To appear in ACM TOG 30(6).

Interactive Hybrid Simulation of Large-Scale Traffic

Jason Sewall[*] Intel Corporation
David Wilkie[†] Ming C. Lin[‡] University of North Carolina at Chapel Hill

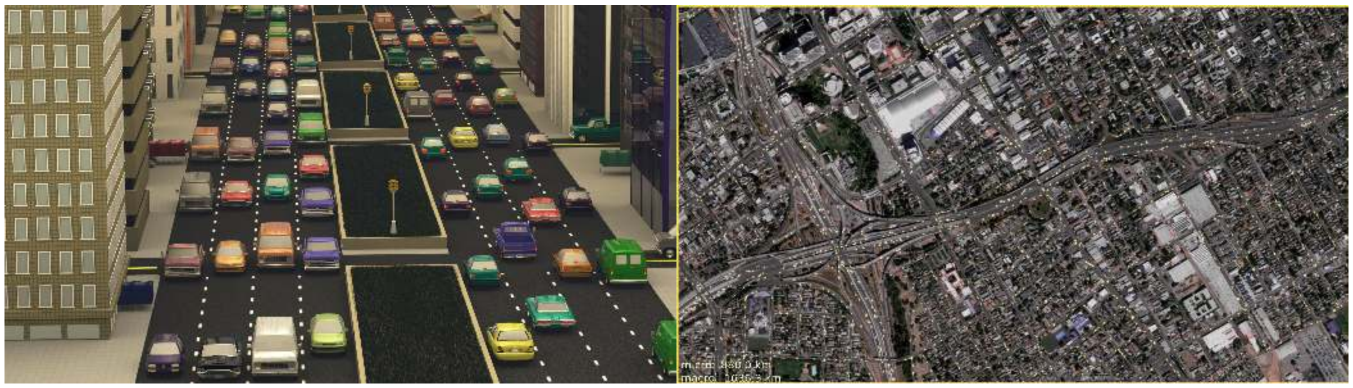

(a) (b)

Figure 1: *(a) Interactive 3D visualization of urban traffic; (b) Augmenting a satellite earth map of a metropolitan region with real-time moving traffic consisting of tens of thousands of vehicles using our method.*

Abstract

We present a novel, real-time algorithm for modeling large-scale, realistic traffic using a hybrid model of both continuum and agent-based methods for traffic simulation. We simulate individual vehicles in regions of interest using state-of-the-art agent-based models of driver behavior, and use a faster continuum model of traffic flow in the remainder of the road network. Our key contributions are efficient techniques for the dynamic coupling of discrete vehicle simulation with the aggregated behavior of continuum techniques for traffic simulation. We demonstrate the flexibility and scalability of our interactive visual simulation technique on extensive road networks using both real-world traffic data and synthetic scenarios. These techniques demonstrate the applicability of hybrid techniques to the efficient simulation of large-scale flows with complex dynamics.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling I.6.8 [Simulation and Modeling]: Types of Simulation—Animation

Keywords: traffic, road networks, hyberbolic models

1 Introduction

Automobile traffic is ubiquitous in the modern world. Traffic simulation techniques for animation, urban planning, and road network design are of increasing interest and importance for analyzing road usage in high-traffic urban environments and for interactive visualization of virtual cityscapes and highway systems.

One of the hallmark applications of 3D graphics is VR flight [Pausch et al. 1992] and driving simulators [Cremer et al. 1997; Donikian et al. 1999; MIT 2011; SUM 2009; Wang et al. 2005] used for training. As today's virtual environments have evolved from the earlier single-user VR systems into online virtual globe systems and open-world games (e.g. Grand Theft Auto), the need to simulate large-scale complex traffic patterns — possibly informed by real-time sensor data — has emerged. These new social, economic, and environmental applications present huge computational demands. Road networks in urban environments can be complex and extensive, and traffic flows on these roads can be enormous, making it a daunting task to model, simulate, and visualize at interactive rates. This paper introduces a hybrid simulation technique that combines the strengths of two broad and disparate classes of traffic simulation to achieve flexible, interactive, and high-fidelity simulation even on very large road networks.

Two classes of simulation techniques are most commonly used in modeling traffic flows. *Agent-based* traffic simulations, also known as *microscopic* methods, determine the motion of each vehicle individually through a series of rules. These rules are easy to vary on a car-to-car basis; such simulation techniques are well-suited to individual vehicles with inhomogeneous governing behaviors. *Continuum*, or *macroscopic*, approaches describe the motion of many vehicles with aggregated behavior; numerical methods are used to solve partial differential equations (PDEs) that model large-scale traffic flows. While agent-based simulation techniques can capture individualistic vehicle behavior, continuum simulations maximize efficiency. Our technique dynamically partitions the simulation domain between these two simulation methodologies to take advantage of their complimentary features.

The resulting hybrid technique can automatically and dynamically select the appropriate method based on user-specified application needs, such as zooming in to a specific region, quickly browsing through large metropolitan areas, maintaining constant simulation rates, etc. We have developed techniques to integrate continuum traffic simulation with agent-based vehicle simulators, enabling distinct regions of the road network to be handled by separate simulation techniques without disrupting the flow of vehicles between regions. We present techniques based on averaging as well as the

[*]jason.sewall@intel.com
[†]wilkie@cs.unc.edu
[‡]lin@cs.unc.edu

Poisson process for handling the transition of vehicle representations between continuum and discrete simulation areas and discuss how the constituent simulation components are adapted to handle this conversion.

The technique we present has a number of attractive properties: **efficiency** — it offers a low-overhead trade-off between performance and simulation fidelity, based on the application requirements; **versatility** — it is applicable to both *real-world* and synthetic traffic and road network data; and **extensiblity** — it is general and suitable for integrating a wide variety of models.

To demonstrate our method, we show a real-time visualization of metropolitan-scale traffic flows on a urban scene and an augmented aerial street map, such as those shown in Figure 1. We also validate the simulation results using real-world traffic data with *string-distance metrics* and analyze the performance of our technique on modern architectures.

2 Related Work

Since the influential 'boids' model of [Reynolds 1987], there has been interest in agent-based simulations and crowd dynamics, covering important sub-problems ranging from motion planning and collision avoidance, to behavioral modeling (see the recent surveys of [Pettré et al. 2008; Pelechano et al. 2008] for more detail). There has been comparatively little investigation of vehicles and traffic flows for visual simulation. Recently, [Sewall et al. 2010] proposed a continuum simulation model for real-world traffic that uses particle-like tracers for visual description of the traffic flow; continuum formulations for crowd dynamics have been proposed by [Treuille et al. 2006; Narain et al. 2009]. There has also been renewed interest in synthesizing vehicle motion using algorithmic robotics techniques [Go et al. 2005; Sewall et al. 2011].

In general, there are three broad classes of traffic simulation techniques: the agent-based *microscopic* and continuum-based *macroscopic* techniques mentioned in Section 1, and the kinetic *mesoscopic* techniques based on Boltzmann-like statistical mechanics. Agent-based techniques were first introduced by the car-following model of [Gerlough 1955]. Later work by [Newell 1961], [Algers et al. 1997], and [Helbing 2001] incorporated more features of traffic into the model. [Nagel and Schreckenberg 1992] describe a method for agent-based traffic simulation using cellular automata.

Macroscopic simulation of traffic was initially developed independently by [Lighthill and Whitham 1955] and [Richards 1956] based on observed similarities between one-dimensional compressible gas dynamics and the way traffic flows along a lane. The resulting so-called 'LWR' equation is a scalar, nonlinear partial differential equation describing the motion of traffic in terms of density, i.e. 'cars per car length'. Because the LWR equation is a scalar equation for density, the velocity of traffic at any point is given by an equation of state; traffic velocity is based solely on the traffic density.

To achieve a more complete model of traffic where velocity does not depend wholly on density, [Payne 1971] and [Whitham 1974] proposed 2-variable systems of equations — later dubbed the 'Payne-Whitham' model — based more directly on the Euler equations of gas dynamics. This was later shown to have incorrect behavior by [Daganzo 1995], who pointed out that the isotropy of gas dynamics was not compatible with traffic dynamics.

More recently, [Aw and Rascle 2000] and [Zhang 2002] each proposed 2-variable models of traffic flow with correct anisotropic behavior. [Lebacque et al. 2007] noted that these two models could be unified through a change of variables, and dubbed the system the 'Aw-Rascle-Zhang' (ARZ) system of equations.

Traffic simulation presents unique challenges in acquiring and representing simulation domains. One option is to use real-world road networks; digital representations of real-world road networks are widely available in the form of connected polylines. The forthcoming work of [Wilkie et al. 2011] describes techniques for synthesizing useful road networks from publicly-available GIS data. On the other end of the spectrum, procedural modeling of virtual cities and roads has been the subject of notable investigations in computer graphics; recent work by [Galin et al. 2010] and [Chen et al. 2008] has enabled the synthesis of detailed, realistic urban layout and roads. Low-cost accessibility of such further enhances the value of our work.

3 Method

In this section, we briefly discuss the data structures used in our simulation and present a description of our hybrid simulation technique for real-time traffic visualization.

3.1 Road networks

Because our method visualizes the behavior of vehicles in both urban and rural settings, in lane-changing scenarios and intersection crossings, we require detailed road network data as our domain. We use a lane-based representation that works well for both agent-based and continuum simulations.

Arc roads Although polylines are often used in digital networks to represent road shapes, they can lead to visible artifacts in the motion of vehicles along these roads. We use a representation based on polylines with fitted circular arcs that is simple, efficient, and allows for realistic, visually smooth vehicle motion. Arc roads are described at length in the supplementary Appendix A.

3.2 Overview of simulation methodology

Our hybrid traffic simulation is both efficient and flexible, combining the strengths of continuum and agent-based techniques. At any given point in simulation, the road network consists of mutually exclusive *regions* of two types: one where we use a continuum technique to describe vehicle movement and another where we use a discrete, agent-based technique for simulation. These regions are not necessarily connected nor static; we can pick either technique to govern a given part of the traffic network based on observations, application requirements, a user's field of view, each car's distance to visual display, the current volume and velocity of traffic in the network, or to enforce certain types of desired behaviors. Switching between the two simulation models can be dynamic and automatic based on specified governing criteria, similar to real-time graphical rendering using geometric levels of detail.

The key component of our hybrid simulation technique is how the two different types of simulations are coupled together; continuum traffic expects a density-like quantity of cars per car length with a velocity component, while discrete simulation is carried out with the explicit position and velocity of each vehicle in the network. We allow for an arbitrary number of interface points where the two types of simulation must be coupled and vehicles under one type of simulation must be transitioned into the other. This step is critical to ensure artifact-free, consistent transitions as vehicles cross boundaries between two simulation models.

Simulation components Our technique for describing the flow of traffic requires a road network with suitably-defined boundary conditions and an initial state for the vehicles in the network, which can also be taken directly from live traffic data. We then proceed by taking discrete time steps of varying length Δt, wherein the state of traffic is considered and integrated forward in time.

At a high level, the steps of our algorithm are as follows:

Step 1: Advance continuum regions:

(a) Determine the dynamics of traffic flow, also known as *flux*, between each adjacent cell.

(b) Compute Δt, the minimum stable timestep, using the maximum speed from each cell solution in Step 1a.

(c) Integrate each cell using the fluxes from Step 1a and Δt from Step 1b.

Step 2: Update *flux capacitors* (see Section 4.3.1) and add discrete cars as needed.

Step 3: Advance agent-based regions (using the same Δt computed in Step 1b).

Step 4: Aggregate all discrete vehicles (Section 4.1) that flow into a continuum region.

In other words, we separately advance the continuum and agent-based simulations and manage the transition of vehicles between the different simulation regimes. Below, we describe the basic simulation techniques used in Steps 1 and 3. Later, in Section 4, we present our main contributions — techniques for converting between different types of simulation.

3.3 Continuum simulation

In continuum models of traffic simulation, each lane is divided into discrete computational cells that represent the traffic. In the case of the Aw-Rascle-Zhang (ARZ) system of equations, these cells store the two conserved quantities in the vector $\mathbf{q} = [\rho, y]^{\mathrm{T}}$, where ρ is the density of traffic, i.e. "cars per car length", and y the "relative flow" of traffic. The solution is then advanced via explicit integration with the finite volume method (FVM); the cells themselves typically cover anywhere from a few car lengths of road to ten or more, depending on the details of the simulation.

The continuum components of the network are advanced in Step 1 before the agent-based simulation components are handled. This is to ensure that the Δt used in each component is the largest stable timestep achievable. The continuum simulation component has more stringent stability requirements than the agent-based component; by performing the continuum update prior to the agent-based, we can use the Δt computed in Step 1b in the later agent-based update performed in Step 3.

More detail on continuum simulation techniques for traffic simulation can be found in any of [Aw and Rascle 2000; Zhang 2002; Sewall et al. 2010]; the results presented in this paper use the ARZ model.

3.4 Agent-based simulation

The regions of our traffic network under the agent-based regime are handled by a discrete 'car-following' method. Each vehicle's position and velocity is explicitly tracked and advanced at each simulation step, and each chooses its acceleration based on the distance to the vehicle ahead of it as well as their difference in velocity. This acceleration is explicitly integrated into the vehicle's velocity, which is then integrated into position.

It should be noted that the details of the agent-based technique are not particularly relevant to our hybrid coupling and instantiation technique. As we show below, we only require that each discrete vehicle have a position and (non-negative) velocity and that it determine its velocity by the state of the vehicle ahead of it. Our demonstrations in this paper use an extended version of the method of [Treiber et al. 2000], considered to be a representative, modern approach to agent-based traffic simulation — our adaptation of their technique additionally supports lane-changing, inhomogeneous driver models, and vehicle response to traffic signals, intersections, and variable speed limits.

Lane changes are handled by first applying a behavior model that determines if vehicles *desire* to change lanes, then determining if such lane changes are safe, and finally transitioning the vehicle between the lanes over a time interval. Driver behavioral models vary across techniques; we have chosen a variety of parameters that we modulate over different agents to effect a variety of drivers and make a more realistic simulation.

4 Transitioning between continuum and agent-based models

To take advantage of agent-based and continuum simulations, we introduce a hybrid technique that allows a road network to be simulated with both simulation types; the network is partitioned into multiple disjoint (and not necessarily connected) *regions* that cover the domain. Each such region is governed by either agent-based simulation or continuum simulation.

These regions in our simulation are dynamic; we can adaptively change the shape of and the simulation method in a region as needed to observe certain phenomena, meet performance requirements, or to respond to user input. To achieve this, we must convert discrete vehicles from agent-based simulation lanes into the aggregate format necessary for continuum simulation, and we must use the distribution of density in continuum lanes to introduce discrete vehicles for agent-based simulation.

Sections 4.1 and 4.2 establish the fundamentals of the conversion process and describe how whole regions are converted from one regime to the other. Later, in Section 4.3, we present how traffic *flowing* from one type of region into another is handled.

4.1 Conversion of agent-based regions to continuum through averaging

Vehicle support functions It is straightforward to compute a continuum representation from a list of discrete vehicles; the $\mathbf{q}_k = [\rho_k, y_k]^{\mathrm{T}}$ stored for continuum simulation are averaged quantities for densities and relative flows of traffic. For each discrete vehicle C_i with front-bumper position p_i, length l_i, and velocity v_i, we define a boxcar-like support function as follows:

$$S_i(x) = H(x - p_i + l_i) - H(x - p_i) \tag{1}$$

where $H(x)$ is the Heaviside function. The support of all n vehicles is then given by the sum:

$$D(x) = \sum_{0}^{n-1} S_i(x) \tag{2}$$

An exemplary plot of this function is depicted in Figure 2. Assume, without loss of generality, that the continuum cells are uniformly

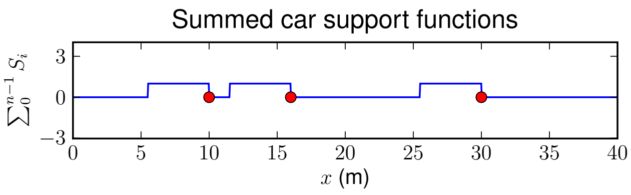


Figure 2: *The car support function $D(x)$ for a series of cars with front bumpers at $x = 10$, 16, and 30.*

spaced by Δx; then given n vehicles to discretize, the traffic density ρ_k for each cell of the continuum is

$$\rho_k = \frac{1}{\Delta x} \int_{k\Delta x}^{(k+1)\Delta x} D(x)\ dx \tag{3}$$

A weighted combination of such ρ_k along with the vehicles' velocities can be used to similarly determine traffic velocity, u_k and the derived 'relative velocity' y_k.

Averaging algorithm In practice, the $\mathbf{q}_k$ can be computed in an efficient manner — linear in the number of cars n — by iterating over each vehicle C_i, $i \in \mathbb{Z}[0, n)$. For each i, we compute the intersection of the nonzero portions of S_i with the continuum grid and apply Equation (3) and its analogues for velocity. So long as $\Delta x = \Omega(l_i)\,\forall l_i$, each vehicle will cover a constant number of grid cells and the averaging process is $O(n)$.

4.2 Conversion of continuum regions to agent-based through Poisson instantiation

The process of initializing an agent-based region given a continuum is more complicated than that described above. This is necessarily so; while agent-based to continuum conversion effects a *decrease* in information, the reverse requires us to *increase* the information in the system. We propose a method inspired by the kinetic theory of gases and Poisson processes that delivers suitable results while remaining simple and efficient.

4.2.1 Poisson processes

The Poisson process is a well-known stochastic procedure used to model occurrences ('arrival times') of independent events t_0, t_1, t_2, and so on. We use a Poisson-like process to determine the location of discrete vehicles in a continuum region given the piecewise-constant density cells ρ_k that comprise the unknowns in that region. Rather than determining how discrete *events* are distributed in *time*, we model where discrete *vehicles* are distributed in *space* — specifically, how they are located along the 1-dimensional space of the lane.

It can be shown [Devroye 1986] that the times between events t_i, t_{i+1} in a *homogeneous* Poisson process with rate λ satisfy an exponential distribution with probability density function (PDF):

$$p(x) = \lambda e^{-\lambda x} \tag{4}$$

More precisely, Equation (4) gives the probability, for each $x \in \mathbb{R}[0, \infty)$, that $x = t_{i+1} - t_i$.

We can efficiently generate exponential random variables through the *inversion* process; that is, we combine uniform random variables U with the inverted cumulative density function (CDF) of the exponential distribution; setting this equal to a uniform random variable U and solving for x, we get an exponentially-distributed random variable:

$$-\frac{\ln U}{\lambda} = x \tag{5}$$

4.2.2 Generating events in an inhomogeneous Poisson process

To properly account for the continuum traffic density data ρ_k in the lane, we would like to model an *inhomogeneous* Poisson process — that is, where the λ in Equation (4) is no longer constant. Indeed, we wish to have $\lambda(x) = \frac{1}{l}\rho(x)$, where l reflects a representative car length; this scaling converts $\rho(x)$ from cars per car length to cars per meter (i.e., to the spatial units of x).

To capture the variation in ρ that generally occurs in a continuum lane, the distribution of the separation between successive events is no longer described by Equation (4), but by the following PDF reflecting the inhomogeneous case:

$$p(x) = \lambda(\tau + x)\, e^{-(\Lambda(\tau+x) - \Lambda(\tau))} \tag{6}$$

Here τ is the time of the last event and $\Lambda(x) = \int_0^x \lambda(t)\, dt$.

Cumulative density function To extend the technique presented in Equation (5) for generating events in a homogeneous Poisson processes to the inhomogeneous case, we compute the CDF to obtain the following:

$$= 1 - e^{-(\Lambda(\tau+x) - \Lambda(\tau))} \tag{7}$$

Here we have assumed that $\lim_{x\to\infty} \Lambda(x) = \infty$.

Inversion As with the homogeneous case, we invert the CDF to achieve the formula for generating exponentially-distributed random variables that match the given $\lambda(x)$:

$$x = \Lambda^{-1}(\Lambda(\tau) - \ln U) - \tau \tag{8}$$

Recall that the above equation gives the separation time between two events with the first occurring at τ; in general, we will be more interested in the actual time of the new event rather than this difference:

$$\tau_i = \Lambda^{-1}(\Lambda(\tau_{i-1}) - \ln U) \tag{9}$$

Observe that we can generate events in an inhomogeneous Poisson process with rate function $\lambda(x)$ if we can compute $\Lambda(x)$ and $\Lambda^{-1}(\nu)$. For general $\lambda(x)$, this may require numerical methods for integration and inversion — computations that may be expensive and prone to issues of numerical stability. In these cases, the *thinning method* [Lewis and Shedler 1979] may be applied with success; this technique uses a secondary rate function $\mu(x) > \lambda(x)$ to which the above inversion process may be applied and then uses a rejection-like process to select only the random variables that satisfy the process with rate function $\lambda(x)$.

Because our rate function $\lambda(x) = \frac{1}{l}\rho(x)$ is actually given by discrete cells' traffic densities ρ_k, our rate function is piecewise-constant — its integral $\Lambda(x)$ and integral inverse $\Lambda^{-1}(\nu)$ are simple to compute. In Section 4.2.4, we give an efficient algorithm for computing the positions of discrete vehicles given a continuum region with a piecewise-constant density function $\rho(x) = \rho_k$; however, we first establish details of the integral and integral inverse of a piecewise-constant function.

4.2.3 Integral and inverted integral of continuum densities

We have a continuum region with n discrete cells $\mathbf{q}_k = [\rho_k, y_k]^{\mathrm{T}}, k \in \mathbb{Z}[0, n)$. Each cell is Δx in length; then

$$\begin{aligned}\Lambda(s) &= \int_0^s \frac{1}{l}\rho(t)\, \Delta x \\ &= \frac{1}{l}\Delta x \sum_{k=0}^{i-1} \rho_k + \frac{1}{l}\int_{i\Delta x}^{s} \rho(t)\, \Delta x \\ &= \frac{1}{l}\left(\Delta x \sum_{k=0}^{i-1} \rho_k + \rho_i (s - i\Delta x)\right) \qquad (10)\end{aligned}$$

where $i = \sup\{j \in \mathbb{Z}[0, n] | \Delta j < s\}$; i.e., the index of the cell 'containing' s (or one past the last grid cell, if $s \geq \Delta n$). We restrict $s \geq 0$ and define $\rho(t) = 0$ for $t \geq \Delta x n$, and also that $\rho_n = 0$. See Figure 3 for a plot of $\lambda(x)$ and $\Lambda(x)$. We wish to use Equation (9)

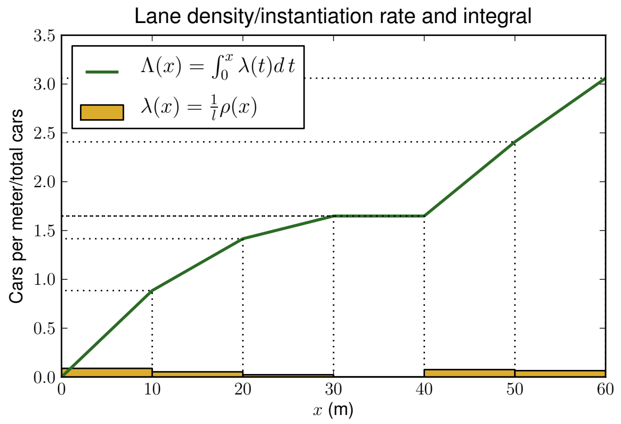


Figure 3: *Plot of $\frac{1}{l}\rho_k$ for a lane and its integral. Exponentially-distributed random variables are mapped to the y-axis and used to locate the x-value of an event (vehicle).*

to generate events that correspond to the density ρ in a continuum region, so we must invert Λ from Equation (10).

Asymptotic behavior Let us consider this $\Lambda(x)$; in Equation (7), we assumed that $\lim_{x\to\infty} \Lambda(x) = \infty$. This is important because the argument to Λ^{-1} in Equation (9) takes its value in the range $(0, \infty)$, so the domain of $\Lambda^{-1}(\nu)$ must match. However, for our $\Lambda(x)$ in Equation (10), $\lim_{x\to\infty} \Lambda(x) = \frac{1}{l}\Delta x \sum_{k=0}^{n-1} \rho_k < \infty$, because each continuum lane has finite length and obviously contains a finite number of vehicles. In our technique, when $\Lambda(\tau) - \ln U > \frac{1}{l}\Delta x \sum_{k=0}^{n-1} \rho_k$ in Equation (9)[1], we simply stop the instantiation process; this is the termination condition.

Monotonicity Finally, while the $\Lambda(x)$ in Equation (10) is monotone (because $\frac{1}{l}\rho_k \geq 0\,\forall k$), it is not *strictly* increasing and $\Lambda^{-1}(\nu)$ is not well defined in the traditional sense. However, given our application, we may easily deal with this issue. A 'flat' spot on $\Lambda(x)$ corresponds to one or more adjacent cells $i+0, i+1, \ldots, i+m$ ($i, m \geq 0$ and $i+m < n$) where $\rho_i = 0$; conceptually, *no vehicles may be instantiated here*. Whenever $\mathbb{X} = \left\{\Lambda^{-1}(\nu)\right\}$ for any ν has cardinality $|\mathbb{X}| > 1$, we define $x = \sup \mathbb{X} = \Delta x\,(i+m+1)$.

[1]The quantity $\Lambda(\tau) - \ln U$ is strictly increasing, and $\frac{1}{l}\Delta x \sum_{k=0}^{n-1} \rho_k$ is finite, so this must eventually occur for some τ

4.2.4 Discrete car instantiation algorithm

The process for generating discrete cars given n cells of density ρ_k, $k \in \mathbb{Z}[0, n)$ with spacing Δx is based on Equations (9) and (10); given a previous vehicle position p_{i-1} and a uniformly distributed random variable U, we add $-\ln U$ to the integrated rate $\Lambda(p_{i-1})$ and look up the x value of this sum in Λ^{-1}; this gives us p_i. When we generate a value that has no value in the inverse integral Λ^{-1}, we have exceeded the length of the region and we are finished. Our algorithm for this process is given in Algorithm 1; it is based

Algorithm 1 INSTANTIATE-VEHICLES

```
INSTANTIATE-VEHICLES($\rho[n]$, $\Delta x$)
     // $\rho[n]$ — an array of $n$ density values,
     // $\Delta x$ — the length of each grid cell
 1   $p = [\,]$
 2   $\Lambda_{\text{last}}, i, \sigma = 0$
 3   while true
 4       $U$ = UNIFORM-RANDOM-NUMBER($(0, 1]$)
 5       $\Lambda_{\text{cand}} = \Lambda_{\text{last}} - \ln U$
 6       $\sigma_{\text{cand}} = \sigma$
 7       while $i < n$ and $\sigma_{\text{cand}} + \frac{1}{l}\rho[i]\Delta x < \Lambda_{\text{cand}}$
 8           $\sigma_{\text{cand}} = \sigma_{\text{cand}} + \frac{1}{l}\rho[i]\Delta x$
 9           $i = i + 1$
10       $p_{\text{cand}} = \frac{(\Lambda_{\text{cand}} - \sigma_{\text{cand}})}{\frac{1}{l}\rho[i]} + i\Delta x$
11       if $p_{\text{cand}} > n\Delta x$
12           return $p$
13       if $p_{\text{cand}} + l > p[-1]$
14           $\Lambda_{\text{last}} = \Lambda_{\text{cand}}$
15           $\sigma = \sigma_{\text{cand}}$
16           $p = p + [p_{\text{cand}}]$
```

An algorithm for vehicle instantiation from continuum data

on several observations on the nature of Equations (9) and (10). First, we note that the τ in $\Lambda(\tau)$ in Equation (9) is the argument to Λ^{-1} from the previous event — $\Lambda\left(\Lambda^{-1}\left(\Lambda(\tau_{i-1}) - \ln U_{i-1}\right)\right) = \Lambda(\tau_{i-1}) - \ln U_{i-1}$. Therefore, we never need to explicitly compute $\Lambda(\tau)$; it was computed in the previous iteration. In the base case, we know from Equation (10) that $\Lambda(0) = 0$. While the process is expected to generate vehicles with spacing of at least l, it is possible that the vehicle identified by p_{cand} in line (10) overlaps the previously instantiated vehicle. To prevent this, we use rejection sampling (see lines 13–16).

Furthermore, we know that the instantiated vehicles have strictly increasing x values. This is a general property of Poisson processes, and it is readily confirmed by the fact that $-\ln U > 0$ and that $\Lambda(\tau)$ is monotone and nondecreasing. This simplifies the computation of each Λ^{-1}; we know that each Λ_{last} will take its value as $i\Delta x < \Lambda^{-1}(\Lambda_{\text{last}})$, where i is the index of the grid cell the last instantiation occurred in. Figure 4 shows the result of running INSTANTIATE-VEHICLES on a continuum lane.

Analysis The performance of INSTANTIATE-VEHICLES is $O(n+k)$, where n is the number of grid cells in the continuum region and k is the number of instantiated vehicles. The outer while loop spanning lines 3–16 in Algorithm 1 is executed k times, and the inner loop spanning lines 7–9 will iterate no more than n times in the course of the entire execution of INSTANTIATE-VEHICLES.

It remains to bound the value of k for a call of INSTANTIATE-VEHICLES; this is a randomized algorithm and k may theoretically be arbitrarily large (although subject to some upper bound based on floating-point arithmetic). We can give a conservative estimate as follows: consider the case where

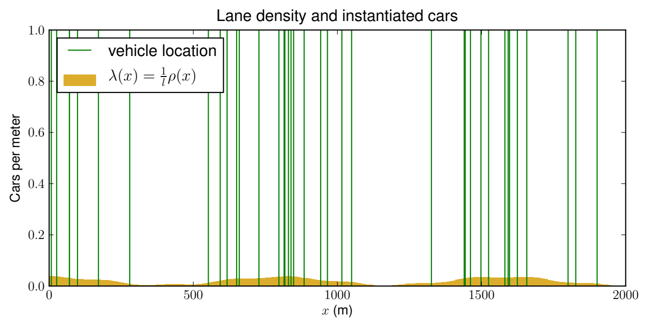


Figure 4: *The results of running* INSTANTIATE-VEHICLES() *on a continuum lane; green vertical lines represent vehicle positions.*

$\rho[n]$ is such that $\rho[i] = 1 \forall i \in \mathbb{Z}[0, n)$ — clearly an upper bound on any real continuum region. Then $\lambda(\tau) = \frac{1}{l}$, and since the expected value for a homogeneous exponential distribution with rate parameter λ is $\frac{1}{\lambda}$, we average an instantiated vehicle every l meters — bumper-to-bumper traffic that precisely matches saturation of density. The total expected number of vehicles k is then $\frac{n\Delta x}{l}$, which itself is $O(n)$. Given that the estimate for k we have just developed is an upper bound, the expected runtime of INSTANTIATE-VEHICLES is $O(n)$.

4.3 Coupling

The continuum and agent-based simulations that occur on adjoining regions of a road network interact in two ways: vehicles passing from one regime to another must be converted to the representation used in the destination regime (akin to the processes described in Sections 4.1 and 4.2), and the flow of traffic in each lane must influence the lanes that precede them.

We introduce 'flux capacitors' to convert continuum flow to discrete agents in Section 4.3.1, and we describe how we adapt the car averaging procedure from Section 4.1 to handle discrete vehicles that flow into continuum regions in Section 4.3.2.

4.3.1 Flux capacitors

Given two adjacent cells in a continuum lane, we use methods described in [Aw and Rascle 2000; Zhang 2002; Sewall et al. 2010] to solve for the flow at the interface between two cells; when we do so, we have computed the *flux* between the cells. When a continuum lane flows into a discrete one, we first create a 'virtual' continuum cell at the start of the agent-based lane using averaging (see Section 4.1). Then we are able to use the standard flux computation process on the last continuum cell and this virtual one; this flux can now be used to convert density flowing out of the continuum region into discrete vehicles entering the agent-based lane and simultaneously provide the proper dynamics for the incoming continuum lane.

To instantiate vehicles due to this flux, we accumulate density until a sufficient quantity has been retained that we may emit a vehicle — we call this a *flux capacitor*. Formally, the accumulated density ρ_{cap} at a flux capacitor increases by $\Delta\rho_{\text{cap}}$ during the time step from t_{i-1} to t_i as per the following:

$$\Delta\rho_{\text{cap}} = \frac{1}{l}\int_{t_i}^{t_{i+1}} \rho_0(t)u_0(t)\,dt \tag{11}$$

where $\rho_0(t)u_0(t)$ is the ρ-component of the flux of the intermediate state at time t, as computed by the solution of a continuum lane, and l is vehicle length; this term is necessary to scale the integrand from cars per car length (ρ) times meters per second (u) to cars per second, or rate of traffic flow. Because we consider the intermediate state $q_0(t) = \rho_0(t)u_0(t)$ to be constant during a timestep, this integral is trivial to evaluate.

4.3.2 Flow averaging

The above discussion on 'flux capacitors' describes how to translate flow from a continuum region into discrete vehicles, and how to account for the downstream (agent-based) region's effect on the upstream continuum region. Here we consider the converse: how vehicles in discrete regions that flow *into* continuum regions may be converted into the appropriate continuum quantities, and how the state of this continuum region can be accounted for in the upstream (agent-based) region.

Transfer of discrete vehicles into continuum regions When agent-based regions flow into continuum regions, one must convert discrete vehicles into continuum data as they pass into the new regime. We achieve this in a manner similar to our method for accounting for discrete downstream vehicles' effect on the outflow of continuum regions (see Section 4.3.1).

A single 'virtual' grid cell at the end of the agent-based region is filled according to the averaging procedure in Sec 4.1; this is then used as the upstream boundary condition for the downstream lane's (continuum) flux computation.

When the motion of a vehicle carries it from a discrete region to a continuum one, its discrete representation simply vanishes; the vehicle is immediately accounted for in the continuum representation and flux computation just described.

Finding leaders in continuum regions In agent-based simulation, a vehicle's motion is determined by the position and velocity of the vehicle directly ahead of it — its *leading vehicle*. When entering a downstream continuum region, a vehicle will have no such leader; we must translate the continuum quantities into a suitable position and velocity pair — a 'virtual' leading vehicle.

We use the vehicle instantiation procedure described in Section 4.2, except that it now terminates after finding just one vehicle; this vehicle's velocity is determined by the continuum data at its location.

4.4 Region refinement criteria

Continuum techniques can efficiently handle both large and dense areas of networks at the expense of being coarse-grained and not precisely capturing individual vehicle behaviors, whereas agent-based simulators are flexible and generally more computationally costly than continuum techniques.

There are numerous criteria to consider when choosing which portions of the network should be simulated with continuum or agent-based techniques:

Visual and spatial For real-time visualization, we consider what areas of the network are visible (via a view-frustum test or a more conservative bound) and ensure that only agent-based simulation is performed there — except for [Sewall et al. 2010], continuum techniques do not admit a ready visual representation.

Performance Based on the performance needs of the simulation, regions that are particularly expensive to compute in one regime can be converted to the other. More formally, every region can be assigned an estimated computational cost of the form:

$$w_T(r) = \alpha_T |r| + \beta_T C^r \tag{12}$$

where r is a region, $|r|$ the region's length, and C^r the number of vehicles in that region. α_T and β_T are weights associated with the computational regimes $T = \{\text{continuum}, \text{agent-based}\}$. These can be determined empirically during computation; based on what we know of the two regimes, at least $\alpha_{\text{continuum}} \gg \beta_{\text{agent-based}}$ and $\beta_{\text{agent-based}} \gg \alpha_{\text{continuum}}$. Given a pair of weights $w_{\text{continuum}}(r)$ and $w_{\text{agent-based}}(r)$ for each r and a computational 'budget', we can assign and convert regions using an inexpensive partitioning scheme, such as a greedy algorithm or polynomial approximation to the makespan problem (for example, as in [Hochbaum and Shmoys 1987]).

Feature-capturing Another useful class of criteria for partitioning arises from the desire to capture specific features and types of flow with a specific simulation technique. An example of this is when one wishes to track the movement of a specific vehicle or group of vehicles through the network. Because the continuum regime aggregates vehicles, were a specific vehicle to be absorbed into the continuum regime, either through the region conversion process described in Section 4.1 or the transfer process described in Section 4.3.2, the specific information associated with that vehicle would be lost. Similarly, should we wish to capture the behavior of heterogeneous vehicles, we must resort to the agent-based model, which allows varied vehicle behavior.

To effect this, any regions containing such features must not be converted, and furthermore, whenever such a feature of interest is about to transition into a region governed by the other (unsuitable) regime, that downstream region must be converted to the upstream regime.

Additionally, there are times when we wish to satisfy certain numerical conditions governing the nature of the simulation itself. For example, in the kinetic theory of gases, there is a dimensionless quantity known as the *Knudsen number* that is used to classify fluid behavior as a function of the fluid state itself and the scale of observation. Precisely, the Knudsen number is given by

$$Kn = \frac{\lambda}{L} \tag{13}$$

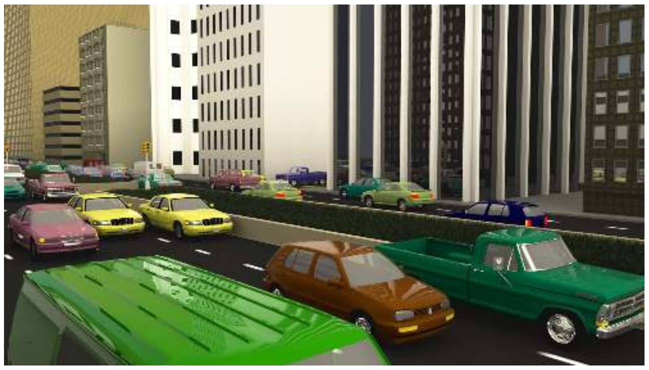
where λ is the mean free path of a particle and L the characteristic length scale of the problem. The prevailing wisdom in gas simulation is that continuum models are valid in the range $Kn < 0.01$, statistical models are valid when $Kn < 0.1$, and discrete models are valid at all scales; see [Hirschfelder et al. 1964] for details.

A rigorous investigation of the applicability of the Knudsen number to continuum traffic simulation has not been performed, but should the user wish to ensure that the above scheme is satisfied, it is straightforward to compute the Knudsen number for each lane and force those that have too large a Knudsen number to use the agent-based regime.

5 Results

5.1 Benchmarks

We have implemented our hybrid technique for interactive visual simulation of large-scale traffic and demonstrate the following scenarios:


(a) *A low-angle view*

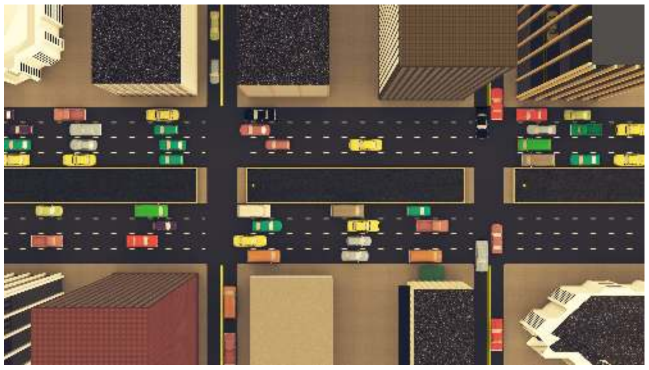

(b) *A top-down view*

Figure 5: *A city scene filled with traffic simulated with our technique*

Virtual downtown of a metropolitan area We have recreated a 'virtual downtown' based on the "The Grid" sequence in Godfrey Reggio's film *Koyaanisqatsi* [Reggio 1982]; a particular shot in this film shows traffic moving along a busy city street, played back at a greatly increased speed. The motion of the vehicles is punctuated by the rhythmic cycling of traffic signals and cross-traffic emerging from behind the skyscrapers evenly spaced at each block. This particular shot is located 52 minutes, 3 seconds into the theatrical release of the film[2]. The supplemental video accompanying this paper has video of similar traffic generated by our technique; a still can be seen in Figure 1.

Augmented satellite street maps One impetus for this work was to be able to interactively visualize real-world traffic simulated using live traffic data to augment online virtual worlds, such as Google Earth, Microsoft Virtual Earth, or Second Life, as well as to enhance mobile geographic information systems, such as car navigation systems for PDAs or on-vehicle GPS systems.

In Figure 1, we show example road networks used as simulation domains for real-time visual simulations of metropolitan-scale traffic — with tens of thousands of vehicles. The models illustrated here were created using GIS data from the Open Street Maps project using the technique described in [Wilkie et al. 2011]. Our hybrid technique is able to recreate real-time traffic flows and individual vehicle motion on complex, metropolitan-scale road networks, which can then be visualized atop aerial imagery.

[2]For readers in most regions, this can be viewed on YouTube: `http://www.youtube.com/watch?v=Sps6C9u7ras#t=0h52m03s`

Figures 6(a)–(d) and the corresponding sequence in the accompanying video show dynamic region refinement based on the movement of a 'region of interest' — a yellow rectangle. Typically, this region of interest would be the visible portion of the scene, allowing our technique to perform simulation in the off-camera areas without incurring the expense of agent-based simulation everywhere.

Although this concept of augmented street maps bears close resemblance to some features in the work by [Kim et al. 2009], their approach to augment aerial earth maps with traffic information requires setting up many video cameras closely located on freeways in order to reproduce the spatial extent and aerial coverage of traffic visualization that we are able to recreate here. With our approach, commonly available live traffic data from sparsely located cameras or merely inexpensive in-road sensors (e.g. inductive loops) would be sufficient to initialize our simulation method for real-time traffic visualization. The two approaches are, however, complimentary; our work could be easily integrated into their overall Augmented Reality framework.

5.2 Performance

One of key objectives for this work is to facilitate the cooperative use of disparate simulation strategies — agent-based and continuum traffic simulation — in a traffic network for extensive metropolitan areas. There are numerous reasons this is desirable: continuum techniques have performance advantages over agent-based simulations in many situations; their computational cost is proportional to size of the network, not to the traffic therein. Furthermore, the limited and regular memory access patterns of continuum algorithms are much more amenable to scalable parallelism than those found in agent-based algorithms.

Figure 7 shows the performance of (single-thread) agent-based, continuum, and hybrid simulations on a city road network for a variety of vehicle densities. Pure continuum simulation outperforms pure agent-based by **10–20x**. Our hybrid scheme, in which a constant portion the road network uses agent-based simulation and the rest uses continuum-based, is **9–17x** faster than pure agent-based. These results were collected on an Intel® Core$^{\text{TM}}$ i7 980X processor running at 3.33GHz.

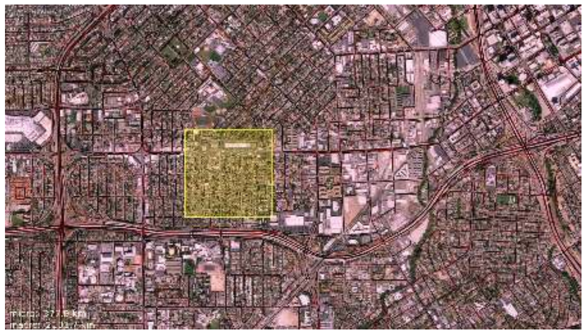
5.3 Validation using real-world data

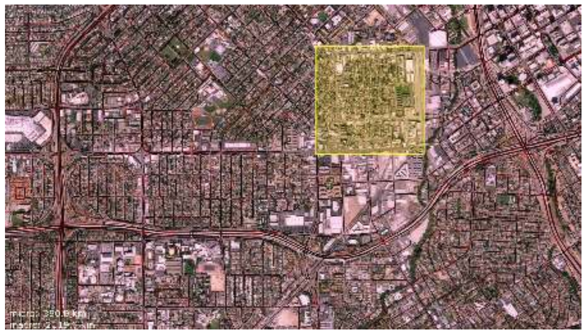
It is useful to understand how the traffic motion produced by the method described in this work compares to the motion of real-world traffic. However, this comparison must be performed with deliberation and care, as there are numerous subtleties at play. For more detail on the validation experiments, data formats and sources, issues involved, methodology, comparison strategies considered, and evaluation on the effectiveness of our hybrid approach, please refer to Appendix B. Below we summarize the results from our validation experiments using the data from the Next-Generation Simulation (NGSIM) project by the Federal Highway Administration.

5.3.1 Comparison results

Agent-Based Overall, the agent-based simulation matches the NGSIM data quite well; near the start of the simulation region, we expect all quantities to match because we are close to the incoming boundary condition. 'Detectors' further down the highway continue to match the vehicle count well, but it is evident that the agent-based simulation technique which we are using aggressively adjusts vehicles to their preferred velocity. The final detector ($x = 620$; Figure 8) matches velocity well as we approach the boundary condition, where the outgoing velocity is set based on the leading vehicle from the

(a)


(b)


(c)

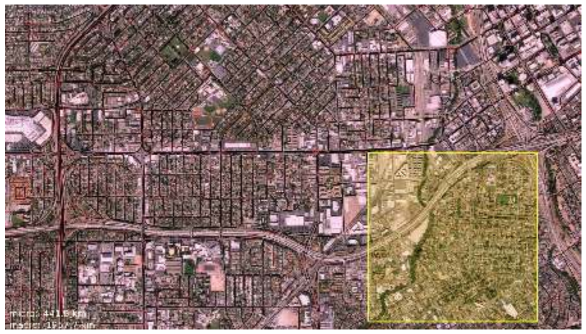

(d)

Figure 6: *A sequence of images illustrating simulation region refinement. 6a: Initially, the whole road network is simulated with agent-based techniques. 6b-6d: Later, only roads whose bounding box intersects the yellow box are simulated with agent-based techniques — continuum techniques are used elsewhere. The averaging and instantiation methods of Sections 4.1 and 4.2 handle changes due to the movement of the rectangle, while the coupling techniques described in Section 4.3 seamlessly integrate the dynamics of different simulation regimes.*

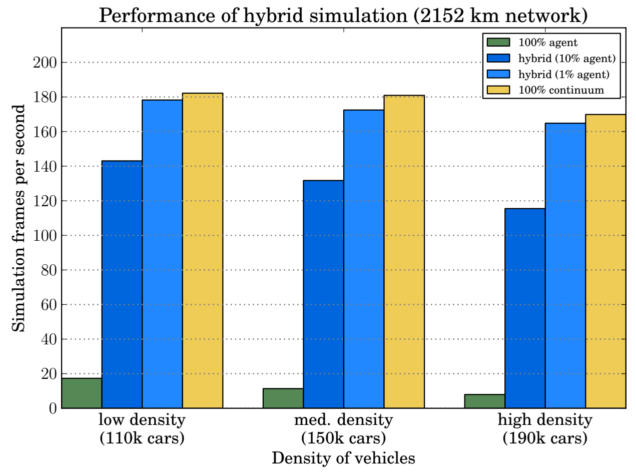


Figure 7: *Performance of all-continuum simulation, our hybrid technique, and wholly agent-based simulation for various densities on a road network with 2151 km of lanes*

NGSIM data. We see some time shifts in features we identify from the NGSIM data which can be attributed to the different preceding velocity values.

Continuum The results of the continuum simulation are fluxes. As with agent-based, we see good matches for the initial detectors and drift accumulating further down the highway. However, despite shifts in overall magnitude and phase, the overall patterns remain, showing that the primary features of the flow are preserved — in particular, the trough in flux that occurs around $t = 800$ is preserved.

Hybrid Because the final leg of the hybrid validation scheme is agent-based, we have data for the number of cars, velocity, and flux for the final detector ($x = 620$, Figure 8). Velocity matches quite well as with the pure agent-based case due to the outgoing boundary condition. The vehicle counts demonstrate the same features as those found in the original NGSIM data, albeit with slight variations in magnitude.

Sequence comparison Computing the number of vehicles and average velocities for the real-world data and corresponding simulation results in comparable time-series data — numbers of cars, average velocity, and flux. Examining these visually can be illuminating, but a quantitative comparison of these data provides a more rigorous measure. A standard infinity– or 2– norm might seem an obvious choice, but these will rather harshly penalize noisy and time-shifted data. As discussed in [Chen et al. 2005] and [Morse and Patel 2007], suitably modified string-distance metrics, such as *longest common subsequence* and *edit distance*, are very useful when comparing time series.

As the name implies, given two sequences R and S — not necessarily of the same length — *longest common subsequence* (LCSS) reports the length of the largest subsequence they share. By normalizing this subsequence length by the length of the shorter of R and S, we can assign a 'score', or distance, to the similarity of the two sequences. Numerous techniques fit into the aegis of *edit distance*; the common theme is scoring the similarity of R and S by the 'effort' required to transform one to the other. Varieties of edit distance are distinguished by how they define the weight of transformations — [Chen et al. 2005] propose *edit distance with real penalties* (EDR).

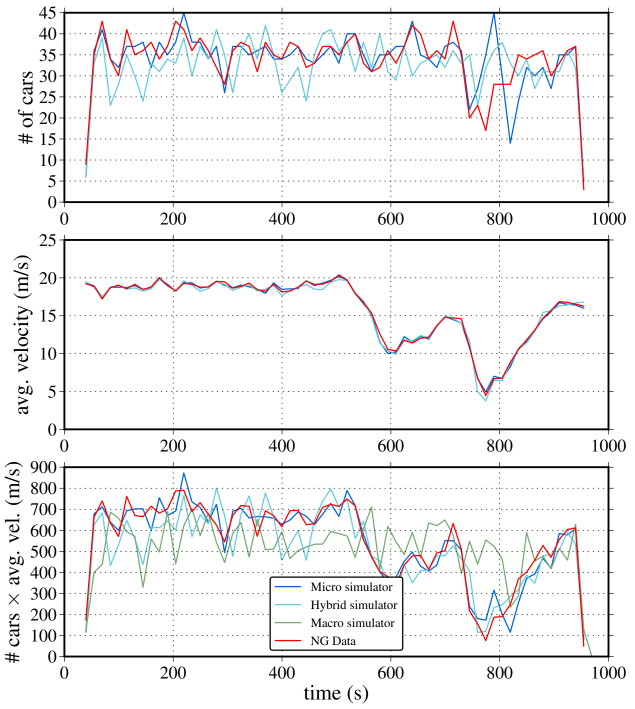


Figure 8: *Comparison between agent-based (micro) simulation, continuum (macro) simulation, our hybrid simulation technique, and real-world NGSIM data for the highway 101 domain. These graphs show density, velocity, and flux recorded over 15-second intervals centered around the times shown at a sensor near the end of the highway (620m from the start).*

These algorithms typically operate on sequences consisting of elements from a finite set — for example, Latin characters in a string. When dealing with sequences of real numbers (velocity and flux in this case), [Morse and Patel 2007] suggest identifying two real numbers $x \in R$ and $y \in S$ as 'equal' when $|x - y| < \epsilon = \sigma_{\min}/2$, where $\sigma_{\min}$ is the lesser of the standard deviations of R and S.

We propose to adopt and modify string-distance metrics, such as LCSS and EDR, for comparing different simulation methods and validating simulation results. For each of the simulation types (agent-based, continuum, hybrid), we have compared the resulting flux time series with the corresponding NGSIM data — see Table 1. As the data in the $x = 620$ row shows, our simulation technique only slightly decreases the score for agent-based simulation while maintaining overall performance comparable to that of the continuum methods.

6 Conclusion

We have presented a novel method to dynamically couple continuum and discrete methods for interactive simulation of large-scale vehicle traffic for virtual worlds and augmented aerial maps. Adopting these two disparate techniques simultaneously in different regions allows for a flexible simulation framework where the user can easily and automatically trade off quality and efficiency at runtime. We have applied this technique to the simulation of large networks of car traffic based on real-world data, as well as synthetic urban settings, and achieved greater-than real-time performance.

x	metric	agent-based	continuum	hybrid
429	flux LCSS	0.833	0.850	
	flux EDR	0.900	0.870	
440	flux LCSS	0.783	0.850	
	flux EDR	0.858	0.862	
474	flux LCSS	0.533	0.800	
	flux EDR	0.675	0.813	
497	flux LCSS	0.533	0.800	
	flux EDR	0.700	0.813	
542	flux LCSS	0.475	0.672	
	flux EDR	0.680	0.750	
598	flux LCSS	0.836	0.607	
	flux EDR	0.877	0.710	
620	flux LCSS	0.934	0.541	0.820
	flux EDR	0.951	0.685	0.861

Table 1: *Flux sequence comparison results for successive detection points along the NGSIM 101 freeway. The final row shows that our hybrid technique gives only a modest drop in comparison score with the real-world data.*

6.1 Extension to crowds and other phenomena

While we have only investigated how our coupling technique may be applied to two models for simulating and visualizing traffic flows, there are potential applications to other areas of simulation.

Other varieties of multi-agent simulation are particularly attractive candidates for our hybrid technique. For hybrid crowd simulation, there several important components that need to be developed — first, an agent-free continuum simulation technique for crowds is needed. This does not yet exist — the work of [Narain et al. 2009] does solve for velocity in a continuum fashion, but their technique uses discrete agents to provide goals. For coupling and instantiation, a key component is a suitable statistical model expressing distribution of discrete elements in the continuum model. The fact that the domain is a 2D rather than a 1D one is not necessarily problematic; work in gas kinetics has dealt extensively with reconciling particle and statistical models in multiple dimensions.

One issue that will require careful consideration is the stability of transitions; while the traffic in a road in this method always moves in a single direction — forward along the road — people in crowds are free to move in many directions, which may include backtracking. This freedom in locomotion presents an issue wherein agents can become 'jittered' between continuum and agent regions; forcing regions to overlap by some amount and only performing transitions where the overlap area meets a single region may be a reasonable way to handle this situation.

6.2 Limitations

While we have emphasized the flexibility of this technique, it has some limitations. In particular, because of the statistical nature of the instantiation technique, quickly alternating the simulation type of an area will likely result in incoherent vehicles — quantities and distributions will be relatively stable, but actual positions will not.

Furthermore, this same instantiation stochasticity can result in irregular leading distances and velocities when computing accelerations for the leading vehicle for agent-based regions flowing into continuum ones. For the agent-based models we have considered, this has not been an issue, but for certain unstable acceleration computations, it could require some further work.

This technique currently will perform well with a number of constituent simulation types, continuum and agent-based, so long as the continuum method can report density and velocity wherever it is required and the agent-based vehicles have query-able positions and velocities and compute their accelerations based on their leading vehicle. This sort of 'car-following' model is by far the most prevalent, but it is conceivable that other types of models are desirable; some modifications to the scheme presented here will be necessary to accommodate them.

6.3 Future Work

This approach opens many other promising avenues for future work. It would be an interesting exploration to use our technique for real-time traffic prediction to examine how the behaviors of individual vehicles at the arterial street level affect overall traffic flow on freeways. This could be used to calculate better routing for individual vehicles and more effective remediation for traffic congestion.

Furthermore, preliminary results have shown that aspects of the hybrid technique are highly parallel; we hope to develop our technique to effectively utilize today's many-core parallel architectures and tomorrow's nascent parallel hardware.

Acknowledgment

This research is supported in part by Army Research Office, Intel Corporation, National Science Foundation, and Carolina Development Foundation.

References

ALGERS, S., BERNAUER, E., BOERO, M., BREHERET, L., TARANTO, C. D., DOUGHERTY, M., FOX, K., AND GABARD, J. F. 1997. SMARTEST project: Review of micro-simulation models. *EU project No: RO-97-SC 1059.*

AW, A., AND RASCLE, M. 2000. Resurrection of "second order" models of traffic flow. *SIAM journal on applied mathematics 60*, 916–938.

BURGGRAF, O. 1966. Analytical and numerical studies of the structure of steady separated flows. *J. of Fluid Mechanics 24*, 01, 113–151.

CHEN, L., ÖZSU, M., AND ORIA, V. 2005. Robust and fast similarity search for moving object trajectories. In *SIGMOD*, ACM, 491–502.

CHEN, G., ESCH, G., WONKA, P., MUELLER, P., AND ZHANG, E. 2008. Interactive procedural street modeling. In *SIGGRAPH 2008*, ACM, New York, NY, USA.

CREMER, J., KEARNEY, J., AND WILLEMSEN, P. 1997. Directable behavior models for virtual driving scenarios. *Trans. Soc. Comput. Simul. Int. 14*, 2, 87–96.

DAGANZO, C. 1995. Requiem for second-order fluid approximations of traffic flow. *Trans. Research Part B 29*, 4, 277–286.

DEVROYE, L. 1986. *Non-Uniform Random Variate Generatiom.* Springer-Verlag.

DONIKIAN, S., MOREAU, G., AND THOMAS, G. 1999. Multimodal driving simulation in realistic urban environments. *Progress in System and Robot Analysis and Control Design (LNCIS) 243*, 321–332.

GALIN, E., PEYTAVIE, A., MARÉCHAL, N., AND GUÉRIN, E. 2010. Procedural generation of roads. In *Eurographics 2010.*

GERLOUGH, D. L. 1955. *Simulation of freeway traffic on a general-purpose discrete variable computer*. PhD thesis, UCLA.

GO, J., VU, T., AND KUFFNER, J. 2005. Autonomous behaviors for interactive vehicle animations. In *Intl. J. of Graphical Models*.

GUY, S., CHHUGANI, J., CURTIS, S., DUBEY, P., LIN, M. C., AND MANOCHA, D. 2010. PLEdestrians: A Least-Effort Approach to Crowd Simulation. In *EG/ACM SCA*.

HELBING, D. 2001. Traffic and related self-driven many-particle systems. *Reviews of Modern Physics 73*, 4, 1067–1141.

HIRSCHFELDER, J. O., CURTISS, C. F., AND BIRD, R. B. 1964. *The Molecular Theory of Gases and Liquids*, revised edition ed. Wiley-Interscience.

HOCHBAUM, D. S., AND SHMOYS, D. B. 1987. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM 34*, 1 (January), 144–162.

KIM, K., OH, S., LEE, J., AND ESSA, I. 2009. Augmenting aerial earth maps with dynamic information. In *IEEE International Symposium on Mixed and Augmented Reality*.

LEBACQUE, J., MAMMAR, S., AND HAJ-SALEM, H. 2007. The Aw–Rascle and Zhangs model: Vacuum problems, existence and regularity of the solutions of the Riemann problem. *Trans. Research Part B 41*, 7, 710–721.

LEWIS, P. A. W., AND SHEDLER, G. S. 1979. Simulation of nonhomogeneous poisson processes by thinning. *Naval Research Logistics Quaterly 26*, 403–413.

LIGHTHILL, M. J., AND WHITHAM, G. B. 1955. On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A229*, 1178 (May), 317–345.

2011. MITSIM. MIT Intelligent Transportation Systems.

MORSE, M., AND PATEL, J. 2007. An efficient and accurate method for evaluating time series similarity. In *ACM SIGMOD*, ACM, 569–580.

NAGEL, K., AND SCHRECKENBERG, M. 1992. A cellular automaton model for freeway traffic. *Journal de Physique I 2*, 12 (December), 2221–2229.

NARAIN, R., GOLAS, A., CURTIS, S., AND LIN, M. C. 2009. Aggregate dynamics for dense crowd simulation. *ACM SIGGRAPH Asia*.

NEWELL, G. 1961. Nonlinear effects in the dynamics of car following. *Operations Research 9*, 2, 209–229.

PAUSCH, R., CREA, T., AND CONWAY, M. 1992. A literature survey for virtual environments - military flight simulator visual systems and simulator sickness. *Presence: Teleoperators and Virtual Environments 1*, 3, 344–363.

PAYNE, H. J. 1971. Models of freeway traffic and control. *Mathematical Models of Public Systems 1*, 51–60. Part of the Simulation Councils Proceeding Series.

PELECHANO, N., ALLBECK, J. M., AND BADLER, N. I. 2008. *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers.

PETTRÉ, J., KALLMANN, M., AND LIN, M. C. 2008. Motion planning and autonomy for virtual humans. In *ACM SIGGRAPH 2008 classes*, 1–31.

REGGIO, G., 1982. Koyaanisqatsi. Film, Oct. MGM Studios.

REYNOLDS, C. 1987. Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH*, ACM New York, NY, USA, 25–34.

RICHARDS, P. I. 1956. Shock waves on the highway. *Operations Research 4*, 1, 42–51.

SEWALL, J., WILKIE, D., MERRELL, P., AND LIN, M. C. 2010. Continuum traffic simulation. In *Eurographics 2010*.

SEWALL, J., VAN DEN BERG, J., LIN, M. C., AND MANOCHA, D. 2011. Virtualized traffic: Reconstructing traffic flows from discrete spatiotemporal data. *IEEE TVCG 17*, 26–37. doi = `http://doi.ieeecomputersociety.org/10.1109/TVCG.2010.27`.

2009. SUMO — Simulation of Urban MObility, October.

TREIBER, M., HENNECKE, A., AND HELBING, D. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E 62*, 2, 1805–1824.

TREUILLE, A., COOPER, S., AND POPOVIĆ, Z. 2006. Continuum crowds. In *SIGGRAPH*, ACM New York, NY, USA, 1160–1168.

WANG, H., KEARNEY, J., CREMER, J., AND WILLEMSEN, P. 2005. Steering behaviors for autonomous vehicles in virtual environments. In *Proc. IEEE Virtual Reality Conf.*, 155–162.

WHITHAM, G. B. 1974. *Linear and nonlinear waves*. John Wiley and Sons, New York, New York.

WILKIE, D., SEWALL, J., AND LIN, M. C. 2011. Transforming gis data into functional road models for large-scale traffic simulation. *IEEE TVCG*. doi = `http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.116`.

ZHANG, H. 2002. A non-equilibrium traffic model devoid of gas-like behavior. *Trans. Research Part B 36*, 3, 275–290.

A Arc Roads

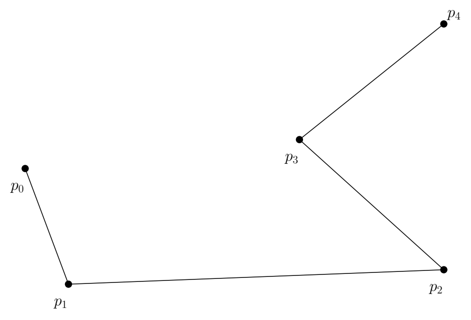
A.1 Preliminaries

We have an ordered sequence P of n points:

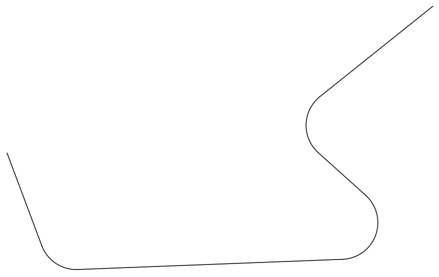
$$P := (p_0, p_1, \ldots, p_{n-2}, p_{n-1}) \tag{14}$$

These points define a (not necessary planar) polyline with $n-1$ segments such as that in Figure 9a. Let us assume that there are no two points adjacent in the sequence that are equal, and that there are no three adjacent points that are colinear; clearly we can eliminate these adjacent repeated points or the interior points in a colinear sequence without modifying the line's shape. We wish to 'smooth' this polyline to something like what is shown in Figure 9b, which we shall refer to as P_S. We construct P_S by replacing the region around each interior point $p_i, i \in \mathbb{Z}\,[1, n-2]$ of P with a circular arc and retaining the exterior points p_0 and p_{n-1}.

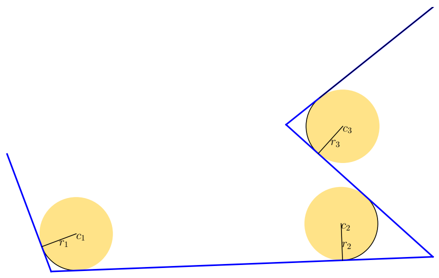
Each of these circular arcs can be characterized by a center c_i, radius r_i, orientation $\mathbf{o}_i$, start radius direction $\mathbf{s}_i$, and angle ϕ_i; see Figure 10.



(a) *A polyline P*



(b) *P_S: A 'smoothed' version of the polyline P*



(c) *The polyline P and the circles defining P_S*

Figure 9: *Polylines*

A.2 Construction of arc roads from polylines

A.2.1 Arc formulation

As mentioned above, each arc is defined by a center c_i, a radius r_i, orientation $\mathbf{o}_i$, start radius vector $\mathbf{s}_i$, and angle ϕ_i. Each arc i corresponds to an interior point p_i, and we require it to be tangent to $\overline{p_{i-1}p_i}$ and $\overline{p_ip_{i+1}}$. See Figure 9c and Figure 10; the full circles corresponding to each arc are shown.

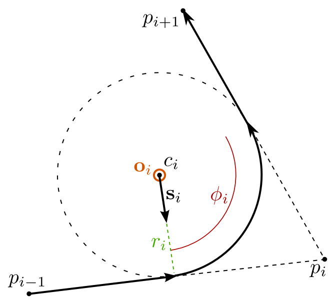


Figure 10: *Quantities defining an arc i corresponding to interior point p_i; the orientation vector $\mathbf{o}_i$ is coming out of the page.*

To help describe each arc i, we introduce the following quantities derived from the polyline P:

$$\mathbf{v}_i = p_{i+1} - p_i \tag{15}$$

their lengths:

$$L_i = |\mathbf{v}_i| \tag{16}$$

and the associated unit vectors:

$$\mathbf{n}_i = \frac{\mathbf{v}_i}{L_i} = \frac{\mathbf{v}_i}{|\mathbf{v}_i|} \tag{17}$$

We shall frequently refer to $-\mathbf{n}_{i-1} = \frac{p_{i-1}-p_i}{|p_{i-1}-p_i|}$. We also refer to the normal of the plane containing the circle:

$$\mathbf{o}_i = -\mathbf{n}_{i-1} \times \mathbf{n}_i \tag{18}$$

At certain times, it is useful to construct a matrix $\mathbf{F}_i$ that is the frame defined by $\mathbf{n}_i$, $\mathbf{s}_i$, and $\mathbf{o}_i$:

$$\mathbf{F}_i = \begin{bmatrix} \mathbf{n}_i & \mathbf{s}_i & \mathbf{o}_i \end{bmatrix} \tag{19}$$

The matrix $\mathbf{H}_i$ representing the homogeneous transform of translation to the arc center c_i along with the frame is defined as follows:

$$\mathbf{H}_i = \begin{bmatrix} \mathbf{n}_i & \mathbf{s}_i & \mathbf{o}_i & c_i \\ & 0 & & 1 \end{bmatrix} \tag{20}$$

Tangent points The projections of $c_i - p_i$ onto $-\mathbf{n}_{i-1}$ and onto $\mathbf{n}_i$ have equal length α_i; then the tangent points of the circle on $-\mathbf{v}_{i-1}$ and $\mathbf{v}_i$ are:

$$(c_i - p_i) \ \text{proj} \ -\mathbf{n}_{i-1} + c_i = -\alpha_i\mathbf{n}_{i-1} + c_i \tag{21}$$
$$(c_i - p_i) \ \text{proj} \ \mathbf{n}_i + c_i = \alpha_i\mathbf{n}_i + c_i \tag{22}$$

Radius vectors We are also interested in the (negative) radius vectors from these points to the center c_i:

$$\begin{aligned}\mathbf{r}_i^- &= -r_i\mathbf{s}_i = c_i - (-\alpha\mathbf{n}_{i-1} + p_i) = c_i + (\alpha\mathbf{n}_{i-1} - p_i) \\ &= r_i\mathbf{n}_{i-1} \times \mathbf{o}_i\end{aligned} \tag{23}$$

$$\mathbf{r}_i^+ = c_i - (\alpha\mathbf{n}_i + p_i) = r_i\mathbf{n}_i \times \mathbf{o}_i \tag{24}$$

Obviously, since $\mathbf{n}_{i-1}$ and $\mathbf{n}_i$ are perpendicular to $\mathbf{o}_i$ (see Equation (18)), $|\mathbf{r}_i^-| = |\mathbf{r}_i^+| = r_i$.

The center The center c_i can be determined by combining Equations (23), (24) and (21), (22):

$$c_i = p_i - \alpha_i \mathbf{n}_{i-1} + \mathbf{r}_i^- \tag{25}$$

$$= p_i + \alpha_i \mathbf{n}_i + \mathbf{r}_i^+ \tag{26}$$

We can also write c_i in terms of the unit bisector $\mathbf{b}_i$:

$$c_i = p_i + \sqrt{r_i^2 + \alpha_i^2}\,\mathbf{b}_i \tag{27}$$

Where $\mathbf{b}_i$ is of course given by:

$$\mathbf{b}_i = \frac{\mathbf{n}_i + \mathbf{n}_{i-1}}{|\mathbf{n}_i - \mathbf{n}_{i-1}|} \tag{28}$$

See Fig. 11 for a visual depiction of these quantities.

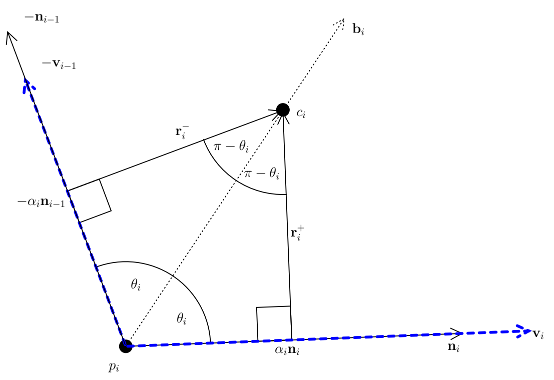


Figure 11: *The interior point p_i with backward vector $-\mathbf{v}_{i-1}$ and forward vector $\mathbf{v}_i$. $\mathbf{b}_i$ is the unit bisector of these vectors*

Angles From Figure 11, we know:

$$\cos\phi_i = \mathbf{r}_i^- \cdot \mathbf{r}_i^+ \tag{29}$$

$$\cos 2\theta_i = -\mathbf{n}_{i-1} \cdot \mathbf{n}_i \tag{30}$$

Combining Equations (29) and (30), we get the following equation for the angle ϕ_i of each arc:

$$\begin{aligned} \phi_i &= 2(\pi - \theta_i) \\ &= 2\pi - \arccos -\mathbf{n}_{i-1} \cdot \mathbf{n}_i \\ &= 2\pi - (\arccos \mathbf{n}_{i-1} \cdot \mathbf{n}_i - \pi) \\ &= \pi - \arccos \mathbf{n}_{i-1} \cdot \mathbf{n}_i \end{aligned} \tag{31}$$

Relating r_i and α_i Say we are given a triple of points p_{i-1}, p_i, p_{i+1} to which we wish to fit an arc. Of the quantities that characterize an arc listed in Section A.1, the orientation $\mathbf{o}_i$, angle ϕ_i depend solely on the normals $\mathbf{n}_{i-1}$ and $\mathbf{n}_i$; see Equations (18) and (31). The center c_i and start radius vector $\mathbf{s}_i$ depend on these normals and the radius r_i.

To fit an arc to an interior point i, we must choose an appropriate radius to complete the definition. The obvious lower bound condition is $r_i > 0$; as upper bound, each radius depends on the geometry of the triple of points about p_i. To remain tangent to both $\overline{p_{i-1}p_i}$ and $\overline{p_ip_{i+1}}$, we must be certain that the points of tangency $-\alpha_i\mathbf{n}_{i-1}$ and $\alpha_i\mathbf{n}_i$ are on those segments. This translates to the following:

$$\alpha_i \leq \min\{L_{i-1}, L_i\} \tag{32}$$

To put this limit in terms of r_i, we need a formula relating r_i, and α_i. From inspection of Fig. 11, we can see that:

$$r_i = \alpha_i \tan\theta_i \tag{33}$$

We know from trigonometry that

$$\tan\theta = \sqrt{\frac{1-\cos 2\theta}{1+\cos 2\theta}} \tag{34}$$

We can combine Equation (33) with Equation (34) to obtain

$$r_i = \alpha_i \sqrt{\frac{1-\cos 2\theta_i}{1+\cos 2\theta_i}} \tag{35}$$

Finally, we can substitute Equation (30) into Equation (35), and obtain

$$r_i = \alpha_i \sqrt{\frac{1-\mathbf{n}_i \cdot -\mathbf{n}_{i-1}}{1+\mathbf{n}_i \cdot -\mathbf{n}_{i-1}}} = \alpha_i \sqrt{\frac{1+\mathbf{n}_i \cdot \mathbf{n}_{i-1}}{1-\mathbf{n}_i \cdot \mathbf{n}_{i-1}}} \tag{36}$$

Then the limits on r_i subject to the following:

$$r_i \in \left(0, \min\{L_{i-1}, L_i\} \sqrt{\frac{1+\mathbf{n}_i \cdot \mathbf{n}_{i-1}}{1-\mathbf{n}_i \cdot \mathbf{n}_{i-1}}}\right]$$

A.2.2 Fitting the r_i

In Section A.2.1, we developed the tools to compute an arc for each interior point of a polyline P given a radius r_i for each.

Given an arbitrary polyline P, it is desirable to automatically select the r_i to complete the definition of a smoothed polyline P_S. A reasonable goal is to pick the r_i such that the quantity

$$\min_{i \in [1, n-2]} r_i \tag{37}$$

is *maximal* over all valid configurations of r_i; this helps minimize the ‘sharpness’ of each corner.

We must consider what values of r_i are valid configurations. The bounds on α_i given in Equation (32) are valid for a given triple of points, but when we are concerned with the α_i for all of the interior points of P, we must consider that the L_i between interior points p_{i-1} and p_i are in contention, i.e.

$$\alpha_i + \alpha_{i+1} \leq L_i \tag{38}$$

where again we set $\alpha_0 = \alpha_{n-1} = 0$.

Fitting algorithm We have a recursive algorithm for selecting the r_i for each arc given a polyline P that satisfies Equation (37). Briefly, we iterate over all of the segments $\overline{p_ip_{i+1}}$, $i \in [0, n-2]$ and consider how *large* a radius it is possible to assign to the arcs i, $i+1$ at either end of the segment i; we take the *smallest* such segment (and associated radius) and assign this radius to the associated arcs. This process is repeated until each interior point has been assigned a radius value.

A key component of the algorithm is how we consider how large a radius can be assigned to the arcs at either end of a segment i; we wish to ‘balance’ the radii of the arcs at either end of the segments such that $r_i = r_{i+1}$. This leads us to:

$$r_i = r_{i+1} = \alpha_i f_i = \alpha_{i+1} f_{i+1} \tag{39}$$

Where we have used Equation (36) and we introduce the convenience

$$f_i = \sqrt{\frac{1+\mathbf{n}_i \cdot \mathbf{n}_{i-1}}{1-\mathbf{n}_i \cdot \mathbf{n}_{i-1}}} \tag{40}$$

Combining Equations (39) and (38), we compute:

$$L_i - \alpha_i \geq \alpha_{i+1} = \frac{\alpha_i f_i}{f_{i+1}}$$
$$L_i - \alpha_i \geq \frac{\alpha_i f_i}{f_{i+1}}$$
$$L_i \geq \frac{\alpha_i f_i}{f_{i+1}} + \alpha_i$$
$$L_i \geq \alpha_i \frac{f_i}{f_{i+1}} + 1$$
$$L_i \geq \alpha_i \frac{f_i + f_{i+1}}{f_{i+1}}$$
$$\frac{L_i f_{i+1}}{f_i + f_{i+1}} \geq \alpha_i \tag{41}$$

This gives us (invoking Equation (38) again):

$$\alpha_i = \min\left\{L_{i-1} - \alpha_{i-1}, \frac{L_i f_{i+1}}{f_i + f_{i+1}}\right\} \tag{42}$$
$$\alpha_{i+1} = \min\{L_{i+1} - \alpha_{i+2}, L_i - \alpha_i\} \tag{43}$$

Since we naturally define $\alpha_0 = \alpha_n = 0$, Equation (42) is not considered when $i = 0$; nor is Equation (43) when $i = n - 1$. Algorithm 2 gives pseudocode for this radius-balancing procedure. The rest of the algorithm is given in detail in Algorithm 3; the procedure ALPHA-ASSIGN is invoked with $s = 0$ and $e = n - 1$ (with A the array of α_i, with $A[0] = A[n-1] = 0$); the r_i are easily computed after ALPHA-ASSIGN completes using Equation (36). Both RADIUS-BALANCE and ALPHA-ASSIGN implicitly make use of (but do not modify) quantities associated with the polyline P: L_i from Equation (16) and f_i from Equation (40).

Algorithm 2 RADIUS-BALANCE

```
RADIUS-BALANCE($i$)
1  $\alpha_a = \min\left\{L_{i-1}, \frac{L_i f_{i+1}}{f_i + f_{i+1}}\right\}$
2  $\alpha_b = \min\{L_{i+1}, L_i - \alpha_a\}$
3  return $\alpha_a, \alpha_b, \max\{f_i\alpha_a, f_{i+1}\alpha_b\}$
```

The RADIUS-BALANCE returns the radius that balances the radii on either end segment i.

Algorithm 3 ALPHA-ASSIGN

```
ALPHA-ASSIGN($A, s, e$)
      // $A$ — array of $n$ $\alpha$ values, $s$ — start segment index,
      // $e$ — end segment index + 1
 1  $r_{\min}, i_{\min} = \infty, e$      // Initialize min. radius, index of min. index
 2  if $s + 1 \geq e$            // Return if interval is length zero
 3      return
 4  $\alpha_b = \min\{L_s - A[s], L_{s+1}\}$
 5  $r_{\text{current}} = \max\{f_s A[s], f_{s+1}\alpha_b\}$         // Radius at initial seg.
 6  if $r_{\text{current}} < r_{\min}$
 7      $r_{\min}, i_{\min} = r_{\text{current}}, s$
 8      $\alpha_{\text{low}}, \alpha_{\text{high}} = A[s], \alpha_b$
 9  for $i = s + 1$ to $e - 2$                    // Radii for internal seg..
10      $\alpha_a, \alpha_b, r_{\text{current}} = $ RADIUS-BALANCE($i$)
11      if $r_{\text{current}} < r_{\min}$
12          $r_{\min}, i_{\min} = r_{\text{current}}, i$
13          $\alpha_{\text{low}}, \alpha_{\text{high}} = \alpha_a, \alpha_b$
14  $\alpha_a = \min\{L_{e-2}, L_{e-1} - A[e]\}$
15  $r_{\text{current}} = \max\{f_{e-1}\alpha_a, f_e A[e]\}$        // Radius at final seg.
16  if $r_{\text{current}} < r_{\min}$
17      $r_{\min}, i_{\min} = r_{\text{current}}, e - 1$
18      $\alpha_{\text{low}}, \alpha_{\text{high}} = \alpha_a, A[e]$
19  $A[i_{\min}] = \alpha_{\text{low}}$  // Assign alphas at ends of selected segment
20  $A[i_{\min} + 1] = \alpha_{\text{high}}$
21  ALPHA-ASSIGN($A, s, i_{\min}$)                 // Recur on lower segs
22  ALPHA-ASSIGN($A, i_{\min} + 1, e$)             // Recur on higher segs
```

The ALPHA-ASSIGN procedure assigning radii based on a polyline P

Optimality of radii-selection algorithm The algorithm described above in Section A.2.2 aims to maximize Equation (37). Here we informally demonstrate that the resulting assignment of r_i makes the value of Equation (37) as large as possible given the shape of the input polyline P.

Consider that we have run ALPHA-ASSIGN on a polyline P. Now consider the set of radii $R_{\min}$ that have the smallest value of radius $r_{\min}$, namely:

$$r_i = r_{\min}, \forall i \in R_{\min} \tag{44}$$
$$r_i > r_{\min}, \forall i \in [1, n-2]/R_{\min} \tag{45}$$

To increase the value of Equation (37), we must increase the value of $r_{\min}$ — i.e increase r_i, $\forall i \in R_{\min}$.

Now recall how ALPHA-ASSIGN works; each call examines the unassigned A in its range and finds the largest radius that could be assigned to each. Then the arc that has the smallest such 'largest' radius is assigned to. Thus the radii assigned (indirectly, through the $A[i]$) in a call must be equal to or greater than that assigned in its caller, and so on up to the top-level call. Then we know that the value of the radius computed in the top-level call to ALPHA-ASSIGN is $r_{\min}$; to show that the computed $r_{\min}$ cannot be improved upon, it is sufficient to show that the two arcs assigned to (one if $i_{\min} = s$ or $e - 1$) in the top-level call to ALPHA-ASSIGN cannot have their radii increased.

Boundary case First consider the case where $i_{\min} = s = 0$ (since this the top-level invocation, $s = 0$). Then in Line 4 of ALPHA-ASSIGN, we know we chose

$$\alpha_b = L_0 - A[0] = L_0 \tag{46}$$

(recall that $A[0] = 0$). Otherwise, that would mean

$$L_0 - A[0] = L_0 > L_1 \tag{47}$$

This would lead to r_{current} having been assigned $f_1 L_1$ at Line 5. But then when $i = 1$ and we invoke RADIUS-BALANCE(1) on Line 10, we know that

$$\alpha_a = \min\left\{L_0, \frac{L_1 f_2}{f_1 + f_2}\right\} = \frac{L_1 f_2}{f_1 + f_2} \tag{48}$$

in Line 1 of RADIUS-BALANCE because $f_2/(f_1 + f_2) < 1$ and $L_0 > L_1$ (from Equation (47)). Then we would have replaced $r_{\text{current}} = f_1 L_1$ from Line 5 with $r_{\text{current}} = f_1 L_1 f_2/(f_1 + f_2)$ at Line 11, and we would not have chosen $i_{\min} = 0$. Thus, by contradiction, if $i_{\min} = 0$ in the first call to ALPHA-ASSIGN, we know that $A[1] = L_0$ and $r_1 = f_1 L_0 = r_{\min}$.

So $A[1] = \alpha_1 = L_0$ and $r_{\min} = r_1 = f_1 L_0$; then the condition in Equation (38) dictates that we cannot increase α_1. Neither, in that case, can we increase $r_{\min}$. A similar argument can be made in the case where $i_{\min} = e - 1 = n - 2$.

General case We shall move on to demonstrating that when $0 < i_{\min} < n-2$, neither of the assigned $A[i_{\min}]$ and $A[i_{\min}+1]$ (nor their associated radii) may be increased.

We shall do this by demonstrating that $r_{i_{\min}} = r_{i_{\min}+1} = r_{\min}$ and that $A[i_{\min}] + A[i_{\min}+1] = L_{i_{\max}}$; then there is no way to increase either of $r_{i_{\min}}$ or $r_{i_{\min}+1}$ without decreasing the other, and thus our $r_{\min}$ must hold.

Consider an $i \in [1, n-3]$ that is the $i_{\max}$ from the top-level invocation of ALPHA-ASSIGN; the associated α_a and α_b must have been computed via RADIUS-BALANCE(i) on Line 10. We contend that Line 1 in RADIUS-BALANCE(i) must have computed $\alpha_a = \min\left\{L_{i-1}, \frac{L_i f_{i+1}}{f_i+f_{i+1}}\right\} = \frac{L_i f_{i+1}}{f_i+f_{i+1}}$.

If it had not, then it would have computed $\alpha_a = L_{i-1}$ and we would know that

$$L_{i-1} < \frac{L_i f_{i+1}}{f_i + f_{i+1}} \tag{49}$$

Furthermore, we know that RADIUS-BALANCE would have returned some Q such that

$$r_{\text{current}} = Q \geq f_i L_{i-1} \tag{50}$$

as computed on its Line 3. Now consider what the computation for $i-1$ must have looked like; if $i > 1$, then we have

$$\alpha_a = \min\left\{L_{i-2}, \frac{L_{i-1} f_i}{f_{i-1} + f_i}\right\} \tag{51}$$

$$\alpha_b = \min\left\{L_i, L_{i-1} - \alpha_a\right\} \tag{52}$$

These equations are from Lines 1 and 2 of a call to RADIUS-BALANCE $(i-1)$. We can combine Equations (49) and (52) to deduce that $\alpha_b = L_{i-1} - \alpha_a$, regardless of the value of α_a. Then, in Line 3, we would compute the radius for $i-1$ to be

$$r_{\text{current}} = \max\left\{f_{i-1}\alpha_a, f_i\left(L_{i-1} - \alpha_a\right)\right\} \tag{53}$$

In fact, we know

$$f_{i-1}\alpha_a \leq f_i\left(L_{i-1} - \alpha_a\right) \tag{54}$$

because α_a is given by Equation (51) — if $\alpha_a = \frac{L_{i-1} f_i}{f_{i-1}+f_i}$, then we know by Equations (39) (42), and (43) that $f_{i_1}\alpha_a = f_i\left(L_{i-1} - \alpha_a\right)$. On the other hand, if $\alpha_a = L_{i-2}$, then by Equation (51), $L_{i-2} \leq \frac{L_{i-1} f_i}{f_{i-1}+f_i}$; thus Equation (54) must be true.

So we know that the radius computed in Equation (53) for $i-1$ is $f_i\left(L_{i-1} - \alpha_a\right)$. Regardless of which value we select for α_a in Equation (51), this radius is obviously less than the Q from Equation (50) that we computed for the segment i that is supposed to be $i_{\min}$. Thus choosing anything but $\alpha_a = \frac{L_i f_{i+1}}{f_i+f_{i+1}}$ in Line 1 in RADIUS-BALANCE(i) results in a contradiction.

In fact, this last discussion particularly demonstrated that contradiction under the assumption that $i > 1$. Having $i = 1$ ($i = 0$ was already covered in our discussion of boundary cases above) results in slightly different Equations (51) and (52), but the contradiction develops on similar grounds.

On our way to show that $r_{i_{\min}} = r_{i_{\min}+1} = r_{\min}$ and $A[i_{\min}] + A[i_{\min}+1] = L_{i_{\max}}$ (as we must do to demonstrate that we cannot increase $r_{\min}$), we have just showed that Line 1 in RADIUS-BALANCE(i) computed $\alpha_a = \frac{L_i f_{i+1}}{f_i+f_{i+1}}$. When we compute α_b in Line 2 of RADIUS-BALANCE(i), we have

$$\alpha_b = \min\left\{L_{i+1}, L_i - \frac{L_i f_{i+1}}{f_i + f_{i+1}}\right\} \tag{55}$$

It is straightforward to show that $L_{i+1} \geq L_i - \frac{L_i f_{i+1}}{f_i+f_{i+1}}$ using a process similar to the one just used to show $\alpha_a = \frac{L_i f_{i+1}}{f_i+f_{i+1}}$; with that condition, clearly $\alpha_{i_{\min}} + \alpha_{i_{\min}+1} = L_i$ and $r_{i_{\min}} = f_{i_{\min}}\alpha_{i_{\min}} = r_{i_{\min}+1} = f_{i_{\min}+1}\alpha_{i_{\min}+1} = r_{\min}$. Since neither $\alpha_{i_{\min}}$ nor $\alpha_{i_{\min}+1}$ can be increased without decreasing the other, and because the radii are equal in size, the minimum radius computed in the top-level call to ALPHA-ASSIGN cannot be made larger; it is the maximum value possible for Equation (37).

A.2.3 Length of a smoothed polyline

It is useful to consider the length of a smoothed polyline P_S. This will be the sum of the circumference of each arc a_i plus the length of the straight line segments connected consecutive arcs and the segments connecting p_0 to arc 1 and arc $n-2$ to p_{n-1}.

The length w_i of an arc i given by the standard formula:

$$w_i = 2\pi r_i \frac{\phi_i}{2\pi} = r_i \phi_i \tag{56}$$

The length s_i of a segment connecting two consecutive arcs i and $i+1$ is simply the length of the original line from p_i to p_{i+1} (L_i) minus the α_i and α_{i+1} of arcs i and $i+1$:

$$s_i = |p_{i+1} - p_i| - (\alpha_{i+1} + \alpha_i) = L_i - (\alpha_{i+1} + \alpha_i) \tag{57}$$

Defining $\alpha_0 = \alpha_{n-1} = 0$ as usual, we can use Equation (57) for the beginning/end segments as well.

For nondegenerate P_S (i.e. with $n > 2$), we have:

$$\begin{aligned}
L\left(P_S\right) &= \sum_{i=1}^{n-2} r_i\phi_i + \sum_{i=0}^{n-2} s_i \\
&= \sum_{i=1}^{n-2} r_i\phi_i + \sum_{i=0}^{n-2} L_i - (\alpha_{i+1} - \alpha_i) \\
&= \sum_{i=1}^{n-2} r_i\phi_i + L\left(P\right) - \sum_{i=0}^{n-2} \alpha_{i+1} - \sum_{i=0}^{n-2} \alpha_i \\
&= L\left(P\right) + \sum_{i=1}^{n-2} r_i\phi_i - \sum_{i=1}^{n-1} \alpha_i - \sum_{i=0}^{n-2} \alpha_i \\
&= L\left(P\right) + \sum_{i=1}^{n-2} r_i\phi_i - 2\sum_{i=1}^{n-2} \alpha_i - \alpha_0 - \alpha_{n-1} \\
&= L\left(P\right) + \sum_{i=1}^{n-2} r_i\phi_i - 2\alpha_i
\end{aligned} \tag{58}$$

A.2.4 Offset polylines

So far, given a planar polyline P and an r_i for each interior point p_i, we can compute the smoothed polyline P_S by computing the associated arcs for each interior point of P using the equations in Section A.2.1; we can even compute the r_i automatically in such a way as to minimize curvature using the algorithm presented in Section A.2.2.

Suppose that we wish to compute a new smoothed polyline P'_S that has the property that at every point, the nearest point on P_S is exactly distance d away. That is, P'_S is 'offset' from P_S to one side by a signed distance d; see Fig. 12. We used the convention that $d > 0$ refers to a 'right' offset (the lower blue line in Fig. 12) and $d < 0$ to a 'left' offset (the upper blue line in the same figure). The new arcs

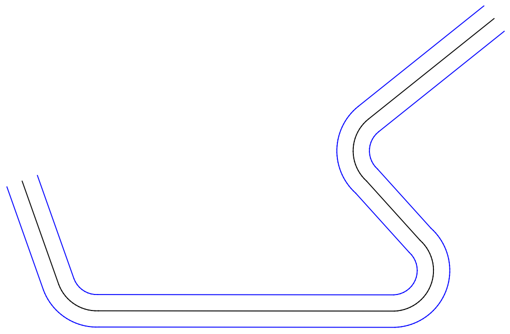


Figure 12: *A 'fattened' smoothed polyline; the original smoothed polyline P_S as computed above is drawn in black. The blue lines represent the same shape offset to either side by an equal distance.*

i corresponding to P'_S (with signed offset d) can be derived from P_S by replacing each r_i with $r_i + d$.

We must also choose new endpoints p'_0 and p'_{n-1} for this line; a reasonable definition is use the plane of the first and last arcs to choose a perpendicular suitable for placing these offset endpoints.

$$p'_0 = p_0 + d(\mathbf{n}_0 \times \mathbf{o}_1) \tag{59}$$
$$p'_{n-1} = p_{n-1} + d(\mathbf{n}_{n-2} \times \mathbf{o}_{n-1}) \tag{60}$$

A.2.5 Discrete approximations of smooth polylines

To visually depict a smoothed polyline P_S, we may wish to compute a discrete representation.

Polylines One obvious way to do this is by approximating the shape by a series of lines — that is to say, a new polyline P^*. See Figure 13a. We must simply approximate each arc i with an sequence Γ_i of $q_i \in \mathbb{Z}_{>1}$ points, and connect those points with lines. Then the sequence of $m = 2 + \sum_{i=1}^{n-2} q_i$ points in P^* is simply:

$$P^*_S = \left(p_0, \Gamma^0_1, \ldots, \Gamma^{q_1-1}_1, \ldots, \Gamma^0_{n-2}, \ldots, \Gamma^{q_{n-2}-1}_{n-2}, p_{n-1}\right) \tag{61}$$

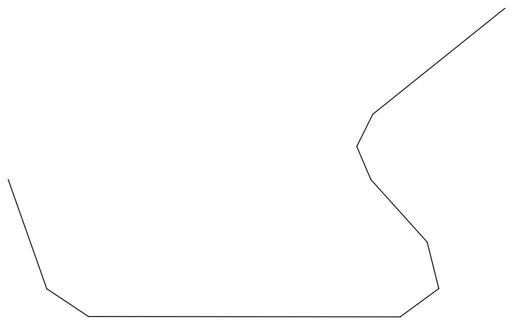
We generate each Γ_i by rotating and scaling the frame $\mathbf{F}_i$ (from Equation (19)) of each arc incrementally and translating by the center c_i:

$$\Gamma^j_i = c_i + r_i \mathbf{F}_i \left[\cos t^j, \sin t^j, 0\right]^{\mathrm{T}}, \; j \in \mathbb{Z}[0, q_i - 1] \tag{62}$$

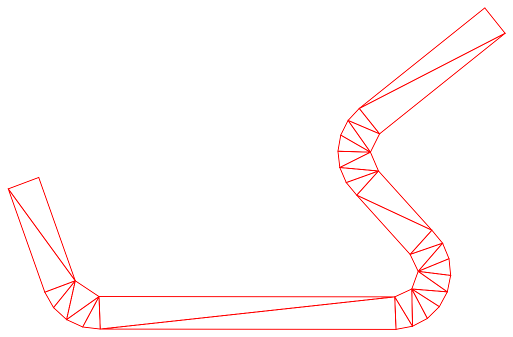
Here the t^j are elements of a sequence $[0, \phi_i/(q_i - 1), 2\phi_i/(q_i - 1), \ldots, \phi_i]$ of length q_i.

Triangle meshes A surface representation of a smoothed polygon can be easily computed from a pair of offset polygons (computed as in Sec. A.2.4). f Given a smoothed polygon P_S and two smooth polygons offset from P_S, order the polygons by offset so that we have a 'left' smoothed polygon P^l_S with a lesser offset than the 'right' smoothed polygon P^r_S.

Now we can use any constrained triangulation technique to compute a planar triangle mesh with P^l_S and P^r_S as the boundaries; see Fig. 13b.



(a) *P^*: A polyline approximation of a smoothed polyline P_S*



(b) *A triangle mesh approximation of a 'fattened' smoothed polyline P_S*

Figure 13: *Discrete approximations of smoothed polylines*

B Validation Experiments and Issues

B.1 The NGSIM project

Despite the ubiquity of traffic in the real world, data collected for a reasonable period of time with correspondingly large spatial scales that are suitable for comparison with simulation techniques can be difficult to find. In addition, it is challenging to set up sensors at these extensive scales, and the problem is made all the more difficult for the necessity of obtaining accurate and precise measurements. Fortunately, reasonably high-quality data has been made available by the Next-Generation Simulation project (NGSIM). This is a joint project between the Federal Highway Administration and a number of partners created to provide useful data to serve the needs of the simulation common. The data provided is per-vehicle trajectory information for relatively short segments of major highways in California.

Data format The NGSIM highway data is organized as a series of time entries each specifying the position, the magnitude of velocity, and the magnitude of acceleration of a given vehicle, among other quantities. A unique identification number associated with each frame is given to identify which vehicle the data describes. Curiously, each frame also specifies the length and width of the vehicle, as well as a discrete 'type' specifying if the vehicle is a motorcycle, truck, or passenger car. For for the data sets we have examined, these remained (blessedly) constant. Each frame also identifies a discrete 'lane number' that the vehicle is traveling in.

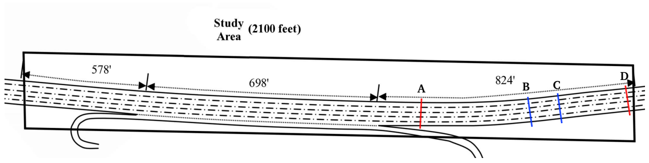


Figure 14: *Section of Highway 101 near Los Angeles. The black rectangle indicates the region where the NGSIM project has recorded trajectories. The red lines (the interval $\overline{AD}$) denote the clipped region used in this comparison, while the blue lines (interval $\overline{BC}$) denote the macroscopic simulation region used in the hybrid simulation test.*

All data is given at 10 frames per second, and this is aligned globally — the position of each vehicle in the system is updated every $\frac{1}{10^{\text{th}}}$ of a second of a global clock. The data for each highway is broken up into several 15-minute intervals. Position information is given in feet to a precision of three digits, while velocity and acceleration data is given to a precision of two digits.

Figure 14 shows the layout of a section of highway 101 near Los Angeles.

Issues It is very useful to have the NGSIM data, although it presents several challenges as well. First, from a validation standpoint, a notion of the accuracy of the measurements in the data would be useful. Unfortunately, no such estimate is given. Additionally, while the given spatial precision of three digits in feet is more than enough, the frame rate of 10 frames per second is quite poor for highway traffic.

Furthermore, the format of the acceleration and velocity data — as scalar magnitudes rather than vector data — is somewhat limiting, since vehicles changing lanes and merging have non-trivial velocity and acceleration vectors.

Finally, while the black box in Figure 14 shows the region where the NGSIM project recorded data, not all vehicle trajectories begin or end where the box actually intersects the highway; some start as much as 30 meters from the beginning of the highway. In order to perform a meaningful comparison, it is necessary to clip all trajectories to the smallest region of the highway where there is data for all vehicles, which somewhat reduces the total amount of trajectory data to work with.

B.2 Methodology

As discussed previously in Section B.1, there is real-world data readily available for comparison with simulation techniques, albeit with some deficiencies. Here, we discuss our approach to achieving a meaningful validation of the results of our technique and real-world data.

Comparison strategies Numerous approaches to validation are taken in the simulation community; these are often developed based on the quality and completeness of real-world data with which to form a comparison. Given a time series of real-world data $\{\eta_0, \eta_1, \eta_2, \ldots\}$, the most straightforward approach is to initialize the simulator with some initial condition $\{\eta_0\}$ and generate the simulated successor states $\{\eta'_0 = \eta_0, \eta'_1, \eta'_2, \ldots\}$.

Such a simple scheme falls prey to a number of complications, ranging from granularity and measurement error in the initial data set to known deficiencies with the simulation technique. Additionally, many classes of phenomena are highly chaotic — even small perturbations in the initial conditions are greatly magnified in successive time steps. Such phenomena is exceedingly difficult to directly compare in a meaningful fashion.

Often, specific qualities of phenomena are tested; simple initial conditions are chosen that are expected to evince certain characteristic features of phenomena — for example, the 'lid-driven cavity' for fluid dynamics described by [Burggraf 1966], or 'arching' behavior in crowd dynamics (see [Guy et al. 2010]).

For agent simulation techniques, such as traffic or crowd simulation, direct trajectory comparisons are usually not performed due to the chaotic nature of the phenomena. Rather, comparisons of averaged velocity and traffic volume over time are common, as in [Treiber et al. 2000]. A cross-sectional region of the road is designated as a 'detector' and the number of vehicles crossing said region are counted along with their average velocities over a series of non-overlapping time spans. It is also sometimes useful to measure the total *flux* of vehicles — the product of the average velocities for all vehicles and the total number of vehicles over an interval. In fact, this is the only quantity that can be directly extracted from macroscopic simulation.

Measuring effectiveness of coupling The hybrid technique presented in this paper consists of both microscopic and macroscopic traffic simulation coupled together through a mechanism described in Section 4.3.

The extent that the base simulation methods — microscopic and macroscopic — are able to match the real-world data gives a baseline for comparison. To gauge the effect of our algorithm on the ability of the underlying simulation methods to match real-world input, we have performed three separate simulations, each of which is compared against the real-world data from NGSIM Project.

Microscopic Pure agent-based simulation is carried out on a section of highway corresponding to the domain of the real-world data — $\overline{AD}$ in Figure 14. Vehicles are introduced to the simulation according to the time in which they enter the corresponding section of the real highway, with the appropriate lane and velocity information. Detectors spaced along the highway — as described above — record crossing times and velocities for the simulated agents (vehicles), which are then accumulated over short intervals.

Macroscopic The same area of highway used for agent-based simulation in the previous simulation is instead used for continuum simulation. Vehicles are introduced as densities and velocities in the appropriate lanes according to the crossing times. Similarly, detectors are employed to compare results against real-world data. However, while agent-based and the real-world trajectory data lend themselves to direct computation of vehicle crossings, the nature of macroscopic simulation dictates that we measure the *flux* of vehicles across detectors — that is, $f = \rho u$. Averaged over intervals, this may be meaningfully compared against the product of crossing densities and velocities from trajectory data.

Hybrid The highway segment used for the simulations above is now divided into *three* segments — a microscopic region including the incoming boundary, an adjacent macroscopic region that occupies the central potion of the highway segment, and an abutting additional microscopic region that includes the outgoing boundary — respectively, $\overline{AB}$, $\overline{BC}$, and $\overline{CD}$ in Figure 14. Vehicles are introduced to the first region just as in the microscopic simulation setup, and our hybrid technique converts vehicles into the neighboring macroscopic region, where they eventually are converted back into discrete vehicles in the succeeding microscopic region. A 'detector' in this final region records crossing times and velocities as in the above microscopic setup.

Boundary conditions Simply supplying initial conditions to a simulator is generally not sufficient if one hopes to achieve a meaningful comparison; we must consider what forces outside the simulation are affecting it. The shape of the highway certainly has an effect; thankfully this is fairly clear from the original data and easily accommodated in the simulation techniques being examined.

Of larger concern is the traffic that is *not* present in the real-world trajectory data; the NGSIM data provided is simply a snapshot of real traffic on a highway — there is necessarily a beginning and an end, and in particular, the leading vehicles in the provided data are behaving according, to some extent, on the behavior of vehicles unknown to us — vehicles that occupy the highway at the start of the simulation, or vehicles further 'down' the highway than the domain captures. It is important to somehow capture the effect of these vehicles on the simulation; in both the agent-based and macroscopic simulation techniques examined in this work, vehicles with unlimited headway — no vehicles ahead of them — will simply accelerate until they reach their maximum velocity; generally, the speed limit of the road.

To account for these unknown leading vehicles, we simply set the velocity of the leading vehicle in each lane in the simulation (leading computational cells, in a macroscopic simulation) to the velocity of the leading vehicle of the corresponding lane in the real-world data.