In Proceedings of the 15th IEEE Intelligent Transportation Systems Conference (ITSC 2012), Anchorage, Alaska, USA, September 2012

# Approximately Orchestrated Routing and Transportation Analyzer: Large-scale Traffic Simulation for Autonomous Vehicles

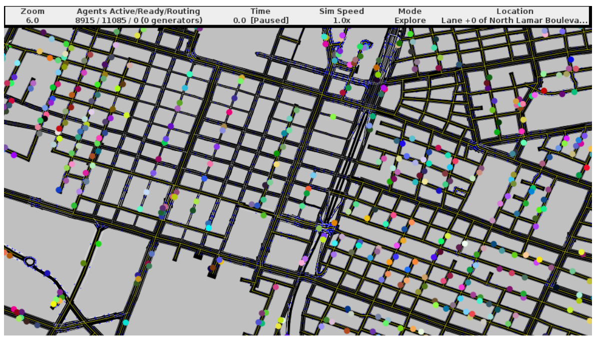
Dustin Carlino, Mike Depinet, Piyush Khandelwal, and Peter Stone
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
{dcarlino,msd775,piyushk,pstone}@cs.utexas.edu

***Abstract*— Autonomous vehicles have seen great advancements in recent years, and such vehicles are now closer than ever to being commercially available. The advent of driverless cars provides opportunities for optimizing traffic in ways not possible before. This paper introduces an open source multiagent microscopic traffic simulator called AORTA, which stands for *Approximately Orchestrated Routing and Transportation Analyzer*, designed for optimizing autonomous traffic at a city-wide scale. AORTA creates scale simulations of the real world by generating maps using publicly available road data from *OpenStreetMap* (OSM). This allows simulations to be set up through AORTA for a desired region anywhere in the world in a matter of minutes. AORTA allows for traffic optimization by creating intelligent behaviors for individual driver agents and intersection policies to be followed by these agents. These behaviors and policies define how agents interact with one another, control when they cross intersections, and route agents to their destination. This paper demonstrates a simple application using AORTA through an experiment testing intersection policies at a city-wide scale.**

## I. INTRODUCTION

Autonomous vehicle technology has made tremendous progress in the last decade. In 2007, six of the competing teams completed the 96 km course set for the DARPA Urban Challenge [1]. They did so while obeying the traffic laws followed by human drivers, navigating along with other moving vehicles, and following correct intersection precedence order. Since then, Google's driverless cars have clocked more than 250,000 km on public roads in urban California, USA [2]. In 2010, researchers from the University of Parma successfully completed an autonomous *intercontinental* run from Parma, Italy to Shanghai, China [3]. The successful completion of all these milestones suggests that autonomous cars are here to stay, and are ever closer to becoming commercially available. With the arrival of autonomous cars, it also becomes possible to optimize traffic in ways not possible for human drivers.

This paper introduces an open source multigent microscopic traffic simulator called AORTA, which stands for *Approximately Orchestrated Routing and Transportation Analyzer*. AORTA is a platform designed for testing autonomous vehicle behaviors and intersection policies. Autonomous vehicles, termed agents, use behaviors to interact with one another, follow intersection policies, and decide on both long term and short term actions. Intersection policies designate when it is safe for an agent to cross an intersection. AORTA's



| Zoom | Agents Active/Ready/Routing | Time | Sim Speed | Mode | Location |
|---|---|---|---|---|---|
| 6.0 | 8915 / 11085 / 0 (0 generators) | 0.0 [Paused] | 1.0x | Explore | Lane +0 of North Lamar Bouleva... |

Fig. 1: Visualizing autonomous agents in downtown Austin, Texas, with AORTA's UI

goal is to allow for the definition of new agent behaviors and intersection policies to optimize autonomous traffic. Additionally, by assigning human-like behaviors such as the "car following model" [4] to agents and traffic-signal-like policies to intersections, AORTA could potentially simulate human traffic.

Like any other microscopic traffic simulator, AORTA needs maps to run simulations. One of AORTA's key features is that it generates maps using real road data available from OpenStreetMap (OSM) [5], a moderated, user-editable interface for world maps. A map for any desired city in the world can be downloaded, which is then parsed by AORTA to set up a scale simulation of the real world in a few minutes. AORTA is available open-source and is easily extensible,[1] making it easy for users to test out a number of agent behaviors and intersection policies in a short time span.

This paper explores the state-of-the-art in traffic simulators in Section II, followed by a description of AORTA's architecture in section III. Use of OSM data in AORTA is explained in Section IV, along with a description of the simulator in Section V. Section VI demonstrates an application built on top of AORTA and presents a simple experiment evaluating different intersection policies in a large scale scenario. The paper then concludes with a discussion on future work.

## II. RELATED WORK

Computational processing power has made excellent advancements in the last two decades. Parallel computing and the use of GPUs have enabled microscopic models of traffic

[1]Code available at `http://code.google.com/p/road-rage`

simulation to generate results at a meaningful scale (city-wide or greater) [6], [7]. As a result, a number of multiagent micro-simulators have been introduced in the past decade. We review relevant simulators and other related work in this section.

OSM data has been used by traffic simulators in the past. MATSim is one such multiagent simulator that focuses on performing large-scale simulations in a relatively small amount of time [8]. Traffic demand is supplied to MATSim in the form of *plans* that an individual may intend to follow in a given day. MATSim aims to improve these plans by attempting to minimize the total amount of time by which individuals are late to their destinations. This goal is somewhat different from that of AORTA, where the routes (demand) supplied by individuals is taken as is. Instead, AORTA focuses on vehicle behaviors and intersection policies to improve the execution of routes.

Another popular open source traffic simulator that can use OSM data is SUMO [9], which shares many similar goals with AORTA. SUMO has been used to study the effect of automated transportation systems, route planning of individual vehicles, and dynamically adapting traffic signal policies to increase traffic efficiency. While SUMO has a flexible system for managing traditional traffic signals, AORTA's general intersection policies are designed with autonomous vehicles in mind. These policies can use traditional schemes such as traffic signals or stop-sign based precedence, or more efficient methods designed specifically for autonomous vehicles [10]. In addition, AORTA's core implementation, written in Scala [11], differs from that of SUMO, written in C++, in that it uses a higher-level language.

Efforts have been made to optimize traffic flow in the context of autonomous vehicles. AIM [10] is one such approach that aims to optimize traffic flow of autonomous vehicles at a given intersection. The AIM approach has been applied to a regular grid of four intersections [12], while AORTA has the potential to apply some of the same research at a city-wide scale to real-world road networks.

Autonomous agents have also been used in the past to model human traffic. For instance, one approach models human merging and lane changing behavior through the use of autonomous agents interacting with one another [13]. Another approach attempted to improve traffic congestion for human drivers by using a network of autonomous intersections. These intersections have the ability to interact with one another and execute a dynamic traffic signal policy minimizing overall wait time [14].

## III. ARCHITECTURE

AORTA is divided into three modular components: the map model, micro-simulation engine, and user interface (UI). The map model transforms OSM maps into AORTA graphs, then answers pathfinding and geometry queries. The simulation engine adds a notion of agents, vehicle dynamics, and collisions. Finally, the UI interactively renders the map and agents. A headless mode also exists to run experiments without the overhead of visualization. The interaction between modules is visualized in Fig. 2.

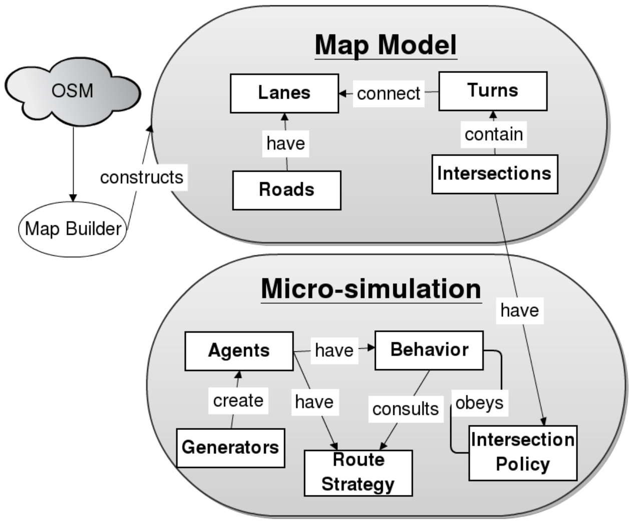


Fig. 2: A summary of AORTA's components.

AORTA's implementation uses Scala, a language implemented on the Java Virtual Machine [11]. Scala provides the advantage of functional programming constructs while still permitting imperative style. Extensions and clients built on top of AORTA can be written in either Java or Scala. The software is open-source and easily extensible. For instance, the stop sign policy used in experiments described in Section VI is implemented in just 70 lines of code.

Sections IV and V describe map construction from OSM data and the simulation engine used in the AORTA framework respectively.

## IV. MAP CONSTRUCTION FROM OSM

One of the key features of AORTA is the simulation of traffic on existing road network data from OpenStreetMap [5]. The data from OSM is not directly suitable for running a simulation and is pre-processed through a *map builder* utility. Section IV-A describes the map model generated by the builder, Section IV-B describes the process of refining OSM data, and Section IV-C describes limitations with this process.

### *A. Map Model*

A map is represented as a directed graph with *lanes* as edges and *turns* as vertices. A *road* groups a set of lanes together, one set for each direction of travel. *Intersections* map the possible turns from incoming to outgoing lanes. Since both lanes and turns support traffic, they are both called *traversables*, meaning agents may exist on them.

The map also has a geometric interpretation, used for visualization and measuring physical distances. A road contains a sequence of points, defining the center line that divides the road into two directions of travel. AORTA interprets these points as straight line segments and projects parallel lanes out using a fixed width. Turns are approximated with a single straight line segment connecting the end of one lane to the start of another.

### B. Map Construction Passes

The builder works incrementally while parsing OSM graphs into the map model, with each pass feeding the next. The map model is then stored in an XML format.

1) OSM encodes many paths besides drivable roads, so the builder first filters these out. Next, the builder marks OSM nodes common to multiple roads, since they implicitly indicate intersections. Overpasses do not share nodes with the roads they cross [15].
2) OSM *ways* (sequences of nodes) include many intersections, so the builder next splits ways into undirected segments of roads between exactly two intersections.
3) The builder multiplies undirected roads into directed lanes in each direction (unless the road is marked one-way). The builder guesses the number of lanes based on OSM's "road type" tag or an explicit number of lanes, when that data is available.
4) The builder constructs turns between incoming and outgoing lanes at each intersection. The angle between the two lanes determines whether the turn is a left, right, straight, or U-turn. When several lanes all cross into fewer lanes, the builder forces merging at the rightmost lanes.
5) Due to OSM's lack of turn data, the directed graph may not be connected. The builder removes small disconnected portions, leaving the largest strongly connected component.

### C. Limitations with OSM

AORTA's map construction process, although flexible, cannot completely account for incompleteness in OSM data. Road type annotations imprecisely imply the number of lanes and typical speed limits. Heuristically enumerating the turns at each intersection often misses common features like right turn-only lanes and shared center left turn lanes. In some circumstances, one intersection appears to be several in close proximity, since OSM has only geometric data, as opposed to functional data, on how roads meet. If OSM encoded functional information, the realism of the simulation would improve. Other simulators have dealt with similar issues [16].

## V. SIMULATION

This section describes AORTA's agent model and the simulator's mechanisms for creating agents, verifying safety, and allowing agents and intersections to control themselves. Comments on current limitations and how to extend the simulator follow.

### A. Simulation Dynamics

AORTA uses microscopic simulation, modeling individual drivers as point agents with a simple acceleration-based dynamics model. Time is modeled discretely, and an agent accelerates at a fixed rate for the duration of a "step" to achieve a new position and velocity. Space is continuous, meaning agents occupy a moving interval of a lane, rather than one fixed tile of the road. Each time step lasts for the same fixed duration, and the simulation does the following for each time step:

1) Introduces new agents into the map
2) Updates the position and velocity of agents based on their choice of acceleration in the previous step
3) Checks for collisions
4) Allows each agent's behavior to choose an action for the next step

### B. Spawning new agents

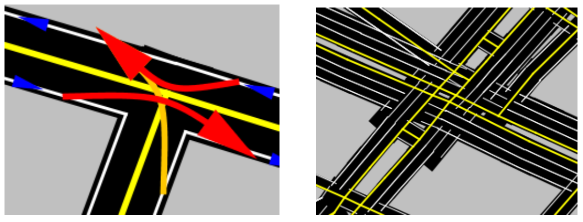
To introduce new agents into the simulation in real-time, users can create *generators* programatically or using the UI. The generator produces the desired amount of traffic by initializing agents from random locations inside a *start* polygon to random destinations within a *goal* polygon. Users can draw these polygons by mouse in the UI, allowing specific traffic patterns like rush-hour scenarios to be easily simulated. A generator polygon may also cover the entire map, uniformly distributing traffic everywhere.

Every step, a generator creates some number of new agents. The generator either immediately performs any timely computation required by the agent's route policy, such as pathfinding, or delegates it to a pool of worker threads to process in the background. Once an agent's route is ready, the agent waits alongside its starting lane (as if it is in a driveway or parking lot). When the lane has no agents that could potentially crash into the new vehicle, the new agent enters the system.

### C. Collision checking

All traversables (lanes and turns) maintain an ordered queue of occupying agents. A collision is detected when two agents reverse order after a time step has completed.

Intersections check for collisions by verifying no two agents are simultaneously performing conflicting turns. AORTA has a low-granularity model of turn conflict. If two vehicles at any point along two different turns could ever come into contact, then the turns are always in conflict, regardless of where the agents actually are along the turns. This is visualized in Fig. 3a. Further work is required before higher granularity can be introduced through space-time tiling intersections [10], due to some of the complex intersection geometries inferred from OSM. An example of such a complex intersection is shown in Fig. 3b.



(a) An agent cannot turn left while agents on the perpendicular road cross

(b) A difficult intersection to tile in New Orleans, Louisiana. Note there are no overpasses pictured.

Fig. 3: Examples of AORTA's modeling of intersections and turns

### D. Agent Behaviors

Each agent has a *behavior* governing it, responsible for obeying intersection policies and avoiding collisions. Before each step, the behavior chooses one of two actions for the agent to perform: disappearing from the map (when the agent is at rest and is done with its route) or accelerating at some rate. Behaviors query current and upcoming lanes and intersections to find other nearby agents and determine all constraints that must be satisfied for a safe choice. The behavior also picks turns once an agent reaches the end of a lane by consulting a route strategy that finds a path to the destination.

### E. Intersection Policies

A behavior interacts with an intersection policy by sending it the agent's desired turn, the agent's speed, the current distance from the start of the intersection, and the length of time the agent has been idling at rest. The policy replies by ordering the agent to *stop* or *proceed*. The behavior will continue to statelessly poll the policy every step until the agent begins the turn. The policy may approve an agent for entry one step and deny it the next, as long as the agent could safely brake without danger of entering the intersection. The policy is free to allow concurrent turns in any arrangement by different agents as long as no two simultaneous turns conflict.

### F. Current Limitations

In the currently released version of the simulator, the ability for agents to change lanes has not yet been implemented. As a result, gridlock potentially occurs. Each agent maintains a safe following distance from the next. If a lane is filled, then an agent may be forced to stop in an intersection mid-turn and block other traffic. The default behavior prevents this by refusing to start a turn before the lookahead engine guarantees no agent will trigger a premature stop.

However, with very high volumes of traffic, it is possible for waiting agents to accumulate and fill lanes to their capacity. When this happens in a circular manner so that the agent at the front of each lane is waiting to perform a turn into another full lane with a similar agent at the front, gridlock [17] occurs: no agent in the system will make progress unless an agent at the front decides to pick a different turn. AORTA has the ability to detect such cyclic dependencies.

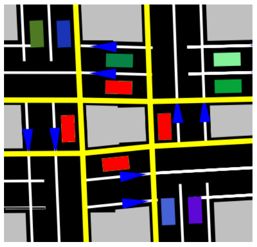


Fig. 4: Each of the red agents wants to turn left, so none of them can.

In many observed cases, the gridlock is caused by AORTA's present lack of lane-changing support: agents loop around intersections such as those pictured in Fig. 4 in order to enter a lane adjacent to their original lane. Since such routes are still legal, it is desirable to prevent or mend this problem without requesting different routes. However, choosing alternative moves (by changing lanes or rerouting) is an important constituent of a sufficient condition for the liveness of a road network [17].

### G. Extending AORTA

There are three main configurable components of the simulation, each with at least one existing implementation: agent behaviors, routing strategies, and intersection policies.

Currently, all agents use a primary behavior that is a generalized, baseline behavior that guarantees not to collide with another agent or enter an intersection at the wrong time. Since it picks the highest safe choice of acceleration at each step, it could easily be extended to mimic human drivers by traveling at some random lesser acceleration or to optimize fuel efficiency by tuning movement.

Intersection policies are one of the most interesting aspects of the simulation to tweak, especially in light of autonomous vehicles. Current policies include traditional stop signs, traffic signals, and an AIM-inspired autonomous reservation policy [10]. The traffic signals are further extensible by assigning groups of non-conflicting turns at each intersection with some duration and offset. Future policies could include signals that adapt timings and refinements to the autonomous reservation policy.

To pick turns, agent behaviors consult a route strategy. Two simple implementations exist already: a static route using A* [18] and a drunken walk that probabilistically moves towards the destination. Users can experiment with dynamic replanning or hierarchical planning [19] without modifying any code other than creating a new route implementation. Route policies could also interact with a single central manager to gain some sort of global insight or with autonomous intersections to avoid congestion.

## VI. EXPERIMENTAL RESULTS

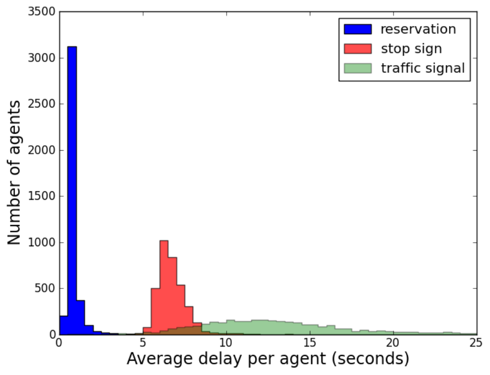
This section presents a simple experiment showing how different policies affect the throughput of intersections. In this experiment, every intersection in the map uses either a stop sign, traffic signal, or autonomous reservation policy. The results demonstrate that policies affect total system performance and that AORTA is a useful tool for evaluating this relation.

### A. Experimental Setup

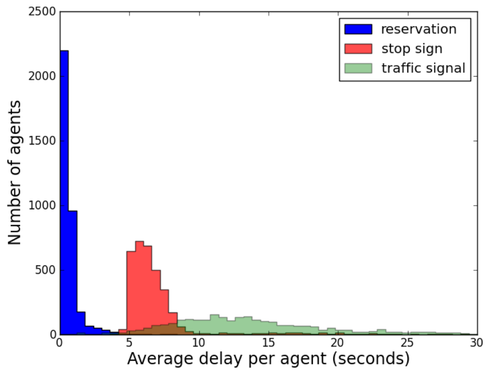
The experiment simulates one hour of traffic in two cities: a $9\times11$ km$^2$ slice of downtown Austin, Texas[2], and a $15\times13$ km$^2$ slice of downtown Houston, Texas[3]. The step duration is fixed at 0.1 seconds. While the intersection policies vary, the agents spawned and their routes remain the same. A

[2]bounded by the Mopac, Springdale Road, Oltorf Street and Koenig Lane

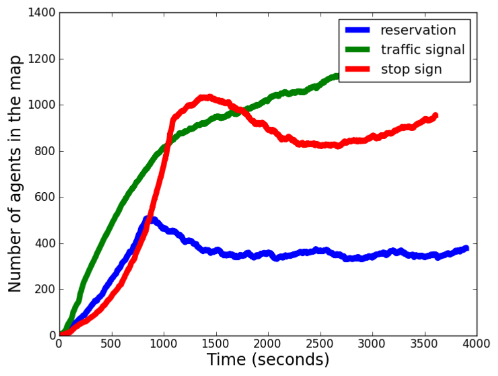
[3]bounded by Ella Boulevard, Eastex Freeway, West Alabama Street and Crosstimbers Street



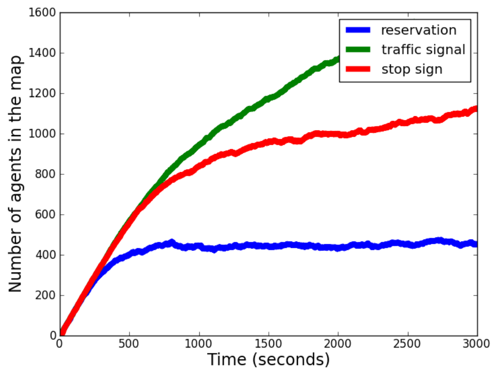
(a) Average delay in Austin



(b) Average delay in Houston



(c) Number of agents driving in Austin



(d) Number of agents driving in Houston

Fig. 5: Autonomous reservations outperform stop signs and traffic signals in these experiments.

generator creates one new agent every 1 simulation second, with a uniformly random starting position and goal.

These experiments are deterministically reproducible. A seed for the pseudo-random number generator, the input graph, and the generators' configuration fully determine the outcome of any simulation [4]. After setting up a scenario in the UI, users can save this configuration for re-simulation.

The speed of simulation depends on the step duration $dt$, the map size, and the number of agents. A measure independent of these factors is the number of agent steps per second. On a 2.4 GHz machine, this can be observed to average at about 150,000, a number comparable to existing simulators [20]. When the time step is 0.1 seconds, this means 15,000 agents can be comfortably simulated in real-time. Further optimizations are planned.

## B. Intersection Policies Evaluated

Stop signs accept agents first-come, first-served, refusing agents until they have idled at rest for 1 second. This delay approximates human reaction time.

The traffic signal policy requires timings and groups of simultaneous turns for each intersection. The heuristic currently assigning these arbitrarily picks one major road, schedules turns from that road to others, then repeats the process from all reachable roads using a breadth-first search. By estimating how long performing one turn and traveling along the next lane takes, the timing of the next intersection can be scheduled so that agents from the first road continue traveling without stopping for several consecutive intersections. This heuristic is far from optimal, meaning the performance of traffic signals described below could be improved, perhaps by incorporating packages such as SYNCHRO [21].

The last policy tested is an autonomous reservation policy inspired by AIM [10]. Because agents explicitly communicate with intersections to schedule their turn, this policy is only suitable for autonomous drivers. This policy groups agents with compatible turns together, allowing new agents to join existing groups. To prevent one heavy direction of traffic from hogging the intersection indefinitely, a timeout preemptively cycles through reservations.

## C. Results

If an agent travels along a lane at the speed limit and does not slow down to wait at an intersection, then it traverses the lane in an optimal amount of time. If the intersection orders it to stop or if other agents in the same lane have slowed down by the intersection's orders, then the intersection has caused some delay. This delay can be used as one measure of intersection performance. Figures 5a and 5b graph the average of this delay per agent. Similar trends

[4]This determinism may break down when generators use worker threads while the simulation is running, since tasks may finish and agents may enter the road at different times.

are observed in both cities: an average of less than 1 second of delay at autonomous reservation intersections, 7 (Austin) or 8 (Houston) seconds at stop signs, and 16 (Austin) or 17 (Houston) seconds at traffic signals using the heuristic described above. These results will vary with different traffic levels and policy configurations, so the important demonstration is that AORTA can be used as a framework for evaluating these policies.

The number of agents driving at some time changes because the continuous generator introduces a new agent every second. This count reaches a steady state when the generator introduces an agent at the same rate as older agents finish their route. Fig. 5c reveals that autonomous reservations let agents reach their destination twice as quickly as the competition in Austin, with a steady-state of about 400 agents. There, the count for traffic signals continues to increase because gridlock occurred in one segment of the map, causing agents to not finish their route. Fig. 5d reveals a similar trend in Houston, except that gridlock occurred both during the trial with traffic signals and stop signs. Unexpected starvation was also observed: although the signals cycle through turns regularly, agents cannot proceed when the previous cycle caused a destination lane to fill and become blocked.

Repeating this or other experiments in more locations only requires those locations to be exported from the OSM website. Editing the intersection policies is similarly convenient, as users can ignore most of AORTA's code-base and just change the logic described in Section V-E. For reference, the most complex intersection policy, traffic signals with the flooding heuristic for timing assignment, is only 400 lines of code in Scala.

## VII. CONCLUSION

This paper has presented AORTA, a new city-scale traffic simulation framework that focuses on configurability. In conjunction with Open Street Maps, AORTA allows users to repeat an experiment in different places with minimal effort. Future work will exploit the flexible infrastructure of interactions between agent behaviors and intersections by exploring dynamic replanning and adaptable signal timings. On top of these structures, applications could experiment with ideas such as dynamic contra-flow [22]. There will also be an effort to improve AORTA as a framework by including a lane-changing model, preventing gridlock, and supporting more agents through parallel computing.

## ACKNOWLEDGMENTS

This work has taken place in the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), and the Federal Highway Administration (DTFH61-07-H-00030).

The authors would like to thank Dr. Michael Quinlan for his support during the seminal stages of AORTA and all OpenStreetMap contributors for collecting the map data used in this work. Efforts by the Freshman Research Initiative, College of Natural Sciences, UT Austin have allowed for the development of this research.

## REFERENCES

[1] J. Volecker. (2007) Autonomous vehicles complete darpa urban challenge. IEEE Spectrum. [Online]. Available: http://spectrum.ieee.org/green-tech/advanced-cars/autonomous-vehicles-complete-darpa-urban-challenge

[2] S. Thrun. (2011) Google's driverless car. TED. [Online]. Available: http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html

[3] J. L. Kent. (2010) Driverless van crosses from europe to asia. CNN. [Online]. Available: http://edition.cnn.com/2010/TECH/innovation/10/27/driverless.car/index.html?iref=allsearch

[4] M. Brackstone and M. McDonald, "Car-following: a historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999.

[5] Openstreetmap: The free wiki world map. OpenStreetMap. [Online]. Available: http://www.openstreetmap.org

[6] K. Nagel and A. Schleicher, "Microscopic traffic modeling on parallel high performance computers," *Parallel Computing*, vol. 20, no. 1, pp. 125–146, 1994.

[7] Z. Shen, K. Wang, and F. Zhu, "Agent-based traffic simulation and traffic signal timing optimization with gpu," in *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*. IEEE, 2011, pp. 145–150.

[8] M. Balmer, M. Rieser, K. Meister, D. Charypar, N. Lefebvre, K. Nagel, and K. Axhausen, "Matsim-t: Architecture and simulation times," *Multi-agent systems for traffic and transportation engineering*, pp. 57–78, 2009.

[9] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo - simulation of urban mobility: An overview," in *SIMUL 2011, The Third International Conference on Advances in System Simulation*, Barcelona, Spain, October 2011, pp. 63–68.

[10] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, March 2008.

[11] The scala programming language. [Online]. Available: http://www.scala-lang.org/

[12] M. Hausknecht, T.-C. Au, and P. Stone, "Autonomous intersection management: Multi-intersection optimization," in *Proceedings of IROS 2011-IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, September 2011.

[13] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transportation Research Part C: Emerging Technologies*, vol. 10, no. 5-6, pp. 351–371, 2002.

[14] V. Manikonda, R. Levy, G. Satapathy, D. Lovell, P. Chang, and A. Teittinen, "Autonomous agents for traffic simulation and control," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1774, no. -1, pp. 1–10, 2001.

[15] Node - openstreetmap wiki. OpenStreetMap. [Online]. Available: http://wiki.openstreetmap.org/wiki/Node

[16] D. Krajzewicz, G. Hertkorn, J. Ringel, and P. Wagner, "Preparation of digital maps for traffic simulation; part 1: Approach and algorithms," *Transportation Research*, pp. 285–290, 2005. [Online]. Available: http://elib.dlr.de/21013/

[17] T.-C. Au, N. Shahidi, and P. Stone, "Enforcing liveness in autonomous traffic management," in *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*, August 2011.

[18] P. Hart, N. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *Systems Science and Cybernetics, IEEE Transactions on*, vol. 4, no. 2, pp. 100 –107, july 1968.

[19] A. Botea, M. Mller, and J. Schaeffer, "Near optimal hierarchical path-finding," *Journal of Game Development*, vol. 1, pp. 7–28, 2004.

[20] J. L. F. Pereira, "An integrated architecture for autonomous vehicles simulation," Master's thesis, University of Porto, 2011. [Online]. Available: http://sumo.sourceforge.net/pdf/mieec1.pdf

[21] D. Husch and J. Albeck, *Trafficware Synchro 6 User Guide*, TrafficWare, Albany, California, 2004.

[22] M. Hausknecht, T.-C. Au, P. Stone, D. Fajardo, and T. Waller, "Dynamic lane reversal in traffic management," in *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC)*, October 2011. [Online]. Available: http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127126