Virtual Chaotic Traffic Simulation

Gaurav Chaurasia[*] B. Radhika Selvamani Nithi Gupta Subodh Kumar[†]

Department of Computer Science & Engineering
IIT Delhi
New Delhi 110016 INDIA

ABSTRACT

This paper presents a novel traffic simulation scheme capable of modeling most forms of urban, chaotic traffic. Different from other lane-based or following-based approaches, ours models traffic as a large navigational problem in an agent based simulation context. While this generalization makes the traffic more reflective of certain scenarios, it also leads to some complexity that we address. It is able to handle dense traffic and selects for each car, independently, the optimal velocity and acceleration to find a path through a fast evolving obstacle network. The selection of parameters at each simulation step is posed as an optimization problem ensuring smooth motion subject to car kinematics. In addition to overtaking, the approach is efficiently able to handle hard cases like behavior at traffic lights and turning. We demonstrate our simulation at real-time rates using average computing resources.

1. INTRODUCTION

Traffic simulation has many applications ranging from urban planning to computer games to movies to learning about driver behavior. It follows naturally that the area boasts a rich body of literature. Much of this literature is in the context of streamlined lane-based traffic flow, as is common in many parts of the world. As a result, many of the ideas are inapplicable to chaotic traffic scenarios as is prevalent in many other parts of the world, where sometimes lane markings may not even exist. We target chaotic traffic simulation. In particular, our goal is to visualize low and high traffic virtual environments: we focus on accurate display of vehicle behavior. Our design allows the display and simulation loops to run asynchronously. This allows the scene to be displayed at a high frequency, even if the simulation frequency is somewhat lower.

Traffic simulation is often categorized into macro and micro simulation (with some examples of hybrids) [20]. Micro-simulation schemes simulate each vehicle, while macro-simulators are concerned mainly with statistics and volumes of traffic flows and bottlenecks. We focus on micro-simulation, for one of our main applications is computer games – one can imagine racing the streets of Bangalore with other gamers, all while "normal" traffic is in full flow. Macro-simulation does not lend itself to such rendering. Furthermore, response to developing traffic situations (like bottlenecks and jams) can be better handled by micro-simulation. We would like our simulation to be realistic enough for urban planning, or for games and movies in the context of, say, urban warfare or criminal pursuit. Our immediate goal is for the physical simulation as well as the resulting graphics to *look* realistic. We leave a detailed physical validation of the simulation for later.

We present a *parameterizable* generic traffic model with few hard-coded assumptions. We neither assume strict lane based driving nor pre-define overtaking trajectories [18], which may apply only to sparse highway traffic. Our technique is able to simulate chaotic urban traffic, and yet converge to more structured movement in low traffic density or low driver aggression levels.

[*]Currently at REVES/INRIA Sophia Antipolis
[†]Contact Email:subodh@cse.iitd.ac.in

2. PREVIOUS WORK

Our problem is primarily one of simulating behavior of multiple agents. Continuum based multi-agent approaches [14] model the system with a set of partial differential equations. Each agent in the system is associated with a force or potential field, and the proximity of the agent to other agents or obstacles is captured in terms of high energy states. The optimal path is one that assures a low energy state to the agent. The behavior of all the agents is the solution of this global system of equations, and therefore such a system is centralized in nature. The major advantage of this approach is that it can capture arbitrary agent motion. And the major challenge is that solving the system of equations can be numerically unstable, especially in situations where the functions are not continuous or differentiable. More importantly, although the results seem plausible at a macro level, individual agent behavior may seem artificial.

Other centralized approaches also suffer from similar unrealistic behavior. The fact is that real drivers are autonomous and are not always able to find the best paths. Indeed, traffic involves a complex and dynamic interaction of multiple independent decision makers in the presence of limited resources (the road). We pursue a decentralized approach, which seems to more naturally capture such complex decision making. Each agent is autonomous and decides its path independently based on its observation of changes around it. We will see how it is easy to incorporate approximate solutions in our framework. Decentralized simulations are also parallelizable and have been shown to work well on multiple cores [17].

One of the very first works on multi-agent simulation in computer graphics can be traced to Reynolds [11], who modeled a flock of birds, called Boids as a particle system. Each boid used its perception of surroundings to find its collision free path independently. Since then, various multi-agent systems have been developed using a variety of approaches to suit specific applications [6, 19].

Our problem also bears resemblance to robot navigation. Our approach is motivated by the class of techniques known as decoupled robot planning [13], as opposed to centralized planning [8]. The centralized approach is popular for robot planning algorithms seeking to find optimal paths. However, our goal is to mimic human behavior. Global path optimization is not a criterion for us. Instead of directly relying on robotic planning, we impose a number of behavioral rules and car dynamics limitations to guide the "robot's" search. Like many decoupled approaches [7], our planning also has two phases; our first phase approximately accounts for other vehicles and the second finds the detailed path.

In a separate domain, car games are popular and provide realistic simulation. However, the focus there is on individual car dynamics. Car navigation is mostly pre-planned. On the other hand, we model only the gross dynamics and focus on the planning, especially in high-traffic and road intersection scenarios.

Previous work in traffic simulations has taken many forms [2, 22]. Public domain agent based simulators exist as well [1, 4, 21]. One of the most recent works in traffic simulation [18] uses agents that follow lanes. They provide a realistic results that fit the data of spatio-temporal traffic data from sensors that monitor highway traffic.

Our technique bears a similarity to that of [9] in that they too model autonomous drivers who make their own decision using individualized parameters like desired and maximum braking, acceleration, speed, inter-vehicle gap, overtake margin, etc. In contrast, we directly use driver aggressiveness level to generate all the other parameters. Also, their path finding is ad-hoc: they make discrete decisions to either follow or overtake based on inter-vehicle gap. Further, in their work gridlock is easily possible and requires the invocation of a special centralized module to relieve it. We require no such module and individual behavioral model is able to find a path.

[5] focuses on intersections and models empirically observed driver behavior as a set of logical pre-conditions and post-conditions. Others have also modeled individual driver behavior using cellular automata [12], case based reasoning [10], robot planning [8, 13], etc. Like many robot planning algorithms, ours is a primarily geometric algorithm with behavior expressed in terms of geometry. We do have a limited configuration space and we have chosen to have vehicles not reverse on a street. Our approach is different from previous traffic simulators in that we do not assume any pre-defined vehicle trajectories or lanes. There isn't even a specific notion of "following" a vehicle. Our experiments with human crowd simulation approaches [16, 17] when applied to traffic did not provide natural behavior.

Most other systems select for each agent, an optimal next velocity. In contrast, our approach selects acceleration, allowing velocity to vary linearly over a time step. We show that this results in a more stable and realistic simulation. Our system does not assume lanes, but in non-crowded situations degenerates to approximately lane based – a phenomenon observed in practice.

3. OUR FRAMEWORK

In the rest of this paper, we refer to a two dimensional road-aligned coordinate frame, when using x and y. Each vehicle also has a local frame of reference aligned with its current velocity (as is the vehicle). We refer to these directions as *forward* and *lateral* directions. At the end of a simulation step, the local frame is updated. Hence, the lateral speed is 0. Similarly, acceleration in the forward direction is due to accelerating or braking and that in the lateral direction is due to turning.

We assume that every vehicle computes its attributes individually. It also maintains an estimate of its neighbors' attributes (which may be imprecise). We use a simple grid-based road partition data structure to locate neighbors. This grid is extremely simple to update as traffic moves on the road.

We assume that the starting point (*source*) and the final destination (*sink*) for each vehicle is known. We have designated sources and sinks at the extremes of the road network. We provide a user interface for specifying the road network on a map. We also allow the user to specify macro-parameters to generate the vehicle agents using a Poisson process.

Given the source and the sink of a vehicle, we use Dijkstra's algorithm to find the shortest route. It's a path in the abstract digraph representing the road map; two-way roads are split into two, one in each direction. The intersections crossed by the path are called *junctions.* The nominal junction is divided into three parts: left, middle, and right, each one third of the road width. Now the *target* of a vehicle is set to the left, the middle, or the right segment depending on the desired direction of the vehicle at the junction at the end of the road. Once at the intersection, the vehicle's next target is again chosen to be the same segment of the beginning of the next road. Thus we reduce the problem to finding a path to the next target.

3.1 Model parameters

Each driver is given an *aggression* level, which is used to derive other properties like preferred speed, safety zones, etc. The aggression level is a normalized value between 0 and 1. For our experiments, we use a normal distribution (with mean=0.5 and variance=0.25) to assign aggression levels randomly to new drivers. Each driver also maintains an estimate of neighboring vehicles' speeds and accelerations. (This estimate is within a range of the real speed and acceleration, respectively. We use a uniform distribution in the range +-10% of the real values. This estimate can, in principle, be individualized based on a driver "expertness" parameter.) Further, each vehicle has the designated preferred speed, the maximum speed, the maximum acceleration and braking power, and the minimum turning radius. Per-driver maximum speed is computed based on a road speed limit. The limit for a driver is generated between $\frac{1}{2}\times$road-limit and $2\times$road-limit, as a linear function of aggression. Similarly, the preferred speed is generated as a random number be-

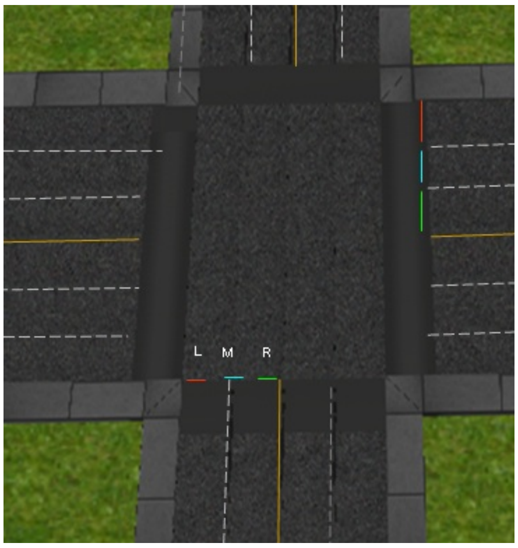

Figure 1: Targets based on route

tween $\frac{3}{4}\times$road-limit and $\frac{5}{4}\times$road-limit, with the condition that the preferred speed is less than the maximum. The maximum and minimum acceleration values (minimum is negative: deceleration) are also assigned in a range. (There is no preferred acceleration, which is deemed to be 0.)

While these choices have not been validated with actual user studies, we have found that the system is stable with a wide range of parameters and once the real parameters are available, it should be possible to plug them in. A few more simulation parameters are introduced later (when the context is clearer) – they can similarly be learned from real data in the future.

3.2 Overview

The simulation proceeds at multiple levels:

- *macro-planner* runs every t_m seconds. For each vehicle, it provides guidance to the vehicle about empty spaces on the road. t_m should be more than a second. At a high level it suggests local *goals* for each vehicle in the intermediate-term. A goal designed to lead it toward the next target.
- A *micro-planner* chooses the next best acceleration, which remains constant over the next simulation step t_{look}, $t_{look} < t_m$. This acceleration is a vector with components respectively tangent and perpendicular to the vehicle (velocity). This acceleration is chosen to avoid collision, while maximizing progress in time t_{look}. We choose t_{look} to be equal to the average human reaction time: 0.7 seconds. Note that it is possible that no collision-free acceleration may exist unless a shorter time-step is employed. In reality, this may indicate a collision.
- Finally, the *graphics update* happens at its own rate, say every t_g. Vehicle positions and orientations are updated for display every t_g, based on the (constant) acceleration value provided by the simulator.

There is no pre-ordained order among the vehicles: the simulation (and rendering) for different vehicles can proceed concurrently. This is useful for parallel contexts.

3.3 Reciprocal Velocity Obstacle

We first discuss a recently developed crowd simulation technique Reciprocal Velocity Obstacle (RVO), which has been shown to give stable results for crowded scenarios [17]. They select the optimal velocity for the next step for each agent individually. In the context of traffic we found that [15] generates jerky movements as vehicles are allowed to make sudden changes to the velocity and the direction. The kinematic constraints in RVO only ensure that the selected velocities have less than the maximum possible speeds. Their model also lacks anticipation: agents merely try to avoid collisions with other agents or obstacles. They don't identify potential colliding or clear paths in advance.

3.4 Our acceleration based model

In order to alleviate the inadequacies of the RVO model, we propose our model with the following salient features:

1. Our system is based on selecting the *best* acceleration to maintain during the next simulation step t_{look}, one that respects the physics of vehicle movement. This ensures that the variation of velocity of any vehicle over time remains smooth and we do not observe jerky movement.
2. We model driver aggression by defining path planning as a function of aggression level of the driver.

4. PATH PLANNING

Given the next target, we break the task of reaching there into smaller steps. Macro-planner provides approximate intermediate goals leading to the target. It is a light-weight operation only designed to help guide the detailed micro-planner. Micro-planner determines a collision free path trying to reach the intermediate target in several micro-simulation steps. Micro-plan is the actual path planner: it computes the constant vehicle acceleration to maintain over the micro-simulation time step. Macro-plan uses high-level information to create an 'advice' for micro-planner helping restrict its search space.

4.1 Macro-plan

Macro-planner overlays a uniform grid on the road and charts *traversability* of grid-cells in a manner similar to [3]. For each vehicle, it assumes that the average relative configuration of traffic around it changes little over macro-simulation time step t_m. Thus it bases its advice on extrapolated path of each vehicle j in the neighborhood.

Macro-planner first (Figure 2) computes potential traversability of each cell in the grid. It starts with the extrapolated path, or the *track* of every vehicle j. Cells on the track are assigned a traversability value of 0, their immediate neighbors a value of 1 and the next neighbors that of 2, etc. The traversability is propagated thus for each track. Clearly, multiple tracks may propagate certain values to any given cell. Among all such values propagated to a cell, finally the smallest is chosen. Thus the final traversability of each cell measures the likelihood of collision free path through the cell.

For vehicle i, a set of goal cells is chosen: the cells at the distance $t_m * p_i$, where p_i is vehicle i's preferred speed. The

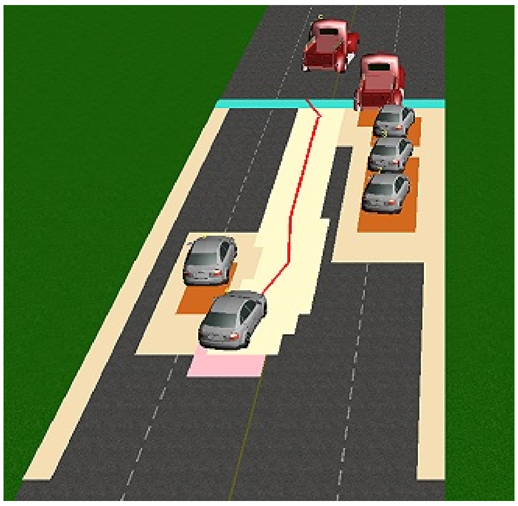

Figure 2: Landmarks computed by Macro-planner

progress cost of these cells is set to 0 – this is where the vehicle would like to be after t_m. Other cells have a positive cost. Again the progress cost is propagated outwards from the zero cost cells. If the progress cost of a cell is c, $c+1$ is propagated to the neighbor. The overall cost of a cell is its progress cost + $(Max\ Traversability - Cell\ Traversability)^3$, where $Max\ Traversability$ is the maximum of all cells' traversability values. This is a normalization of the traversability across cells. The power of 3 is chosen empirically using concepts similar to [3]. The plan finally chooses the target cell which has the least cost path to it.

4.2 Micro-plan

For each vehicle i, the macro-planner suggests a track, divided into a sequence of landmarks. Micro-planner attempts to find the acceleration that vehicle i must maintain for the next t_{look} to reach near the next landmark. It may choose a different course to avoid collision but attempts to to continue in the direction of the track.

The task of the micro-planner for i is to choose its acceleration vector, given its approximate knowledge of other vehicles' speeds and accelerations. The motivation is that even the aggressive driver wants to avoid collision.

During a small interval with fixed acceleration A, the position of vehicle j is given by $P(t) = S + Ut + \frac{1}{2}At^2$, where S is j's starting location and U its initial velocity. S, U, P, A are all 2D vectors.

The path $P(t)$ can now be considered a 3D curve in x, y, t space. However, since i only estimates that the magnitude of U lies in a range [k:K], U may be in the range $[kU : KU]$. Similarly, the estimates of the forward and lateral acceleration ranges give a range for each component of A.

As a result $P(t)$ lies in a range, i.e, each component of $P(t)$ lies in a range, namely, $(A^x_{min} : A^x_{max})$ and (A^y_{min}, A^y_{max}). Thus $P(t)$, for a given t, is bounded by the rectangle with corners at

$$(S^x + kU^x t + \frac{1}{2}A^x_{min}t^2, S^y + kU^y + \frac{1}{2}A^y_{min}t^2)$$

and

$$(S^x + KU^x t + \frac{1}{2}A^x_{max}t^2, S^y + KU^y + \frac{1}{2}A^y_{max}t^2)$$

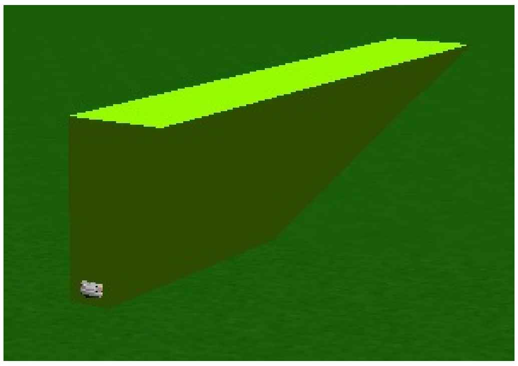

Figure 3: t-shape in x, y, t space

Therefore path $P(t)$ is not the trajectory curve of a point but rather of a rectangle whose size grows non-linearly as it traverses t. This represents a 3D solid in $[x, y, t]$ space (Figure 3). We call it the t-shape of a vehicle. Vehicle i does not collide with vehicle j, if their t-shapes do not intersect.

Now our micro-planning problem reduces to finding the next acceleration for every vehicle i, given its current velocity U and the t-shapes of each neighboring vehicle j. We assume that i knows its current parameters and computes its next real acceleration, not just an estimate.

Note that $P(t)$ is just a point on the vehicle. Its shape must be accounted for. We use the vehicle's bounding box B_j for this purpose. Now the extremal coordinates of $P(t)$ are given by B_j. We only need to modify the starting point S to obtain the full vehicle trajectory:

$$(S^x_{min} + kU^x t + \frac{1}{2}A^x_{min}t^2, S^y_{min} + kU^y + \frac{1}{2}A^y_{min}t^2)$$

and

$$(S^x_{max} + KU^x t + \frac{1}{2}A^x_{max}t^2, S^y_{max} + KU^y + \frac{1}{2}A^y_{max}t^2)$$

where, S^x_{min} is the minimum x coordinate of B_j and so on. Furthermore, we allow a safety zone around vehicle j to account for the fact that two vehicles are not allowed to get too close. The safety zone is a function of the speed of vehicle i and its aggression value. Aggression value is linearly mapped to a *safe-time* lying between 0.5 and 3 seconds. The safe distance is now $V_i \times$ *safe-time* computed separately for the x and y components. Here V_i is i's velocity. B_i is padded with this safety zone.

To simplify the search of acceleration for vehicle i, we reduce its t-shape to a curve by using $B_{ij} = B_j^{\oplus} B_i^{(padded)}$ instead of B_j, where $\oplus$ denotes Minkowski sum [7].

In fact, to further simplify collision detection, we replace each obtained t-shape of j with a bounding polyhedron: its curved sides are replaced by planes. Note that t varies from 0 to t_{look} and t_{look} is small (we use 0.7 seconds to match average human reaction time), so a linear approximation is reasonable. This is easily done by increasing the maximum x and y at both $t = 0$ and $t = t_{look}$. The shift is given by equating tangent of the plane equal to the surface at t where:

$$\frac{Ut_{look} + \frac{1}{2}At^2_{look}}{t_{look}} = U + At,$$

i.e., at $t = \frac{t_{look}}{2}$. This distance is $\frac{1}{8}At^2_{look}$. See Figure 4 for illustration.

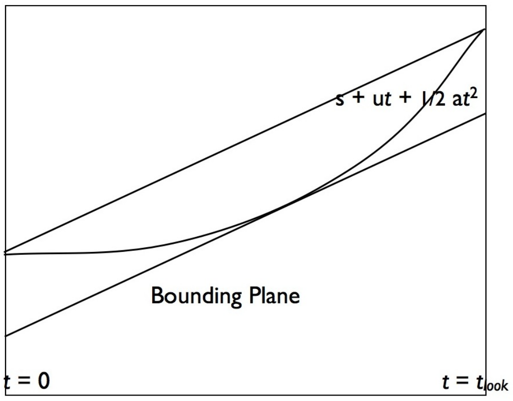

Figure 4: Bounding a quadratic by a linear curve. We choose a bounding plan which is tangent to the quadratic surface. A choice of tangent of given by the end points.

The solution, $t = t_{tan}$ is a vector and the value of expansion in x and y are given by $Ut_{tan} + \frac{1}{2}At_{tan}^2$. We expand the rectangles by this amount and draw planes connecting corresponding edges.

The problem now reduces to finding, for each vehicle i, a path for its center that is free of collision from the modified t-shapes of its neighbors. The search for the path is posed in terms of a search for acceleration. The acceleration curve is approximated again by a straight line for efficiency (although one might resort to quadratic programming). The maximum distance between the curve and the line is also padded to the Minkowski sum B_{ij} to ensure that if the straight line has no collision, the curve hasn't either.

Our goal now is to find the "best" acceleration that leads to a collision free path. Collision can be tested by considering the plane equations of the t-shapes and posing a linear programming (LP) problem – we want to reach a point that is on the exterior of each plane.

In its generic form, each possible acceleration can be associated with a complicated reward function in terms of motion continuity, comfort-zone of the driver and progress toward targets. If LP is feasible, we need to find a point in the feasible space that maximizes the reward. Sampling is a common approach to optimize arbitrary reward functions.

Although we implemented sampling based optimization as well, we found that the following simplified scheme works surprisingly well. We choose an objective function based on the macro-plan track. The track provides an estimate of point $P(t_{look})$. Assuming the current position is P_0, $P(t_{look}) - P_0$ provides an objective function leading to the best progress along the track. We incorporate this into the LP.

4.2.1 Final choice

Once the acceleration is found by the linear programming solution, it may violate physical constraints. For example, it may require a lateral acceleration requiring fast turn. We incorporate the notion of turning radius by limiting the lateral motion as a fraction of the forward motion.

$$U^{lat}t_{look} + \frac{1}{2}A^{lat}t_{look}^2 < r[U^{forw}t_{look} + \frac{1}{2}A^{forw}t_{look}^2]$$

The constraint can be posed in the road's frame of reference and incorporated into the linear program

$$x < \alpha y$$

Similarly absolute ranges for the speed and acceleration can be limited (in lateral and forward directions) by more inequalities in the road's coordinate frame. All constraints lead to constraints on x and y. For example if $A^x \leq A^x_{max}$, $x \leq U^x t + \frac{1}{2}A^x_{max}t^2$

Unfortunately, acceleration limits lead to quadratic constraints. We again approximate them by linear constraints.

LP may not lead to a solution as it may be infeasible. This means no acceleration or braking exists to guarantee a collision-free path for vehicle i for time t_{look}. This may be because a collision is imminent or simply that the estimates of vehicle i are too conservative. In this case a smaller value of t_{look} may succeed. In our implementation we check once for $\frac{t_{look}}{2}$ before declaring collision.

4.2.2 Special Cases

Red lights.

Road is modeled as another obstacle with zero velocity and acceleration. Similarly, traffic lights also are modeled as artificial obstacles. Ones that appear when the light is red. Unfortunately, the sudden appearance of an obstacle can easily lead the simulation into a collision state. Traffic lights are usually visible from a distance and this "preparation to stop" behavior is missing in our geometric algorithm: deceleration happens only in response to stopped vehicles or "sudden traffic barriers." In order to model a more ordered slow-down in the presence of red-light, we force the preferred speed of vehicles to modulate as they approach the red-light. Use of adaptive preferred speed ensures a slowing down before an approach to the intersection. The preferred speed for a vehicle approaching a red light is decreased non-linearly as a function of distance from that red light. This forces the vehicle to slow down and come to a halt gracefully. Our overtaking model as explained above ensures that vehicles still overtake other stopped ones and find vacant spots for themselves.

Turns.

Sudden turn is also unrealistic. Real drivers planning to turn do slow down when approaching an intersection. Otherwise, the lateral constraints would force the vehicles to fail to negotiate the turn. We decrease the preferred speed of vehicles approaching a turn – even without the red light – to ensure that their speed just before the turn is close to a *preferred turning speed.* Their aggression levels in the intersection are also turned low (we use 0). High aggression levels lead to constant attempt to overtake leading to unnatural looking turns. We conjecture that drivers do tend to drive less aggressively inside intersections. Vehicles are further constrained to remain on the right of onward coming vehicles (assuming driving on the left hand side of the road).

5. SIMULATION RESULTS

We implemented the model as described in the previous chapter in OpenGL on a Windows platform. Each simulation has agents injected with randomized attributes at random time intervals.

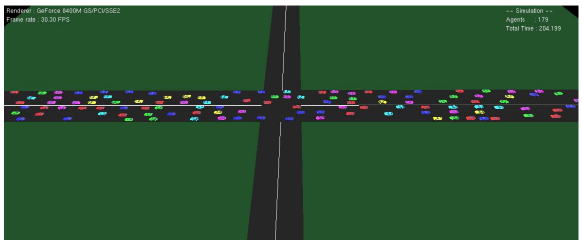

Figure 5: High vehicle density

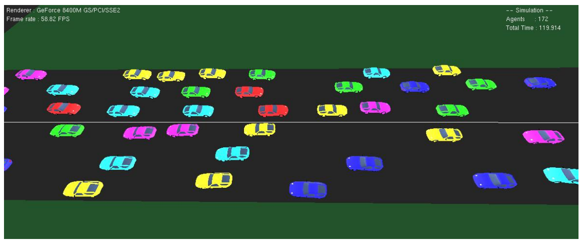

Figure 6: Automatic lanes under dense traffic

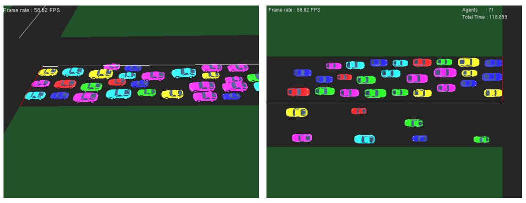

Figure 7: Vehicles stopped at red light

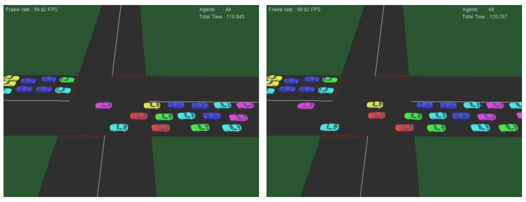

Figure 8: Vehicles emerging out of a red light

5.1 Vehicular behavior

This section presents the visual or qualitative results of the simulation under various traffic conditions.

5.1.1 *Overtaking*

Previous traffic simulations define overtaking in the form of lane changes, and the lane changing trajectory is also predefined, which makes overtaking predictable and robotic e.g. [18] defines a hard coded *clothoid* trajectory for lane changes. Our model does not assume any overtaking path. The simulations show vehicles overtaking on trajectories as per their own speeds and the space available. Moreover, some vehicles try to change 2 - 3 lanes at a time[1]. The simulated trajectories are similar to those defined in earlier lane based simulators (because of vehicle dynamics being respected), but the advantage is that there is no need to predefine any paths.

5.1.2 *Dense traffic*

The simulation results show that in crowded situations, vehicles tend to automatically arrange in virtual lanes in an attempt to get as tightly packed as they can. This can be seen in Fig. 6 and also when vehicles are stopped at red lights.

5.1.3 *Behavior at traffic lights*

Traffic lights are one of the trickier cases of traffic simulation, because at traffic lights the density of vehicles is much higher than on moving roads, and also the vehicles themselves are in somewhat aggressive states - trying to find the smallest vacant gaps to fit into, and then trying to accelerate as soon as they can. The simulation results for traffic lights show that the path planning model is capable of handling complicated vehicle behavior at red lights. In real life, vehicles tend to arrange themselves in a tightly packed chunk at red lights. Our model simulates this well because of proper overtaking behavior. A vehicle approaching a red light tries to overtake already stopped vehicles and in the process ends up trying to fit itself in any vacant space it could find as shown in Fig. 7. The red light is conveyed only to the front most row of stopped vehicles, the other vehicles remain stopped for the lack of space ahead of them. When the red light is released, the front row starts accelerating. The vehicles behind start moving only when the row in front of them has moved a little as shown in Fig. 8. Not all vehicles start accelerating immediately, rather they start only when they see sufficient space. This appears exactly like real life traffic where every vehicle starts moving only when the vehicle in front of it has made sufficient room. As soon as vehicles start gaining speed, there is some lane changing and one can observe that at times one lane starts moving out faster than others.

5.1.4 *Behavior at turns*

The special case handling of turns as explained in the previous section ensures that a vehicle approaching a turn has already slowed down sufficiently according to its expected turning radius. Vehicles move at nearly constant speeds along the turn in low aggression modes, much like lane based driving without any lane changes. When vehicles which have to turn are stopped at a red light (Fig. 9), they start turning with nearly zero speed. In this case, they slowly accelerate up to their maximum turning speeds while turning. After completing the turn, they again switch to their usual driving modes.

5.1.5 *Effect of lookahead time*

Look ahead time t_{look} is the time period for which each vehicle is planning in advance. The simulation results for smaller lookahead time show less but sharper overtaking. For larger lookahead times, the behavior is much more realistic and the degree of overtaking becomes higher and smoother. This is because at smaller lookahead times, every vehicle tends to react to the presence of neighbors only when it is sufficiently close to it. So it either keeps following the vehicle in front of it (creating beelines) or tries to overtake at a sharp angle. The effect of smaller lookahead time is

[1]Vehicle changing 2-3 lanes simply means that the lateral distance covered by the vehicle while trying to overtake was equivalent to 2-3 virtual lanes. There are no lanes as such in this system.

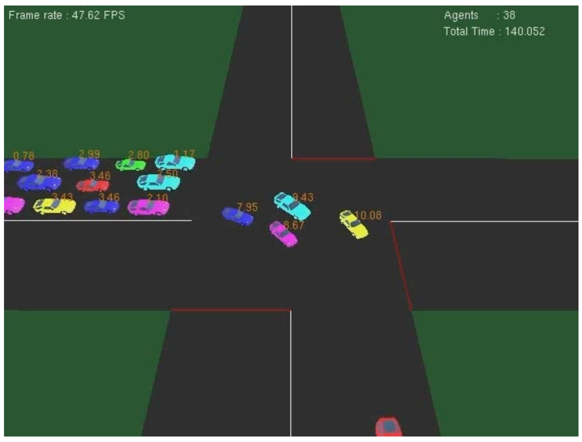

Figure 9: Vehicles at turn (speeds in km/hr marked)

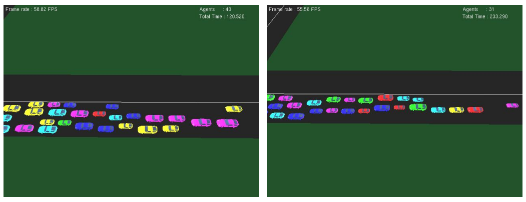

Figure 10: Stopping at red light with lookahead time = 0.7 sec and 0.4 sec. Note the unrealistic queuing up of vehicles one behind the other.

best exemplified by the case of vehicles stopping at a red light. A vehicle with smaller lookahead time cannot anticipate stopped vehicles or find vacant spaces early enough. By the time it realizes a jam ahead, it is already too close to other vehicles to turn and find itself a vacant spot further up. This leads to stacking of vehicles one behind another which gets worse as the lookahead time is decreased as can be seen in Fig. 10. For larger lookahead time, a vehicle approaching a red light anticipates other already stopped vehicles in front earlier and has time to find a vacant spot for itself.

5.2 Performance

We implemented a prototype of the simulation on a machine with Windows XP and 3.3 GHz Intel core 2 Duo processor. Although we did not focus on code optimization, microsimulation can proceed at over twenty frames a second for more than a hundred vehicles. Even for a sufficiently large scene with 500 vehicles, the simulation can complete in time less than $0.5t_m$, i.e., 0.35 seconds in our experiments

No. of Vehicles	Simulation time
50	16 ms
75	32 ms
100	47 ms
200	94 ms
300	172 ms
400	242 ms
500	318 ms

Table 1: Simulation Performance

(see Table 1). This is sufficient granularity to allow rendering at interactive rates (more than 30 frames a second). The rendering thread receives updates to vehicle accelerations at the simulation rate. Indeed, for realistic car models we found rendering to be the bottleneck. Again, no rendering acceleration techniques were employed.

6. FUTURE WORK

This paper presents a simple yet fairly complete model for simulating traffic. There is still scope for optimizing and parallelizing this model so that it can simulate thousands of vehicles in real time. We have presented a visual analysis of the simulation in this paper. A very important future step would be to compare simulation statistics with real traffic flow data. Such quantitative validation would be critical before this model can be put to scientific use.

7. REFERENCES

[1] S. Algers, E. Bernauer, M. Boero, L. Breheret, D. Taranto, M. Dougherty, K. Fox, and J. F. Gabard. Smartest project: Review of micro-simulation models. *EU Project No: RO-97-SC*, 1059, 1997.

[2] A. Aw and M. Rascle. Resurrection of second order models of traffic flow. *SIAM Journal of Applied Math*, 60(3):916–938, 2000.

[3] P. Batavia and I. Nourbakhsh. Path planning for the cye personal robot. In *Proc. International Conference in Intelligent Robots and Systems(IROS 2000)*, pages 15–20, 2000.

[4] A. Byrne, A. de Laski, K. Courage, and C. Wallace. Handbook of computer models of traffic operations analysis. *Technical Report FHWA-TS-82-213*, 1982.

[5] A. Doniec, R. Mandiau, S. Piechowiak, and S. Espié. A behavioral multi-agent model for road traffic simulation. *Eng. Appl. Artif. Intell.*, 21(8):1443–1454, 2008.

[6] J. Ferber. *Multi-Agent System: An Introduction to Distributed Artificial Intelligence.* Addison Wesley Longman, 1999.

[7] J. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[8] T. Li and H. Chou. Motion planning for a crowd of robots. In *in International Conference on Robotics and Automation (ICRA)*, pages 4215–4221. IEEE Press, 2003.

[9] P. Paruchuri, A. Pullalarevu, and K. Karlapalem. Multi agent simulation of unorganized traffic. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 176–183, New York, NY, USA, 2002. ACM.

[10] D. A. Reece and S. A. Shafer. A computational model of driving for autonomous vehicles. *Transportation Research*, 27(1):23–50, 1993.

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

[12] H. J. Ruskin and R. Wang. Modelling traffic flow at an urban unsignalized intersection. In *Proc. International Conference on Computational Science*, pages 381–390, 2002.

[13] M. Saha and P. Isto. Multi-robot motion planning by incremental coordination. In *In IROS*, pages 5960–5963, 2006.

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

[15] J. van den Berg. RVO library. `http://www.cs.unc.edu/~geom/RVO/Library/`.

[16] J. van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA*, pages 1928–1935. IEEE, 2008.

[17] J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin. Interactive navigation of multiple agents in crowded environments. In *SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 139–147, New York, NY, USA, 2008. ACM.

[18] J. van den Berg, J. Sewall, M. Lin, and D. Manocha. Virtualized traffic: Reconstructing traffic flows from discrete spatio-temporal data. In *IEEE Virtual Reality Conference 2009*, pages 183–190, March 2009.

[19] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 200.

[20] J. C. Williams. Macroscopic flow models. In C. M. N. Gartner and A. Rathi, editors, *Traffic flow theory*, page chapter 6. Oak Ridge National Laboratory, 1997.

[21] Q. Yang and H. Koutsopoulos. A microscopic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C*, 4(3):113–129, 1996.

[22] H. M. Zhang. A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B*, 36(3):275–290, Mar 2002.