field-based model that has never been attempted before at large-scale in other models and simulators.

1.4. Related Work

Commonly-used execution approaches span a continuous spectrum, between fully disaggregated, agent-based models, and fully abstracted, network flow analysis formulations. Examples of flow analysis-based methods include macro-simulators CORSIM [5, 6] and OREMS [1]. Examples of disaggregated approaches include micro-simulators such as TRANSIMS [7], VISSIM [8], and SCATTER[9], among many others. The literature on macro-simulators and micro-simulators in the mobility domain is extensive. The reader is referred to [1, 2, 5-

15] as starting points. However, field-based modeling is relatively new; we are not aware of any existing or published documentation on a field-based vehicular queuing simulator. Here, we demonstrate the applicability of field-based mobility for efficient execution on GPUs.

Another distinguishing aspect of our field-based formulation, compared to field-based formulations of other domains (such as electro-magnetic fields), is the ability to include additional non-linear behavioral details, including queuing and randomized, directed flows. The additional complexity in execution that entails such a generalized field-based model is handled using a GPU-based execution approach. The published methods closest to our present subject are [16, 17] in which crowd behavior is simulated on a GPU. However, they do not deal with mobility in constrained paths; in our road network scenarios, mobility along specific paths (streets and highways) needs to be enforced, along with queuing, which makes our model more complex than unconstrained motion on a plane.

1.5. Our Approach

To help exploit the power of the GPUs, we define a novel field-based formulation of vehicular mobility. In a field-based formulation, the road network control is viewed as a spatially distributed field (analogous to physical fields such as magnetic field) in which vehicles are immersed. These vehicles are influenced by the field and undergo corresponding movement. The field is defined in terms of vectors (directionality and intensity) of movement at each network node. A canonical regular vector field grid scheme is defined, to which any arbitrary road network can be mapped.

On a GPU, the field-based mobility model is executed with probabilistic transfers of vehicular counts between adjacent road network segments. State variables of the road network, along with detailed information on vehicular traffic loading, are carefully encoded to minimize memory requirements for representation and manipulation during simulation. Thereafter, execution is mapped using traditional GPU-based techniques for data-parallel execution.

An early version of our prototype implementation is currently operational. Preliminary results show scalability to a field of over 2 million network nodes, and 20 million represented vehicles. Mobility can be described in a generalized field-based model view. In evacuation simulations, for example, arbitrary fields can be defined to represent any evacuation control scheme. Execution of our prototype implementation shows that results from our system are achieved in real-time, which is significantly faster than any existing vehicular mobility simulator. Simultaneously, the capability with respect to network size is significantly increased, from tens of thousands of nodes of extant systems to millions of nodes in our new system.

1.6. Organization

The rest of the document is organized as follows. Our mobility model is described in Section 2. The method for mapping an arbitrary vehicular road network to our canonical, field-based grid is described in Section 3. The framework for scenario specification and configuration is presented in Section 4. Details of our system implementation and computing platform are given in Section 5. Following that, an experimental study of scalability and performance is described in Section 6. Final remarks and a discussion of future work are provided in Section 7.

2. Mobility Model

The mobility model consists of the following components: (1) a global routing model for choice of turns or hops in a trip towards destinations (2) a mobility model for representation of vehicles moving along a link, and (3) a queuing model for stalling, congestion and dynamic re-routing of vehicles along congested paths. These three components are described next.

2.1. Field-based Mobility and Routing

In our field-based view of mobility, the directional vectors for movement are defined per cell on a specially-defined spatial grid of cells (the grid is described in greater detail in Section 3.1). A vehicle placed at a cell is directed in its movement along the direction vector of the cell, independent of the vehicle's historical path. When properly defined, fields can be readily used to formulate evacuation plans. With the addition of a dynamic-update capability to the field specifications, field-based models can be used to create other non-emergency activities as well. The field is somewhat analogous to physical fields such as electric and magnetic fields.

In our field-based formulation, each cell is categorized into either a "vertical cell" or a "horizontal cell." Vertical cells are those in which vehicles are constrained to move only in vertical direction in our specially-defined two-dimensional grid. Horizontal cells are those in which vehicles move only in horizontal direction. For a specific cell and a direction of movement, a vector of probabilities is defined as V_c=[v_L, v_R, v_S, v_U], where v_L represents the probability that the vehicle turns left, v_R for turning right, v_S for proceeding straight and v_U represents a probability of taking a U-turn, and the sum of the four probabilities equals unity. Each vertical cell is assigned two such vectors V_c^T and V_c^B, for vehicles exiting from the top face and exiting from bottom face respectively. Similarly, each horizontal cell is assigned two vectors V_c^L and V_c^R, for vehicles exiting from the left face and from the right face respectively. To summarize, the entire domain is represented as vertical and horizontal cells, with each cell possessing two probability vectors, each vector containing four probability values as vector components. The probability vectors per cell completely define the field and determine the effects of the field on mobility. Note that any of the probability elements can be zero, which can be used to represent the absence of connectivity.

2.2. Semi-Aggregated Movement

Vehicles are represented as an aggregate count at each direction in each cell. Although the counts are integral at initialization, they are allowed to take on fractional values, as needed in the mobility model, during the course of the simulation. Each cell maintains two counts, one for each direction: left and right for horizontal cells, and up and down for vertical cells. Thus, two floating point numbers represent the number of vehicles occupying the cell in each direction at any given simulation moment. Simulation follows a time-stepped mode of execution, each time step being split into two phases. In the first phase, called the "split phase," each cell determines the fraction of its current vehicle count that will move in each of the four neighbor destinations for each direction. In other words, for a horizontal cell, the number of outgoing vehicles is computed for each of its two (left and right) directions, each of which computes outflow to four neighbors: reachable via left-, right-, straight- and u-turns. An upper bound on the total number of outgoing vehicles is determined based on the speed limit and link length.

2.3. Queuing, Congestion and Rerouting

The queuing phenomenon is modeled in terms of accumulation of vehicle counts at each cell. In the second phase ("merge phase") of each time step, each cell gathers the counts of all incoming flows and adds them to its current occupancy. When a cell becomes full (i.e., its vehicle count reaches its capacity), its neighbors detect the lack of capacity in the first (split) phase and refrain from sending any vehicles to that neighbor. Under congested operation, this results in a chain of blocked traffic, with cells reaching their capacity along a path. Note that the probabilistic turns also automatically induce dynamic re-routing. The re-routing is automatically achieved by the fact that more vehicles will be available to get diverted to other neighbors when one neighbor becomes full.

3. Mapping the Network Graph to a Texel Grid

An important objective in our efforts is in determining how we could exploit the great computing potential of graphical processing units or of parallel computing platforms in general. While the computational platforms are generally optimized for "rectangular" data structures, road networks, on the other hand, are graphs. A method is needed to reconcile the disparity. Our solution approach to this apparent mismatch is to map the road network graph to a rectangular data structure. By and large, GPUs are very highly optimized to process rectangular data in the form of image textures; we exploit this fact in our implementation. This approach can also be suitable for other parallel platforms as well.

The mapping is performed as follows. A canonical form of a rectangular grid is defined. This canonical grid will be capable of capturing the structural elements and also provide holders for key behavioral parameters at a cell-level resolution. An encoding scheme is then defined to represent the grid in a form that reduces the memory usage to minimal levels. Finally, an algorithm is used to map any input graph onto a corresponding canonical grid. This algorithm generates the field, and the initial loading pattern. Each of these steps is described in the following sub-sections.

3.1. Canonical Grid Network

As mentioned previously, the geographical region is decomposed and encoded as a discretization into cells. Each cell represents both "directions" for that point: vertical cells have up and down, and horizontal has left and right. In our canonical encoding, vertical cells are represented in even-numbered rows and horizontal cells are represented in odd-numbered rows. Figure 1 illustrates the incoming connectivity for horizontal cells. The outgoing connectivity for the horizontal cells, and the incoming/outgoing connectivity for vertical cells are analogously organized. Given a cell (i,j), the neighbor offsets are $(0, \pm1 \text{ or } \pm2)$ for the neighbors relative to (i,j). In the figure, the cell (i,j) in question is colored in pink, and the arcs are labeled L for left turn, R for right turn, S for straight traversal and U for a U-turn. Horizontal rows are labeled H and vertical rows are labeled V.

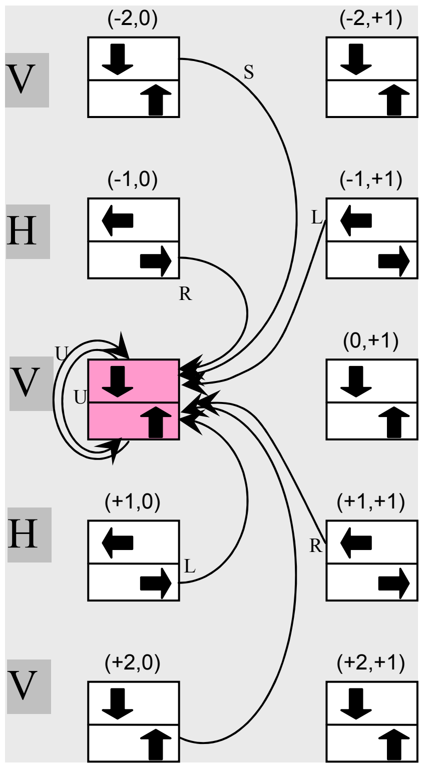

Figure 1: Incoming dependencies for vertical links

3.2. Representation and Allocation of State

The discretization of vehicular mobility model is realized using multiple instances of a two dimensional array template. These array instances are mapped to a common GPU data structure called a *texture* whose elements are called *texels*. We utilize textures of a "32-bit RGBA" (32-bit Red Green Blue Alpha) format in which each texel is used to represent various static and dynamic attributes in our simulation. In these textures, each texel comprises four floating point values, each floating point value represented by 4 bytes. Additionally, information about global settings such as the grid size and the time step of simulation are passed as individual parameters to the GPU processing routines. The following encoding methodology is used, whose underlying objective is to minimize the memory usage for representing the simulation state.

In our current version of functionality, the simulation state (constants and variables) is represented in the model by six textures, as described next.

Model constants are contained in two textures representing all turn probabilities for a given cell. The RGBA elements of the first of texture store the left and right turn probabilities that are computed for each grid point. The second texture holds straight and U-turn probabilities. All the corresponding left-, right-, straight- and U-turn probabilities add up to unity. These two textures are read-only, and hence remain unmodified during simulation. The rest of the textures are used in a read-write fashion, and they represent aggregate vehicle counts for both split and merge phases, as will be described in greater detail later. Two random number seeds are also stored within the state texture. These will be used and updated at every iteration step. An additional encoding of these random values is used to distinguish between destination cells and normal cells. A negative seed value indicates that the corresponding cell is a destination/sink cell (e.g., an evacuation point), and a positive seed value indicates a normal cell.

A final constant value governing mobility is the segment (road) length. Segment length affects mobility in that it partially determines the maximum number of vehicles leaving a given intersection. To calculate the maximum number of outgoing vehicles at any cell (subsequently subjected to random distribution and field of evacuation constraints) we observe the following relation:

$$Max_{ts} = Max\ outflow\ per\ traffic\ signal = speed\ limit \times time\ step \div vehicle\ length$$

$$Max = Max_{ts} \times current\ vehicle\ count \times vehicle\ length \div segment\ length$$

The quantum of outgoing vehicles determined by the preceding relations helps us vary vehicle evacuation speed according to model size, allowing both flexibility and fidelity of our model.

3.3. Memory Size and Precision

The dominant factor in quantifying memory usage is texture size. For our encoding, textures of 16 bytes per texel are used, with six textures in total (two constant textures, and one intermediate state texture, and two for use in the ping-pong scheme for updating the state). This gives $N \times N \times 96$ bytes of required space where N is canonical texel grid size (the canonical grid is described in the next section). Therefore, for a grid size of 2048, we consume roughly 384 MB for these textures. We have achieved simulation sizes of $N > 3750$ (over 14 million texel grid elements). Larger grid sizes can be supported with the next generation graphics processors, such as the NVIDIA 9000 series.

3.4. Mapping Input Graph to Texel Grid

The input road network graph consists of nodes that represent intersections, and edges that represent the road segment links between intersections. The input also specifies two-dimensional Cartesian coordinates (or latitude and longitude) for every node. To map the input road network graph to our canonical texel grid, the Cartesian coordinates of the nodes in the network graph are first spatially translated or shifted such that they all lie in the positive *xy*-quadrant. The smallest rectangle (in units of a pre-specified cell-level resolution) that encloses the resulting nodes is then computed from the Cartesian coordinates of the nodes in the network. This enclosing rectangle is decomposed into a two-dimensional array of cells using the available cell-level resolution. Each cell has a one-to-one mapping with a texel. Each texel is then marked as colored or uncolored. If a texel has at least one intersection node mapped to it, then, it is marked as occupied. Additionally, if at least one link passes through that texel, the texel is marked as occupied. Algorithmically, cells that are encountered by a road link edge $E(u,v)$ when traversing from node u to node v are also marked occupied.

With respect to the canonical texel grid, there are three possible orientations for any given edge, namely, vertical, horizontal and otherwise. For horizontal (or vertical) edges, all the encountered cells can be trivially determined. This is achieved by simply marking every cell that is encountered along the horizontal (or vertical) direction between the two nodes u and v. An edge $E(u,v)$, that is neither vertical nor horizontal, is viewed as a straight line on a Cartesian plane that passes through points (u_1,u_2) and (v_1,v_2) where u_1 , u_2 are the Cartesian coordinates of node u and v_1 , v_2 are those of node v, respectively. Note that, to determine which cells are encountered along any edge $E(u,v)$, it is sufficient to compute the points of intersection of that edge with the horizontal and vertical grid lines of the cell array. Given an edge $E(u,v)$, the identities of the horizontal and vertical grid lines that are contained within the smallest rectangle that encloses the two cells occupied by u and v, respectively, can be easily determined in constant time from the knowledge of the integer coordinates of the cells that contain u and v. The remaining task is therefore to determine the points of intersection of the vertical and horizontal grid lines contained within this rectangle with the edge $E(u,v)$. It can be shown that the following constant time algorithm returns the point of intersection between a horizontal grid line L_1 that pass through Cartesian points (x_1,y_1) and (x_2,y_2) and an edge $E(u,v)$ whose end points have Cartesian coordinates (u_1,u_2) and (v_1,v_2):

```
Algorithm: Intersection ($L_1(x_1,y_1:x_2,y_2), E(u_1,u_2:v_1,v_2)$)
    1. $b_1 = (y_2-y_1)/(x_2-x_1)$; $b_2 = (v_2-v_1)/(u_2-u_1)$;
    2. $a_1 = y_1-b_1*x_1$; $a_2 = v_1-b_2*u_1$;
    3. return $I_x = -(a_1-a_2)/(b_1-b_2)$, $I_y = a_1+b_1*I_x$;
```

Note that for a vertical grid line whose slope is infinity, the above algorithm with a coordinate rotation accomplishes the same task.

Each point of intersection returned by the preceding algorithm marks two neighboring cells as occupied, one on either side of the grid line with which the edge intersects. For example, if the Cartesian point of intersection of an edge with a vertical grid line is (I_x,I_y), then the integer coordinates (X,Y) of the cells in the texel array which are marked as occupied are: ($X = floor(I_x/s)$, $Y = Y_{max} - floor(I_y/s)$) and ($X = floor(I_x/s)-1$, $Y = Y_{max} - floor(I_y/s)$), where s is the side-length of each cell and Y_{max} is the total number of cells in the vertical direction. Similarly, if an edge intersects a horizontal grid line at the point (I_x,I_y), the integer coordinates (X,Y) of the cells in the texel array which are marked as occupied are: ($X = floor(I_x/s)$, $Y = Y_{max} - floor(I_y/s)$) and ($X = floor(I_x/s)$, $Y = Y_{max} - floor(I_y/s)-1$). Thus, each point of intersection (I_x,I_y) immediately yields the identities of the cells through which the edge $E(u,v)$ passes. The corresponding cells are then marked occupied.

Repeating this for each edge in the network graph marks those grid cells in the array that either contain a node (road intersection) or has an edge (road) passing through it. In this manner, the above algorithm maps the original road network onto the texel grid, which can now be viewed simply as an array of occupied or unoccupied texels. If $|E_{max}|$ denotes the length of the longest edge in the input graph, then the runtime for the above mapping algorithm is bounded by $O(|V|+|E_{max}|/s)$. Storage considerations are considered in a later section.

4. Scenario Configuration

In order to use a road network specified by the user as input to the simulator, the network is first preprocessed to a form suitable for field-based model execution. An induced graph is generated as part of the preprocessing, with related grid resolution effects and computation of vehicular turn probabilities, as described next.

4.1. Induced Graph and Its Generation

The neighborhood of occupied cells in the texel grid determines an aggregated, rectangular version of the input road network graph. The resultant graph of texel cell neighborhood is called the induced graph. The induced graph approaches in equivalence the original input graph as the texel grid size is increased, and in a limit (in the worst case, not less than one vehicle length) reaches fully faithful and accurate representation (e.g., at 5m×5m resolution).

4.2. Grid Resolution Effects

When the input graph is mapped to the canonical grid, translation accuracy in the mapping scheme needs to be considered. Segments that are perfectly horizontal or vertical are induced in a way such that only vertical and horizontal cells are marked filled. For diagonal segments, increase in cell resolution makes our mapping eventually reach consistency with the road network. The resolutions afforded by our texel grid mapping are shown in Table 1. It is clear from the table that for smaller networks (such as Washington, D.C.), the cell resolution is extremely high, approaching that of an individual vehicle. As the geographical area increases, the resolution decreases due to the fixed texture size; yet, the cell resolution is at the level of one or two city block sizes in width, which is sufficiently detailed for state-level simulations.

Table 1: Grid and cell resolutions with a 4K×4K texture

Region (state)	Area (km^2)	Cell (m^2)
Washington (DC)	16×16	4×4
Louisiana (LA)	610×210	148×148
Tennessee (TN)	710×195	173×173
Florida (FL)	582×721	170×170
Texas (TX)	1,244×1270	310×310

4.3. Computing Turn Probabilities

At initialization, probabilities are carefully assigned to correspond to the road network. Vehicles off the road network will be guided towards the closest road, i.e., probabilities of turning in any direction other than towards the closest road will be zero. For all the texels at which there is no road, the turn probabilities can be assigned arbitrary values. Alternatively, they can be assigned to make the vehicles flow towards the closest road. For the texels on which at least one road appears, probabilities are assigned based on a field of vectors for a given evacuation strategy defined by the count of sinks and their specific geographical placement.

Every unoccupied texel is assigned a zero probability vector. For each occupied texel, its probability vector is assigned by querying the occupancy of the six cells to the top and bottom if it is a vertical cell or the six cells to the left and right if it is a horizontal cell. The grid of probability vectors resulting from the above mapping of the input road network to the texel grid completely defines the mobility field.

The input specifies a set of destination points or cells. The vehicles at any cell in the canonical grid move towards one of its closest destination cells. Since, the destination cells are known, the turn probabilities for each cell can be generated such that the all traffic at that cell is routed to its closest destination cell(s).

If multiple destination cells are equidistant from a particular cell, the turn probabilities of that cell are equally divided among the links that route traffic to those multiple destinations. If there is only a single closest destination cell then the link that routes traffic to that destination will be assigned a turn-probability of unity, and the turn-probabilities on all the other links will be assigned zero probability.

5. Implementation and Benchmarks

We now describe the implementation of our prototype system called GARFIELD (Graphical Agents Reacting in a Field), and the set of benchmarks we use to verify the correctness and evaluate the performance of our simulator.

5.1. GPU-based Implementation

Our system has been developed using the NVIDIA Cg Toolkit, OpenGL, and Microsoft Visual Studio .NET. The hardware for GPU-based experiments is a recent NVIDIA GeForce 8800 GTX unit with 768MB of onboard memory, 128 stream processors, and a core clock speed of 575MHz. The CPU is an Intel Core2 Duo 2.4 GHz processor with 4 GB memory.

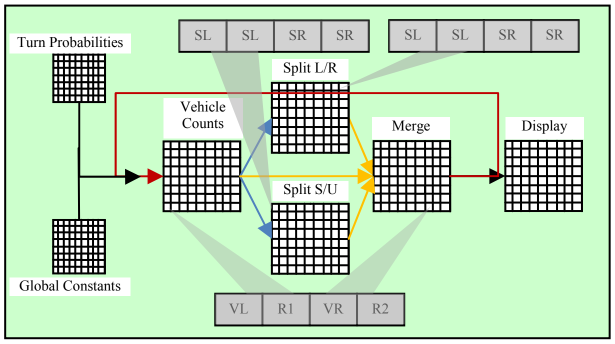

Figure 2: Multi-phase computation in GARFIELD. V*D* is aggregate vehicle count for a road direction *D*, R1 and R2 are random number seeds, and S*D* is the number of outgoing vehicles in direction *D*.

In the GPU-based execution, evolution by one time step in the model equals one iteration on the GPU. Each GPU state update is performed in distinct phases, within split and merge operations (Figure 2). In the split phase, the number of vehicles to send to neighboring cells is computed, accounting for traffic conditions such as speed limits, signal timing delays, and vehicular congestion. A later phase is the merging of neighboring split vehicles.

5.2. Visualization and Customized Displays

GARFIELD provides visualization to render the simulation state periodically to the screen. Since all simulation state resides on the graphics card and is updated only by graphics kernel invocations, we optimize the display by rendering directly from the graphics memory. The display, however, is fully customizable, by applying a fragment kernel that uses the simulation state as read-only state. We apply a display kernel at the end of each iteration step to properly shade each texel according to the desired view. The vehicular mobility animation can be overlaid on top of a traditional geographical map, or only the street occupancy can be displayed over a contrasting background. Congestion can be highlighted in distinct color (we render congested cells in red). Using the OpenGL Utility Toolkit (GLUT), we also allow for user interaction such as frame rate control and the ability to modify both display color and the presence of a background geographical map. Additionally, users can dynamically pan and zoom in/out the visualization at runtime.

5.3. Benchmarks

We exercise our system using two different types of benchmarks. The first type is a set of synthetic road network whose size and parameters can be varied easily for experimentation. The second type is a set of actual city- and state road networks.

We developed synthetic networks of an N×N grid format whereby road segments are only either vertical or horizontal. Every link of the grid is 500m long, with a speed limit 35 miles per hour. Every intersection behaves similar to a stop sign. Varying the parameter N is sufficient to vary the offered road network load, and consequently the simulation runtime performance. Two scenarios were tested with these networks:

1. Least Congestion (*LC)*: The four corner nodes of the grid network constitute the evacuation points (sinks).

2. Most Congestion (*MC*): The network node at the center of the grid is the only sink.

These scenarios serve to test the performance at the different levels of congestion. Consequently, they also represent the variations in the expected least simulation time and greatest simulation time.

For actual city-scale and state-scale networks, we started with the freely available US Tiger road database, and pre-processed it to suit the input format of our system, which includes important information such as segment lengths, latitude/longitude information, and omits other detail that is not necessary in our model.

6. Performance Study

6.1. Synthetic Benchmark Networks

For the synthetic benchmark scenarios, we vary the grid dimension N from 16 to 1024. Constants governing mobility for these networks are kept the same as those of the regional-scale benchmarks. Initially, each node is loaded with 10 vehicles, and we again continue execution until approximately 75% of the total vehicle count has evacuated.

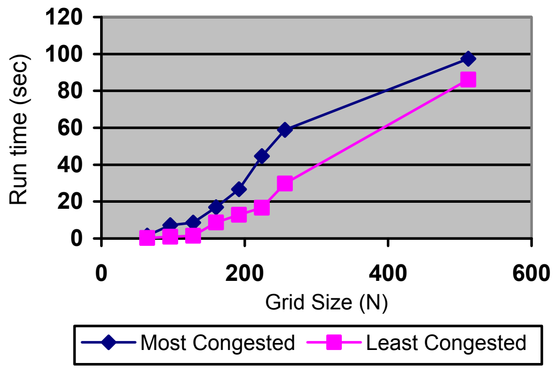

Figure 3: Synthetic benchmark runtime

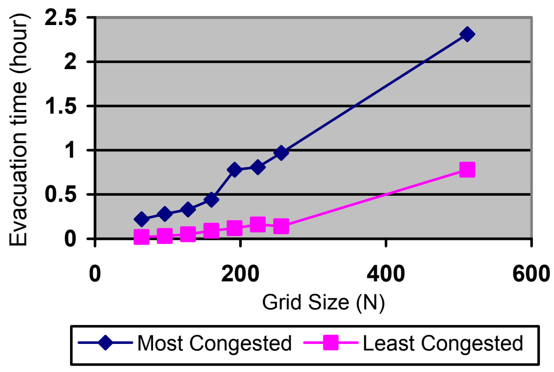

Figure 4: Synthetic benchmark evacuation time

The performance of our simulator on the synthetic benchmarks for the LC and MC scenarios is shown in Figure 3 and Figure 4. While the actual evacuation times are not material to our discussion here, the runtime performance clearly demonstrates the super-real-time speed of the simulation. The same simulation, when carried out on an optimized, state-of-the-art micro-simulator SCATTER[18] failed after N=128, because of the memory needs of the number of links, nodes and vehicles needed to represent the benchmark.

6.2. Region-scale Benchmark Networks

We ran the geographical benchmarks for the simulated evacuation of approximately 75%, with an initial loading of 10 vehicles per network node. In these benchmarks, constants governing vehicular movement that are of concern are: vehicle length = 4.0m, speed limit = 13.3 m/s, and traffic signal time = 1.0s. It also noteworthy that, by controlling texture size used in the simulations, we effectively control the road distance represented by a single texel. This distance is consistently kept greater than vehicle length. Table 2 shows conducted benchmarks with both the estimate evacuation time and the simulation run time.

Table 2: Performance on US networks

			Texture	Evac Time	Run Time
State	Nodes	Links	X×X	Hours	Sec
DC	9,559	14,884	1048576	35.20	54.90
LA	413,574	988,458	4194304	65.07	409.59
TN	583,484	1,335,586	3211264	157.91	353.89
FL	1,048,506	2,629,268	4194304	179.20	611.83
TX	2,073,870	5,116,492	3211264	217.60	777.65

It is important to note that the objective of the performance study is in estimating the expected runtimes of a state-scale scenario. The runtime and evacuation times shown in Table 2 demonstrate that real-time or faster simulations are indeed conceivable for delivering first-order metrics in decision-making by the planning agencies.

Figure 5 and Figure 6 are snapshots of the simulations for the Texas network using our customized display. For a higher simulation speed, the animation display can be turned off. For qualitative assessment purposes, however, the graphical display can be valuable.

Please note that the figures are shown for illustration purposes only, with no specific correspondence to an actual evacuation scenario (we chose an evacuation scenario with only one destination/sink cell, for simple illustration purposes). The focus of this paper is less on the domain-specific study and more on the computational aspect. Validation is beyond the scope of this paper; we plan to undertake validation efforts as future work. However, we have verified our model to be correct in its working, and the images demonstrate the functional status of our simulator, with the ability to sustain the scale of one of the largest state road network sizes among the US states. The simulator can be applied to a realistic study, in which users such as government agencies will specify the actual evacuation points and the actual initial loadings of the roads based on expected population distributions and evacuation plans.

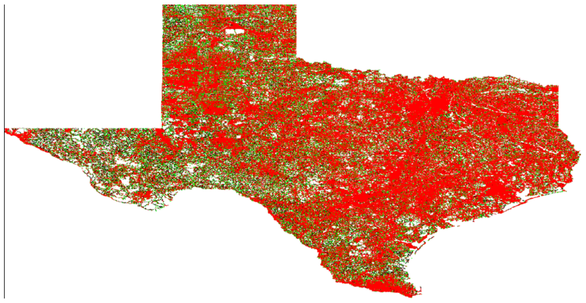

Figure 5: A view of an initial loading of the Texas state road network.

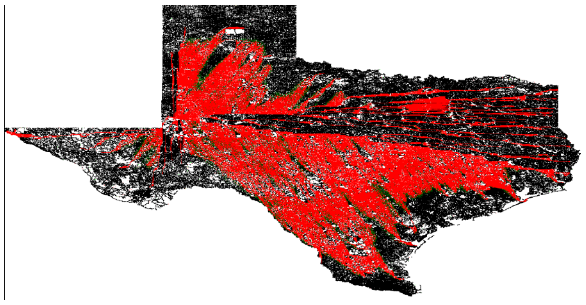

Figure 6: A snapshot of a simulation of the Texas state road network.

Additional scenarios can be viewed at www.ornl.gov/~2ip/RealSim/demo.htm.

7. Summary and Future Work

We presented the design of a field-based mobility model, and described its efficient implementation to exploit the computing power of graphical processing units. Performance evaluation of our system shows real-time (or faster) execution on synthetic and real road network topologies. Scalability to state-sized networks is observed, with some of the largest and densest state networks tested, including Texas and Florida, with millions of road intersections and links.

Improvements are possible to the model and implementation. The mobility model can be enhanced by incorporating greater fidelity, such as non-linear congestion behavioral distributions. Geographical mapping potentially wastes texels due to empty (non-road) regions in space. For dense traffic networks, this is not a problem. For sparse regions (such as Montana), the geographical sparsity of the street network can be result in significant level of wastage. Non-square regions also incur wasted memory space due to empty cells padded to make the rectangle square. Recently, a newer generation of NVIDIA GPUs has become available. These cards, such as the 9800 GTX, offer an increase in core clock speed while remaining comparable in available memory. Should the increase in speed of these prove to be nontrivial over the 8800 GTX, further benchmarks will be conducted.

Acknowledgements

This paper has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the U.S. Department of Energy. Accordingly, the United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

References

[1] O. Franzese and L. Han, "A Methodology for the Assessment of Traffic Management Strategies for Large-scale Emergency Evacuations," in *11th Annual Meeting of ITS America*, Miami, FL, USA, 2001.

[2] B. Bhaduri, C. Liu, and O. Franzese, "Oak Ridge Evacuation Modeling System (OREMS): A PC-Based Computer Tool for Emergency Evacuation Planning," in *Symposium on GIS for Transportation*, 2006.

[3] GPGPU, "General Purpose Computation Using Graphics Hardware," 2005.

[4] M. Pharr and R. Fernando, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*: Addison Wesley Professional, 2005.

[5] ITT Systems & Sciences Corporation, "CORSIM User's Manual, Version 1.04," Federal Highway Administration, U.S. Department of Transportation 1998.

[6] P. D. Prevedouros and Y. Wang, "Simulation of Large G=Freeway and Arterial Network with CORSIM, INTEGRATION, and WATSIM," *Transportation Research Record,* pp. 197-207, 1999.

[7] L. Smith, R. Beckman, D. Anson, K. Nagel, and M. E. Williams, "TRANSIMS: Transportation Analysis and Simulation System," in *Proceedings of the Fifth National Conference on Transportation Planning Methods* Seattle, Washington: Transportation Research Board, 1995.

[8] I. Innovative Transportation Concepts, "VISSIM Simulation Tool," 2001.

[9] K. S. Perumalla, "A Systems Approach to Scalable Transportation Network Modeling," in *Winter Simulation Conference*, Monterey, CA, 2006.

[10] S. Taori and A. Rathi, "Comparison of NETSIM, NETFLO I, and NETFLO II Traffic Simulation Models for Fixed-Time Signal Control," *Transportation Research Record,* pp. 20-30, 1996.

[11] M. Fellendorf, T. Schwerdtfeger, and H.-J. Stauss, "DYNEMO, A Mesoscopic Traffic Flow Model to Analyze ATT Measures," *Transportation Planning Methods,* 1996.

[12] N. H. Gartner and C. Stamatiadis, "Integration of Dynamic Traffic Assignment with Real-Time Traffic Adaptive Control System," *Transportation Research Record,* pp. 150-156, 1998.

[13] D. Bernstein and T. L. Friesz, "Analytical Dynamic Traffic Assignment Models," in *Handbook of Transportation Modeling* Amsterdam, Netherlands: Elsevier Science Publishers, 2000, pp. 181-195.

[14] Q. Yang, H. N. Koutsopoulos, and M. E. Ben-Akiva, "Simulation Laboratory for Evaluating Dynamic Traffic Management Systems," *Transportation Research Record,* pp. 122-130, 2000.

[15] K. Meister, M. Balmer, K. W. Axhausen, and K. Nagel, "A Comprehensive Scheduler for a Large-scale Multi-agent Transportation Simulation," in *International Conference on Travel Behaviour Research*, Kyoto, Japan, 2006.

[16] N. Courty and S. R. Musse, "Simulation of Large Crowds in Emergency Situations Including Gaseous Phenomena," in *IEEE Computer Graphics International*, New York, USA, 2005, pp. 206-212.

[17] C. Reynolds, "Big Fast Crowds on PS3." vol. 2006: Sony Computer Entertainment, 2006.

[18] S. B. Yoginath and K. S. Perumalla, "Reversible Discrete Event Formulation and Optimistic Parallel Execution of Vehicular Traffic Models," *International Journal of Simulation and Process Modeling,* vol. To Appear, 2008.