

Transforming GIS Data into Functional Road Models for Large-Scale Traffic Simulation

David Wilkie, Jason Sewall, and Ming C. Lin

Abstract—

There exists a vast amount of geographic information system (GIS) data that models road networks around the world as polylines with attributes. In this form, the data is insufficient in and of itself for applications such as simulation and 3D visualization – tools which will grow in power and demand as sensor data becomes more pervasive and as governments try to optimize their existing physical infrastructure. In this paper, we propose an efficient method for enhancing a road map from a GIS database to create a geometrically and topologically consistent 3D model to be used in real-time traffic simulation, interactive visualization of virtual worlds, and autonomous vehicle navigation. The resulting model representation also provides important road features for traffic simulations, including ramps, highways, overpasses, legal merge zones, and intersections with arbitrary states, and it is independent of the simulation methodologies. We test the 3D models of road networks generated by our algorithm on real-time traffic simulation using both macroscopic and microscopic techniques.

Index Terms—Virtual World, Geometric Modeling.



Fig. 1. A road network generated directly from GIS data by our method. The road network has been overlaid on top of a satellite image. Note that the cars on the road network are animated using a traffic simulator running on our road network representation.

1 INTRODUCTION

TRAFFIC is an integral component of any virtual environment that attempts to realistically portray the contemporary world, be it a video game, movie, or virtual globe. Traffic is also a global challenge with a direct impact on the economy, energy consumption, and the environment in today's society. Traffic simulation is

a key tool to address both the challenges of traffic and its visualization. However, traffic simulation takes place on a complex domain and realistic road networks. The main objective of this work is to create road network representations from polyline data that can be used directly for real-time traffic simulation and visualization in a virtual world.

Traffic simulation describes large numbers of vehicles on a traffic network by taking advantage of the reduced dimensionality typically found on road networks: vehicles follow roads and their motion can be described with few degrees of freedom. Research on techniques for traffic simulation has been carried out since the 1950s; see the survey of Helbing [1] for a good overview of the field.

Traffic simulation presents unique challenges in the acquisition and representation of the underlying simulation domain, namely the road network. Digital representations of real-world road networks are commonly available, but the level of detail of these data is not immediately usable for many queries related to traffic simulation. Traffic simulations take place on a network of lanes. This network needs to be represented with all its details, including the number of lanes on a road, intersections, merging zones, and ramps.

The work presented in this paper is primarily aimed at augmenting freely-available data sets with sufficient detail to allow for useful vehicle motion synthesis. We introduce an efficient approach for automatically transforming geographic information system (GIS) data, i.e. polyline roads and associated metadata, into functional road models for large-scale traffic simulations. The resulting representation consists of two tightly integrated components, (1) a lane-centric topological representation of complex road networks and (2) an arc-road represen-

The authors are with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599, U.S.A

E-mail: {wilkie, sewall, lin}@cs.unc.edu

tation for geometric modeling of the road networks. The resulting model has the following characteristics:

- It provides a road network representation with the necessary details for traffic simulation and realistic visualization using GIS data as input, instead of digital models created manually;
- The resulting road models are C^1 continuous and well-defined across the entire simulation domain;
- It is computationally efficient for performing geometric operations, such as computing the distance between cars, location-based queries, *etc.*

We demonstrate the effectiveness of the detail-enhanced road networks automatically generated by our technique on two different contemporary traffic simulation techniques, the continuum based method of Sewall et al. [2] and the agent-based simulation method of Treiber et al. [3]. We use these simulators to create traffic visualizations on realistic road networks overlaid on satellite images. In Fig. 1, we can see an example road network generated by our method seamlessly overlaid on a satellite photograph and used for a real-time traffic simulation and visualization.

Challenges. This project entails numerous scientific challenges. First, constructing the intersection, ramp, and road geometries presents numerous special and degenerate cases, typical of geometric computation. Our method is specifically designed to automatically handle as many of these cases as possible. Second, GIS data of road networks are not intended to be used for simulation. We reformulate these networks in order to extrapolate a network on which simulation can be done. Third, the data as available requires filtering in order to be processed; while this is not the main focus of our work, it is a challenge that we have addressed in this paper. Fourth, these networks are large in scale, and so efficient algorithms and implementations are required. Fifth, the scale of the implementation itself is a challenge as this project is a combination of multiple systems, a road network importer, a road network representation, a simulation system, and a visualization system. Finally, there are algorithmic challenges in capturing details such as overpasses and in defining arc roads, which further address the needs of traffic simulation.

The paper is organized as follows. In Section 2, we discuss existing road network representations, both commercial and public domain, and prior work in representing roads. In Section 3, we discuss the specific requirements that traffic simulation imposes on a road network representation and give an overview of our approach. In Section 4, we discuss the topological processing we do in order to create our road network. In Section 5, we discuss the handling of overpasses and underpasses. In Section 6, we discuss our geometric representation. In Section 7, we discuss our results and validation. In Section 8, we present our concluding remarks.

2 RELATED WORK

Digital representations of traffic networks have been widely used for tasks such as civil planning, consumer-level GPS systems, simulation, and visual applications like maps, games, films, and virtual environments, yet each application requires different types of information about the road network. For many display and routing applications, simple graphs with edge metadata are sufficient; for other applications, such as traffic simulation or driving in a virtual world, geometric details about the lanes that constitute the network, their topological arrangement, the layout of intersections, traffic-light timing behavior, road surfaces, and other information are needed.

2.1 GIS Data, Tools, and Software Systems

While digital road networks are widely available, the amount of detail varies widely across sources. Data for North America and Europe are freely available from the U.S. Census Bureau's TIGER/Line[®] database [4] and 'crowd-sourced' community projects like OpenStreetMaps [5], but these data sets contain polyline roads with minimal attributes — information about lanes and intersection structure is wholly missing. Commercially-available data sets, such as those provided by NAVTEQ [6], often contain some further attributes, such as the lane arrangements at intersections, but they are expensive to obtain, the techniques used are not known, and they do not capture all of the desired detail.

Numerous methods have been proposed for automatic and semi-automatic GIS road extraction from aerial and satellite images. Extensive surveys include [7], [8], and [9]. These methods are complimentary to our work: the GIS network we assume as input could be the product of a satellite image extraction method.

Procedural modeling of cities and roads have been an active area of research interest in computer graphics. For example, recent work by [10] and [11], among a notable body of investigation, have enabled the generation of detailed, realistic urban layouts and roads for visualization.

Commercial procedural city modelling software is also available. For example, consider the intersection geometry generated by CityEngine[®] shown in Figure 2. Here, the intersection is modelled as a square connected to neighboring rectangles with narrow triangles. In our work, we construct the geometry for every lane, not just the roads; the lane connections are C^1 continuous, and the geometry defines all the needed parameters for vehicle animation, including orientation and steering angle.

2.2 Geometric Representation

Numerous spatial representations of curves have been developed over the years — see the comprehensive books by Farin [13] and Cohen et al. [14]. However,

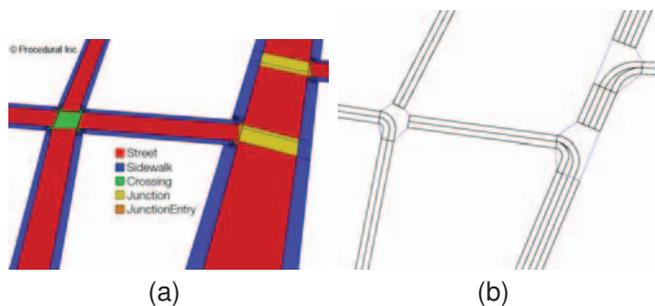


Fig. 2. Geometry for a simple road network created by CityEngine[®][12] is shown in (a). In (b), we show the geometry created by our method for a similar road network. Note that only the lanes in the intersections that currently have a green light are shown.

road networks and traffic behavior have specific requirements: existing curve representations are not the best suited for modeling road networks to support real-time traffic simulations.

For example, the popular NURBS formulation [15], despite of its generality of representations, is costly in space and efficiency. In particular, many splines do not readily admit arc-length parametrizations: those must be obtained using relatively expensive numerical integration techniques for establishing vehicle positions and for describing quantities of vehicles on each lane in traffic simulators.

Willemsen *et al.* [16] describe *ribbon networks*, specifically discussing the need for ‘fattened’ splines to describe road shapes, and our technique is potentially complementary to the modeling technique for road networks they present. However, they use the representation of Wang *et al.* [17], which is only approximately arc-length parametrized and requires iterative techniques for evaluation. In contrast, our method only needs a simpler and much cheaper direct evaluation.

van den Berg and Overmars [18] proposed a model of roadmaps for robot motion planning using connected *clothoid* curves. However, their choice of representation is based solely on the need to generate vehicle motion. For both traffic visualization and simulation, the representation must also be suitable for the generation of road surfaces, which are not necessarily clothoid curves. Additionally, clothoid curves are expensive to compute — requiring the evaluation of Fresnel integrals — whereas our method relies solely on coordinate frames, sines, and cosines.

Nieuwenhuisen *et al.* [19] use circular arcs, as we do, to represent curves, but these arcs are used to smooth the corners of roadmaps for motion planning as in [18]. Furthermore, neither of these techniques have been investigated for the case of extracting ribbon-like surfaces, as we do, nor is there an established technique for fitting them to multi-segment, non-planar polylines.

We have developed an *arc road* representation that offers (1) a visually smooth (C^1) appearance and close

resemblance to real roads, (2) an ease of extension from widely available polyline data, and (3) a low cost to compute, evaluate, and perform geometric queries on the road model.

3 PRELIMINARIES

3.1 Simulation Requirements

The common formulations for traffic simulation are lane-based. These lanes are treated as queues of cars, represented either as discrete agents or by continuous density values. For traffic simulation, lane geometry is irrelevant as long as speed limits and distances are available. However, geometry matters for visualization and for localizing data, such as cell phone or GPS transmissions sent to inform about traffic conditions. These lanes are connected in various ways to form a road network, and cars traverse these connected lanes by crossing intersections and merging between adjacent lanes.

The principle requirement for simulation is the creation of this network of lanes. This includes the division of roads into lanes, but also the creation of transient ‘virtual’ lanes within intersections: these virtual lanes exist only during specific states of a traffic signal. The creation of the network of lanes also entails determining the topological relationships between lanes (so that vehicles can change lanes and take on- and off-ramps) and making geometric modifications to the road network to allow the construction of 2D or 3D road geometry.

To efficiently support traffic simulation, there are a number of queries the network needs to be able to answer in a computationally efficient manner. The nature of these queries depends on the simulation technique, (i.e. whether the technique is continuum-based or discrete). Additionally, it is desirable that the road network representation abstract away the details the queries on the road network to maintain clear separation and software modularity between the traffic simulation and the road network.

3.1.1 Discrete Simulation

A discrete formulation, commonly called *microscopic* simulation (e.g. agent-based simulations), focuses on the interactions between individual cars, typically by using a *leader-follower* formula to calculate each cars’ acceleration. For example

$$a_c = f(v_l, a_l, |c - l|)$$

calculates the acceleration for the car c based on the acceleration and velocity of the leading car l as well as the distance between c and l . Therefore, one requirement is that the road network representation be able to facilitate this leader-follower query. The specific formulas for this type of equation vary, but they typically require the state of the leading car and the distance along the road to that car, which we respectively call $get_leader(c)$ and $get_free_dist(c)$.

$get_leader(c)$ is defined as a mapping from a car c to a car l . Let R_c be the *route* of c , where *route* is defined as an ordered set of roads such that for $r_i, r_{i+1} \in R_c$, the last vertex of r_i is the first vertex of r_{i+1} . Note that this does not require the simulator to use routing: the route can be defined as the current free path through the network ahead of the car c . It must be the case that for c and $l = get_leader(c)$, no cars exist between c and l along R_c . When there are no cars along R_c (or when there are no cars on R_c up to some specified distance), $get_leader(c)$ must return a *virtual car*. The state of this virtual car can be defined on a per simulation basis: some reasonable definitions would be 1) a stationary car at a position sufficiently far ahead of c as to have a minimal impact on its calculations, i.e. the free distance is expected to dominate the leader-follower calculation, and 2) a car moving at the speed limit of some road in R_c at a sufficient distance ahead of c . For boundaries, such as the end of lanes and temporary stops at intersections, a virtual car of type (1) should be returned such that its position is at the end of the lane.

$get_free_dist(c)$ is defined as the distance from c to $l = get_leader(c)$ along R_c . This operation is dependent on the geometric structure used and motivates our method of *arc roads*, which have a closed form for length calculation.

3.2 Continuum simulation

For continuum formulations, commonly called *macroscopic* methods, the lanes are divided into cells where traffic state data are stored. As with the microscopic formulation, this requires that distances along the lanes can be computed.

Both formulations require that the network have the capability to efficiently cycle through the cars in all the lanes, in order to update their states (or update the continuum quantities of all the lane elements). Additionally, cars must be easily moved between lanes to allow for merging behavior and intersection traffic. Finally, for both visualization and for accurate representation of roads, the road network must use a visually smooth (C^1) geometric representation for lanes.

In summary, our method constructs a representation capable of efficient simulation by fulfilling specific requirements for traffic simulation, such as

- **A network of lanes:** we construct a graph with formal properties, then process the graph to construct a network of lanes with the correct topological relationships, including temporal connections at intersections and intervals that allow merging.
- **Intersections with connections and states:** we use a geometric method to truncate roads at intersections and create internal lanes for the intersection to allow through traffic. Our method can ensure that no turn is made that would violate a car's kinematic constraint on turning radius.

- **Fast calculations for $get_leader(c)$ and $get_free_dist(c)$:** our method uses a geometric representation with a closed form length formula and a well-defined network of lanes and intersections for easy graph traversal.
- **Simple interface between simulation and the road representation:** our system allows for a high-level language interface. The road network representation is independent from any single simulation methodology.
- **Visually smooth spatial representation:** for visualization and increased accuracy, we introduce a formulation for representing roads as arc and line segments with C^1 continuity that can be quickly created, queried, and used to create 3D mesh geometry.

3.3 System Overview

Our system takes a road network representation from a GIS source as input. This representation is assumed to contain polyline roads along with metadata consisting primarily of road classifications. From these road classifications, we estimate data such as the number of lanes on the road and the speed limit.

There are two phases for our system and two resulting outputs. First, there is a topological phase, in which the semantics of the network are encoded in a graph. And second, there is a geometric phase, in which the lanes and intersections are described by visually smooth, ribbon-like geometry.

In the topological phase, we first enforce constraints on the network. Primarily, as will be discussed below, we enforce a formal definition of a road as a polyline with two boundary vertices of degree not equal to two and all internal vertices having degree two. GIS data often requires filtering, including removing duplicate nodes, ensuring the vertices in a road follow the logical order of the road, ensuring one way roads are defined in the correct direction, *etc.*. We discuss filtering below.

This phase also ensures that all the interfaces between the lanes are well-defined: normal intersections have states and internal lanes; neighboring lanes have merging zones defined and the functionality for a simulator to use the zones; and ramps flow into highway merging lanes, even if the final geometry of the ramps is not yet defined.

In the geometric phase, every lane is assigned boundary curves that are calculated using the underlying polyline road representation, the offset of the lane from the road's center line, and a geometric representation introduced in Section 6. This representation both captures the curves of the physical roads and allows fast distance calculations needed for the simulation formulation.

3.4 GIS Data Filtering

We filter the GIS data we use to remove the most commonly occurring errors. These changes are not meant to change the underlying geometry or topology of the

network, only to correct sloppy data creation. The first filter removes points that are ϵ -coincident, where ϵ is a distance argument that is kept on the order of feet. This is done prior to the splitting and joining algorithms discussed in Section 4.1, while the remaining filters are applied afterwards. The second filter removes collinear points within roads. The third filter ensures that no point added to a road causes it to turn too sharply or double back on itself. This filter calculates the offset, as in Figure 3, that would be required for a circle of minimum turning radius to be inscribed within the polyline segments. If this offset is greater than half the length of either segment, the node is not added. This ensures that when a point is added to the road, the road still satisfies the kinematic constraints of a typical car. Further filtering includes ensuring that one way roads are defined in the correct direction and that roads have been assigned the correct classification.

4 A LANE-CENTRIC GRAPH REPRESENTATION

This section discusses the transformation of GIS map data into a road network representation suitable for use in traffic simulation.

For the purposes of formal communication, we present aspects of this process using matrix notation. The road network can be represented as a directed graph, consisting of vertices, V , and edges, E . Every edge $e \in E$ has a starting vertex, e_s , and an ending vertex, e_e . We assume the vertices are sampled along the center lines of the physical roads of the network. We can describe the connectivity between the edges and vertices using a graph represented by an incidence matrix, M ,

$$M_{|V|,|E|} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,|E|} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,|E|} \\ \vdots & \vdots & \ddots & \vdots \\ m_{|V|,1} & m_{|V|,2} & \cdots & m_{|V|,|E|} \end{pmatrix}.$$

Each element of the matrix at row i and column j is defined as

$$m_{i,j} = \begin{cases} 1 & \text{if } v_i \in e_j \\ 0 & \text{if } v_i \notin e_j \end{cases}$$

Every vertex has the operator $degree$ defined as the number of coincident edges, $degree(v_i) = \hat{e}_i^T \cdot M = \sum_{j=0}^n m_{i,j}$.

4.1 Roads

We introduce the data structure of *road* and define it as an ordered set of vertices, R , with a starting vertex r_s and an ending vertex r_e such that for all $r_i \in R$, $degree(r_i) = 2$ if and only if $r_i \notin \{r_s, r_e\}$ and $edge(r_i, r_{i+1}) \in M$. This implies that a *road* ends at a higher degree node or a node with degree one, *i.e.* an intersection or a dead end.

While GIS data sets have roads defined, it is likely that the data contains errors or does not strictly adhere to the rules we want to assume. To ensure the above definition

holds on our data set, we perform two operations, *road split* and *road join*. These operations are performed on sets of vertices derived from GIS polylines.

4.1.1 Road Split

Let the internal vertices, $internal(R)$, be all $r_i \in R$ such that $r_i \notin \{r_s, r_e\}$. To satisfy the *road* definition given above, $\forall r_i \in internal(R)$, $degree(r_i) = 2$. Intuitively, this differs from the colloquial use of *road* in that *roads* do not go through intersections: they start and stop at dead ends or intersections.

The split operation is defined as a mapping from a set of vertices $p \in P$, where $edge(p_i, p_{i+1}) \in M$, to a set of sets $S = \{S_0, S_1, \dots\}$ such that for all $S_i \in S$, for all $s \in internal(S_i)$, $degree(s) = 2$ and $\bigcup S_i = P$. This is achieved by Algorithm 1.

Algorithm 1 Algorithm for Road Splitting.

Require: A set of vertices V' such that $edge(v'_i, v'_{i+1}) \in M$.

Ensure: For all output $S_j \in S$, for all $v \in internal(S_j)$, $degree(v) = 2$.

```

 $S = \{\}$ 
 $S_j = \{V'_s\}$ 
for all  $v \in internal(V')$  do
   $S_j \leftarrow v$ 
  if  $degree(v) > 2$  then
     $S \leftarrow S_j$ 
     $S_j = \{v\}$ 
  end if
end for
 $S_j \leftarrow V'_e$ 
 $S \leftarrow S_j$ 
return  $S$ 

```

4.1.2 Road Join

A set S_i described above differs from a *road* only in that it lacks sufficient constraints on its starting and ending vertices. This condition, $degree(v_s) \neq 2$ and $degree(v_e) \neq 2$, is satisfied by Algorithm 2, which iterates over each vertex and joins neighbors S_i and S_j if their coincident vertex has $degree(v_c) = 2$. This algorithm uses $roads(v)$, which maps a vertex to the set of *roads* coincident with that vertex: $roads(v) = \{R | v \in R\}$.

Of final note in Algorithm 2, the *join* operation adds every vertex of its second argument to its first argument in order and removes the vertices from the second argument, updating $roads(v)$.

4.1.3 Proof of Road Creation

Before proving that the above creates *roads*, we define a *degenerate road* D as a *road* in all ways except for $degree(d_s) = degree(d_e) = 2$ and $roads(d_s) = roads(d_e) = D$. In other words, a degenerate road is a loop, disconnected from the rest of the network.

Algorithm 2 Algorithm for Road Joining.**Require:** The set of all vertices V in M , a set of road R **Ensure:** For all $v \in V$, $\text{degree}(v) = 2 \Rightarrow |\text{roads}(v)| = 1$.

```

toDelete  $\leftarrow$  {}
for all  $v \in V$  do
  if  $\text{degree}(v) = 2$  then
    if  $|\text{Roads}(v)| = 2$  then
       $a, b \leftarrow \text{Roads}(v)$ 
      if  $a_e = b_s = v$  then
         $\text{swap}(a, b)$ 
      end if
      if  $a_s = b_e = v$  then
         $\text{join}(a, b)$ 
         $\text{toDelete} \leftarrow b$ 
      end if
      if  $a_e = b_e = v$  then
         $a \leftarrow \text{reverse}(a)$ 
         $\text{join}(a, b)$ 
         $\text{toDelete} \leftarrow b$ 
      end if
      if  $a_s = b_s = v$  then
         $b \leftarrow \text{reverse}(b)$ 
         $\text{join}(a, b)$ 
         $\text{toDelete} \leftarrow b$ 
      end if
    end if
  end if
end for
return  $R \setminus \text{toDelete}$ 

```

Theorem 1: Given a road network M and disjoint sets of vertices $S_i \in S$, the result of applying Algorithm 1 to each set S_i and applying Algorithm 2 is a set of roads $R_i \in R$ and a set of degenerate roads.

Proof: Suppose on the contrary there exists a set of vertices R' produced by the above methods that is not a road or degenerate road.

R' then either has a vertex $v \in \text{internal}(R')$ with $\text{degree}(v) \neq 2$ or a vertex $u \in \{r_s, r_e\}$ with $\text{degree}(u) = |\text{roads}(u)| = 2$.

For any vertex $v \in V$, Algorithm 2 will ensure that v cannot have $\text{degree}(v) = 2$ and $|\text{roads}(v)| = 2$. Every vertex is processed. For any vertex with $\text{degree}(v) = 2$ and $|\text{roads}(v)| = 2$, one of the exhaustive joining cases will be executed resulting in $|\text{roads}(v)| = 1$. As no vertex exists with $\text{degree}(v) = 2$ and $|\text{roads}(v)| = 2$, the road R' cannot begin or end at such a vertex. Therefore, R' must either begin and end at vertices with $\text{degree}(v) \neq 2$, or R' must be a degenerate road that begins and ends and the same vertex.

Therefore, R' must have a vertex $v \in \text{internal}(R')$ with $\text{degree}(v) \neq 2$. However, as R' is a result of Algorithm 1, and as Algorithm 1 splits the set at every vertex with $\text{degree}(v) > 2$, no vertex in $\text{internal}(R')$ can have $\text{degree}(v) > 2$. Further, no vertex in $\text{internal}(R')$ can have $\text{degree}(v) < 2$, as that would contradict v being

an internal vertex.

Therefore, every vertex $v \in \text{internal}(R')$ has $\text{degree}(v) = 2$ and neither v_e nor v_s have $\text{degree}(v) = 2$ and $|\text{roads}(v)| = 2$. R' is either a road or degenerate road, which contradicts our assumption. \square

4.2 Lanes

The commonly used simulation formulations are lane-based. Therefore, lanes must exist to hold cars, and they must have a relation to the roads. We assume that every road has a known number of lanes, and that these lanes belong fully to their associated roads. Each lane has the following data: an offset value, which defines how far its center line is displaced from the road center line; adjacency intervals, which define which lanes are adjacent to the lane and where they are adjacent (to allow for merging); a road membership, and a lane width value. The adjacency intervals of a lane are defined as $\{A_1, A_2, \dots, A_n\}$, where $A_i = \{s_i, e_i, os_i, oe_i, l_i\}$ and $s_i \in [0, 1]$ is the parametric starting point on the lane of the adjacency interval, e_i is the intervals parametric ending point, and os_1 and oe_1 are the parametric bounds for the adjacent lane. l_i is a reference to the lane which is adjacent in the i^{th} interval. The road membership is simply one interval $\{s, e\}$ where $s, e \in [0, 1]$ are the parametric bounds that determine where on the road the lane starts and where it stops.

4.3 Intersections

Our road network contains polyline roads that terminate at dead ends or at intersections. In a realistic road network, intersections have their own geometries. For physical roads that meet at intersections, we can say that the roads are 2-manifolds with boundaries. As simulation systems require 1D lane structures, it is not sufficient to only create the geometry of these intersections; lanes also need to be created to define how traffic can move through the intersection at time t .

In this work, we consider two classes of intersections, signaled intersections and highway ramps. Other classes of intersections, such as n -way stops or traffic circles, have similar geometric construction as the intersection classes described here, but they require different handling at the simulation level. Signaled intersections feature a traffic light that determines the state of the intersection. This state defines which incoming lanes can send traffic into the intersection and to which outgoing lanes that traffic can flow. In our representation, this corresponds to a state defining which internal lanes exist at a certain time. For ramp class intersections, one road becomes an additional lane for a second road for some spatial interval. This allows cars on the first road to merge onto or off of the second. Our method uses a rule-based classifier¹ to determine the intersection

1. Our system classifies based on the road type information provided in the GIS metadata. Intersections on the highway type roads are treated as ramps, and all other intersections are considered signaled.

type, but the classifier is separate from the intersection construction, and a more advanced classifier could be used with no modification to our method. For example, a classifier using a machine learning technique on satellite image data could be used to determine intersection class.

4.3.1 Signalized Intersections

Let $s \subset V$ be the vertices classified as signalized intersections. As shown in Fig. 3, we calculate an offset o for each road that is dependent on the desired minimum turning radius of the intersection, which is a user specified value that can be intersection specific and parametrized by speed limit, road type, or other road safety requirements, for example.

To calculate this offset, the roads are sorted by the angle each forms with the x-axis to yield a clockwise ordering. For each road R_j , we calculate the offset needed for a circle of the specified radius to be tangent to both the boundary of R_j and its clockwise and counterclockwise neighbors. The final offset assigned to R_j is the maximum offset found for either neighbor, which guarantees that no radius smaller than the specified is needed to make a turn from the end of R_j to either of its neighbors.

For some roads, the offset calculated to satisfy the minimum turning radius will be longer than the roads themselves. This is typically the case for small roads and roads that make very acute angles. If an offset for a road R is longer than the length of R , we propose collapsing the vertices v_e and v_s , the starting and ending vertices of R , combining the intersections those vertices form. The road R is then deleted from the network. As the vertices were collapsed, the topology of the network is preserved, if not the geometry.

States. Timer-based signalized intersections have an ordered set of states S in which each state $s \in S$ is defined as $s = \{P, h\}$, where $P = \{\{I_1, O_1\}, \{I_2, O_2\}, \dots, \{I_m, O_m\}\}$ and $\{I_j, O_j\}$ is a pairing of an input lane and an output lane, and h is the duration for the state. The actual states for an intersection are unknown from the GIS data alone. Therefore, we assume that every pair of roads in $roads(v)$ are joined in a state, and each state is of equal duration. Further data on the actual states or more advanced methods of estimating the states could trivially be integrated with our approach.

4.3.2 Ramp Intersections

For vertices classified as ramp intersections, we will call one road the *ramp* and one road the *highway*, as this is where this class of intersection commonly occurs. Our end goal is to have the ramp end alongside the highway and to have a merging lane added to the highway for an interval before or after the ramp, depending on whether the ramp is an *onramp* or *offramp*². The steps needed to

2. The ramps are defined in the direction of the flow of traffic. If last vertex in the ramp is the intersection point, the ramp is an onramp. Else it is an offramp.

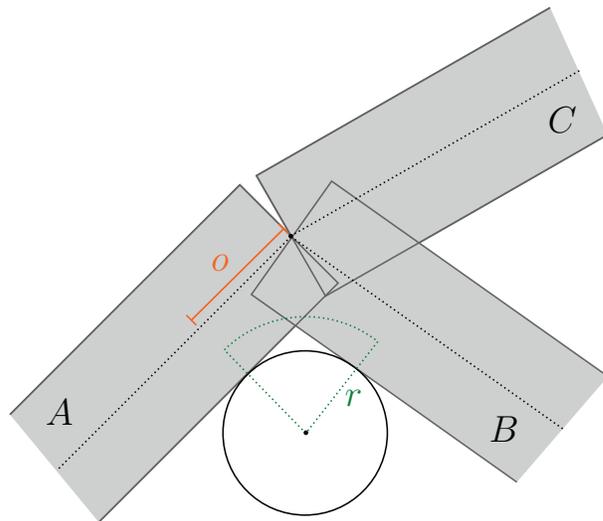


Fig. 3. A simple intersection with three roads. For each road, we calculate an offset, o , based on each of its neighbors. Here, we see the calculation of the offset for the road A with respect to B . To calculate the offset, first the position of a circle tangent to A and B is calculated with a radius such that a car turning from A to B will have a turning radius of r . The offset is then the length on A from the intersection to the projection of the center of the circle onto A .

perform this transformation are 1) joining the highway roads that connect at the intersection, 2) transforming the geometry of the ramp so that the ramp becomes tangent to the highway, and 3) adding a merging lane to the highway.

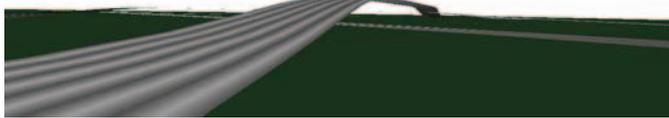
1) To accomplish this, we remove R_m from $roads(v_t)$ and decrease $degree(v_t)$ by one. We then execute Algorithm 2 on the intersection point to merge the highway roads that contain it.

2) The ramp needs to be tangent to the highway so cars do not appear to vanish from one road and appear on another or undergo a sudden change in orientation. To do this, we create a new vertex v_r to serve as the ramp's intersection point. We locate the closest point p on the highway's geometric representation at an offset equal to the $(n + 1)^{th}$ lane, where n is the number of lanes of the highway. The intersection point of the ramp is set to be p . Additionally, a vector \vec{u} tangent to the highway at p is calculated in the opposite direction of the ramp. A vertex is added to the ramp equal to $\vec{u} * \epsilon + p$ to ensure that the ramp approaches tangent to the highway.

3) Finally, the cars entering or leaving the highway need an interval in which they can merge onto or off of the highway. We consider these merging lanes to be part of the highway. Therefore, we add a merging lane to the highway that starts at the parameter value of p and continues for a user-defined distance. In our demonstrations, we used a value of 60m.



(a) An overhead view of a highway interchange with overpasses. The highways are colored to show the arrangement computed by our method.



(b) A view of the blue highway, which is given the greatest height.

Fig. 4. A highway interchange with overpasses generated by our method.

5 OVERPASSES AND UNDERPASSES

Another feature of real road networks is the presence of overpasses and underpasses, including the complex weaving of roads at multi-highway interchanges. While it is only necessary to capture the topological relationships between these roads for current traffic simulation formulations (as traffic simulations only take place in 1D), a reasonable method for estimating road heights is needed for 3D modeling. These height calculations can be done following the procedures described above, resulting in overpasses as shown in Figure 4.

We assume that underpasses and overpasses are represented in the GIS data by roads that contain line segments that intersect but have no shared nodes. In other words, the overpasses are represented implicitly, and the point of intersection must be calculated from the data.

5.1 Intersection Points

The problem of calculating intersection points among line segments lacking any structure is well-studied in computational geometry, with optimal algorithms being $\mathcal{O}(N \log N + K)$ where N is the number of line segments and K is the number of intersecting points [20]. These algorithms typically consider segments with shared endpoints to be colliding, which is not a desired assumption for our application. To handle this, every road segment of every road can be truncated by ϵ to avoid these intersections being found, or a post-processing step can

be done to discard all intersections for which the only intersecting segments are neighbors.

5.2 Road Height Levels

Once the intersection points have been either given or found, the road heights at those points need to be determined. In simple cases, one road will be above and one below. The specific order might be given in the metadata. In general, however, there could be multiple levels of overpasses. Our method divides the overpasses into levels with the goal of having the smallest number of roads elevated, as would be the most cost-efficient approach for a real road system.

We consider the k intersection points and their corresponding roads. As a pre-processing step, we first divide every road into multiple sets such that (1) no road intersects itself and (2) no two intersections on a road are separated by a distance greater than δ . The reason for (1) is so that we can consider every road as being on a single level, and the reason for (2) is that intersections that are separated by a large distance should be considered independently.

Given the above, our proposed method is to generate the set of conflicting roads, C , for each road with an intersection. The roads are then sorted by $|C|$ in ascending order³ and placed in a priority queue Q_1 . For the i^{th} step of the algorithm, we build level i : until Q_i is empty, we remove the road j with the smallest $|C_j|$. This road is set to level i . For every road c in C_j , we remove c from Q_i , and add it to Q_{i+1} . When Q_i is empty, the loop repeats for Q_{i+1} until no roads remain.

The result of this loop is that all the roads that don't conflict with one another are placed on the same level, and the levels are ordered such that those roads that conflict with the greatest number of other roads are placed at higher levels: this is a greedy approach to create levels that have fewer roads the higher the level.

6 GEOMETRIC MODELING OF ROAD NETWORKS: ARC ROADS

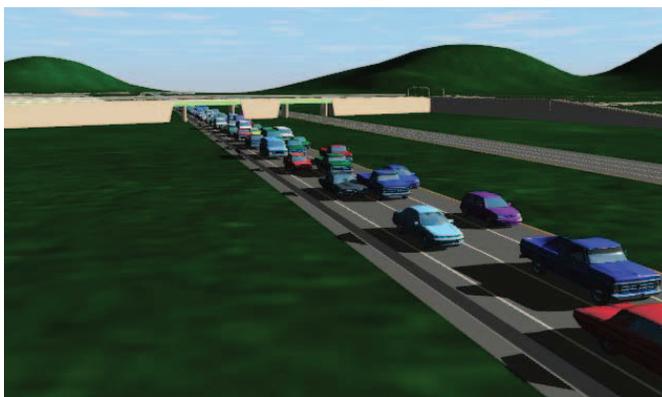
Most digital maps use *polylines* — C^0 series of line segments — to represent road shapes. However, real-world roads are curved, and these polylines visibly deviate from the real shape of the road and give the road an angular shape. Furthermore, linear segments generally lead to visible artifacts in the motion of vehicles along these roads: the sudden change in direction between segments produces large instantaneous changes in vehicle orientation, in violation of car kinematics. Figure 6(a) shows a polyline-based road imported from GIS data.

Given the true curve of the underlying road, one could simply refine the approximation of the underlying curve

3. We can also add meta data into the sorting procedure. For example, if we wanted highways to be underpasses wherever possible, we could add road type as a tie-breaker in the sort.



(a) A large scene featuring two towns and a highway.



(b) An overpass created with our method and procedural modeling.



(c) A suburban neighborhood road network created by our method and aligned with a satellite image.



(d) A divided highway with two ramps created by our method.

Fig. 5. A collection of road networks, all generated by our method. The traffic in these figures was simulated using the technique of [2]

by adding more straight line segments. This can give acceptable results in certain cases, although it leads to a proliferation of data points. However, barring extreme levels of refinement, the C^0 nature of the representation is still observable, and it should also be noted that there is no clear method by which to use polylines in three dimensions to describe vehicle motion with consistent orientation. However, we have only the polylines in the input GIS data set to work with and lack information needed for such refinement.

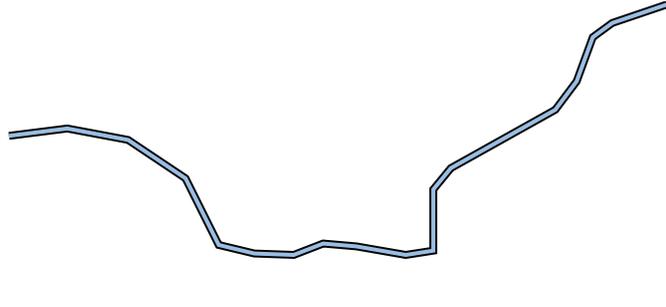
Numerous techniques have been proposed for fitting curves to polyline road data and for the refinement and smoothing of these curves [21], [22], [23], [24]. These methods are useful for visual description, but can be computational obstacles for simulation. For example, splines are frequently costly to compute with. B-splines have no general closed form for arc-length and require numerical quadrature to evaluate. Traffic simulation techniques frequently represent vehicles' positions along roads parametrically. To advance the position of vehicles along roads, information about velocity is integrated into position and translated into parametric space; to display vehicles' locations, this parametric information

is translated into a point x in \mathbb{R}^3 . This process can occur multiple times per vehicle per simulation step, and when it requires quadrature, its expense can dominate runtimes.

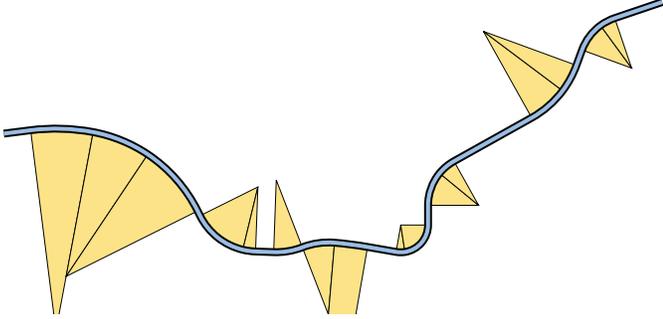
As an alternative, we propose *arc roads*, which consist of alternating straight line segments and circular arcs. This representation has numerous advantages over polylines:

- The curve is C^1 .
- It is well-defined in three dimensions and suitable for traffic simulation — a consistent Frenet frame defining 'forward', 'up', and 'right' is available at each point.
- It is straightforward to derive from existing polyline data.
- It admits a simple and inexpensive parametrization.
- It allows efficient computation of vehicle position and orientation.
- It enables much smoother animations of vehicle motion.

Figure 6(b) shows an exemplary arc road derived from the polyline shown in Figure 6(a). Arc roads are particularly suitable for describing vehicle motion; the turning



(a) Polyline road geometry (imported from the TIGER[®] database [4] via OpenStreetMap [5])



(b) An *arc road* derived from the above polyline. The orange arcs show the center and radius of each arc used to give the road its smooth appearance.

Fig. 6. A polyline and derived arc road

behavior it prescribes matches the kinematic model of the simple car [25].

6.1 Arc Formulation

We have an ordered sequence P of n points:

$$P := (p_0, p_1, \dots, p_{n-2}, p_{n-1}) \quad (1)$$

These points define a polyline, which is not necessarily planar, with $n - 1$ segments such as that in Figure 6(a). Assume, without loss of generality, that there are no two points adjacent in the sequence that are equal, and that there are no three adjacent points that are collinear; we wish to smooth this polyline to what is shown in Figure 6(b), which we shall refer to as P_S . We construct P_S by replacing the region around each interior point p_i 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 7. 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 $\overline{p_i p_{i+1}}$.

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

$$\mathbf{v}_i = p_{i+1} - p_i \quad (2)$$

$$L_i = |\mathbf{v}_i| \quad (3)$$

$$\mathbf{n}_i = \frac{\mathbf{v}_i}{L_i} = \frac{\mathbf{v}_i}{|\mathbf{v}_i|} \quad (4)$$

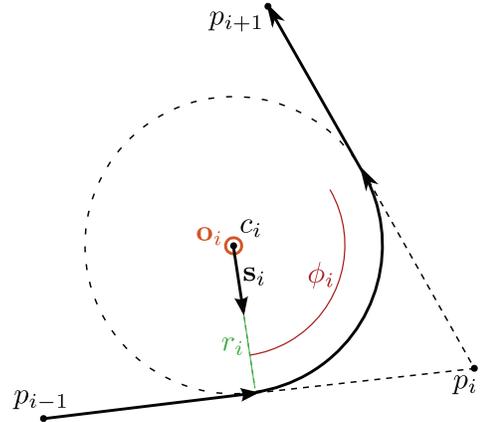


Fig. 7. The quantities defining an arc i corresponding to interior point p_i . The orientation vector \mathbf{o}_i is coming out of the page.

These are: vectors from point p_{i+1} to point p_i (Eq. (2)), the vector lengths (Eq. (3)), and associated unit vectors (Eq. (4)), respectively. We will also refer to $-\mathbf{n}_{i-1} = \frac{p_{i-1} - p_i}{|p_{i-1} - p_i|}$ and to the normal of the plane containing the circle,

$$\mathbf{o}_i = -\mathbf{n}_{i-1} \times \mathbf{n}_i \quad (5)$$

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

$$\mathbf{F}_i = [\mathbf{n}_i \quad \mathbf{s}_i \quad \mathbf{o}_i] \quad (6)$$

Parametrization: Arc roads admit straightforward parametrizations $P_s(t) = \vec{x}$ because their lengths are simple to compute. They are simply the length of the original polyline P adjusted by the difference between each arc and the ‘corner’ of the polyline it replaces. Just as with a polyline, parametrization operations can be accelerated by storing the cumulative length of each segment and arc. Binary search can then be used to find the relevant length for a given t . This is considerably less compute-intensive than performing quadrature to determine lengths, as is necessary for many spline representation.

Speed limit estimation: Where input GIS data lacks detail about speed limits, arc roads can be used to estimate speed limits:

$$v_{\max} = \sqrt{g\mu_s r} \quad (7)$$

Here g is acceleration due to gravity, μ_s the coefficient of static friction, and r the radius (of a given arc) — v_{\max} is the highest velocity achievable on the arc without slipping; safe speed limits should be proportional to this value to meet the road safety requirements.

6.2 Fitting arc roads to polylines

Given an arbitrary polyline P , it is desirable to automatically select the r_i to complete the definition of a C^1

smoothed polyline P_S . A reasonable goal is to pick the r_i such that the quantity

$$\min_{i \in [1, n-2]} r_i \quad (8)$$

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

We have developed a recursive algorithm for selecting the r_i for each arc given a polyline P that satisfies Equation (8). Briefly, we iterate over all of the segments $\overline{p_i p_{i+1}}$, $i \in [0, n-2]$ 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 radius over all considered segments and assign it to the associated arcs. This process is repeated recursively until each interior point has been assigned a radius value.

In the interest of space, we have omitted the details of the algorithm; a complete description and proof of its optimality in satisfying Equation 8 is given in [26].

6.3 Offset polylines

It is natural to consider roads as 2-dimensional surfaces in 3-dimensional space. The arc road formulation above describes 1-dimensional curves — we can define such a 2-dimensional surface by ‘offsetting’ the curve in a given direction. That is, we consider a new curve P'_S is ‘offset’ from P_S to one side by a signed distance d ; see Fig. 8. P'_S that has the property that at every point, the nearest point on P_S is exactly distance d away. We have used the convention that $d > 0$ refers to a ‘right’ offset (the lower blue line in Fig. 8) and $d < 0$ to a ‘left’ offset (the upper blue line in the same figure). The new arcs i

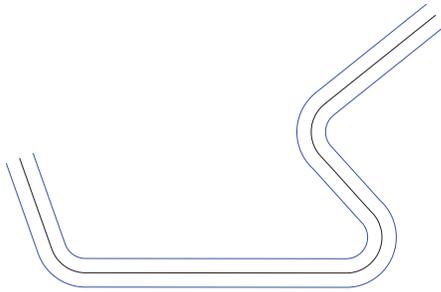


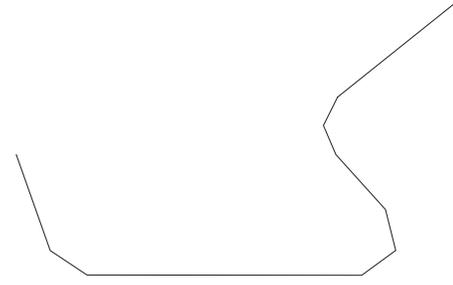
Fig. 8. A ‘fatted’ arc road; the original arc road P_S as computed above is drawn in black. The blue lines represent the same shape offset to either side by an equal distance.

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

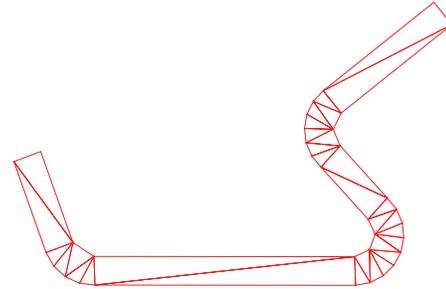
New endpoints p'_0 and p'_{n-1} must be established for this line. A reasonable definition is use the plane of the first and last arcs to choose perpendiculars suitable for placing these offset endpoints, resulting in,

$$p'_0 = p_0 + d(\mathbf{n}_0 \times \mathbf{o}_1) \quad (9)$$

$$p'_{n-1} = p_{n-1} + d(\mathbf{n}_{n-2} \times \mathbf{o}_{n-1}) \quad (10)$$



(a) P^* : A polyline approximation of an arc road P_S



(b) A triangle mesh approximation of a ‘fatted’ arc road P_S

Fig. 9. Discrete approximations of arc roads

6.4 Discrete approximations of arc roads

To visually depict an arc road, we may wish to compute a discrete representation.

6.4.1 Polylines

One way to do this is by approximating the shape by a series of segments, a new polyline P^* . See Figure 9(a). Each arc i must be approximated with a sequence Γ_i of $q_i \in \mathbb{Z}_{>1}$ connected points. Then the sequence of points in P^* is simply:

$$P^* = \left(p_0, \Gamma_1^0, \dots, \Gamma_1^{q_1-1}, \dots, \Gamma_{n-2}^0, \dots, \Gamma_{n-2}^{q_{n-2}-1}, p_{n-1} \right) \quad (11)$$

Each Γ_i is generated by rotating and scaling the frame \mathbf{F}_i (from Equation (6)) of each arc incrementally and translating by the center c_i ,

$$\Gamma_i^j = c_i + r_i \mathbf{F}_i [\cos t^j, \sin t^j, 0]^T, j \in \mathbb{Z}[0, q_i - 1] \quad (12)$$

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

6.4.2 Triangle meshes

A surface representation of an arc road can be easily computed from a pair of offset arc roads (computed as in Sec. 6.3). Given an arc road P_S and two polylines offset from P_S , order the polylines by offset so that we have a ‘left’ polyline P_S^l and a ‘right’ polyline P_S^r . Now we can use any constrained triangulation technique to

compute a planar triangle mesh with P_S^l and P_S^r as the boundaries; see Figure 9(b).

6.5 Road mesh geometry

In Sec. 6.4, we show how to construct a mesh for a pair of given offsets. To create geometry for roads, we need only compute the width of each road — this is easily done by examining the position of each member lane of a given road: the position of the left edge of leftmost member lane forms the left offset, and the position of the right edge of the rightmost member lane forms the right offset.

Our data structure allows for the member lanes of a road to vary along its length; in the case, we simply extract a mesh for each parametric interval along the road that has a constant arrangement of member lanes.

7 RESULTS

We have implemented our method described in this paper. In Figure 5, we show some example road networks created by our technique used in real-time traffic simulation. The models illustrated here were created using GIS data from the Open Street Map website. The first image, Figure 5(a), shows a large road network created within only a few seconds by our method. The network features two small towns in North Carolina, Biscoe and Star, connected by a highway with multiple overpasses and ramps. Figure 5(b) shows a highway overpass that was created using our GIS processing and 3D procedural modeling. Figure 5(c) shows a road network created by our method from GIS data overlaid on top of a satellite image. Notice the degree to which the roads align, despite the fact that the original data is C^0 polylines. Figure 5(d) shows another road representation created by our method and overlaid on a satellite image. This map is from another set of highways with an on-ramp and an off-ramp. Note that the ramps C^1 smoothly connect to the highway, enabling simulated cars to merge onto and off of the highway seamlessly, given the input polyline data not suitable for traffic simulation. This is also shown in the sequence of images in Figure 10, which show an on-ramp and off-ramp to a highway from above at a decreasing height.

As shown in Figure 5(a), 5(c) and 5(d), the seamless alignment between the large-scale road networks in the satellite images and the models created directly from GIS data using our method provides a convincing visual validation of our technique. The functional validity of the road networks created by our algorithm is further demonstrated by the successful real-time traffic simulations on the road networks generated from real-world GIS data, with the resulting visual simulations mapped back to the original satellite images showing no noticeable artifacts in the simulated traffic flows. The video clips for the real-time traffic simulations using the road network models created by our method, as shown in Figure 5(a) to Figure 5(d), can be found at:

<http://gamma.cs.unc.edu/RoadNetwork>.

8 CONCLUSION

We have presented a method for transforming GIS data into a topological and geometric representation suitable for use in traffic simulation. Our geometric representation of roads is visually smooth, including at ramps and intersections. Our method preserves the topological relationships of the GIS road network. We have shown examples of GIS data that have been processed by our method and composed with satellite images. These figures illustrate features of the road networks generated by our method, such as intersection handling and highway ramps, as well as the extensive scale of models that our method can process within a matter of few seconds.

ACKNOWLEDGMENTS

This research is supported in part by Army Research Office, National Science Foundation, and RDECOM.

REFERENCES

- [1] D. Helbing, "Traffic and related self-driven many-particle systems," *Reviews of Modern Physics*, vol. 73, no. 4, pp. 1067–1141, 2001.
- [2] J. Sewall, D. Wilkie, P. Merrell, and M. Lin, "Continuum Traffic Simulation," in *Computer Graphics Forum*, vol. 29, no. 2. Blackwell Publishing, 2010, pp. 439–448, (Proceedings of Eurographics 2010).
- [3] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000.
- [4] U.S. Census Bureau, "TIGER/Line®," 2010. [Online]. Available: <http://www.census.gov/geo/www/tiger/>
- [5] OpenStreetMap community, "OpenStreetMap," 2010. [Online]. Available: <http://www.openstreetmap.org/>
- [6] NAVTEQ, "NAVTEQ," 2010. [Online]. Available: <http://www.navteq.com/>
- [7] J. Mena, "State of the art on automatic road extraction for gis update: a novel classification," *Pattern Recognition Letters*, vol. 24, pp. 3037–3058, 2003.
- [8] J. Park, R. Saleh, and Y. Yeu, "Comprehensive survey of extraction techniques of linear features from remote sensing imagery for updating road spatial databases," in *ASPRS-ACSM Annual Conference and FIG XXII Congress*, 2002.
- [9] A. Fortier, D. Ziou, C. Armenakis, and S. Wang, "Survey of work on road extraction in aerial and satellite images," *Center for Topographic Information Geomatics, Ontario, Canada. Technical Report*, vol. 241, 1999.
- [10] E. Galin, A. Peytavie, N. Maréchal, and E. Guérin, "Procedural Generation of Roads," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 429–438, (Proceedings of Eurographics 2010).
- [11] G. Chen, G. Esch, P. Wonka, P. Mueller, and E. Zhang, "Interactive procedural street modeling," in *SIGGRAPH 2008*. New York, NY, USA: ACM, 2008.
- [12] Procedural Inc., "CityEngine Manual," February 2011. [Online]. Available: <http://www.procedural.com:8080/help/topic/com.procedural.cityengine.help/html/manual/is/create/streets.html>
- [13] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Code*. Academic Press, Inc. Orlando, FL, USA, 1996.
- [14] E. Cohen, R. Riesenfeld, and G. Elber, *Geometric modeling with splines: an introduction*. AK Peters Ltd, 2001.
- [15] L. Piegl and W. Tiller, *The NURBS book*. Springer Verlag, 1997.
- [16] P. Willemsen, J. Kearney, and H. Wang, "Ribbon networks for modeling navigable paths of autonomous agents in virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, pp. 331–342, 2006.

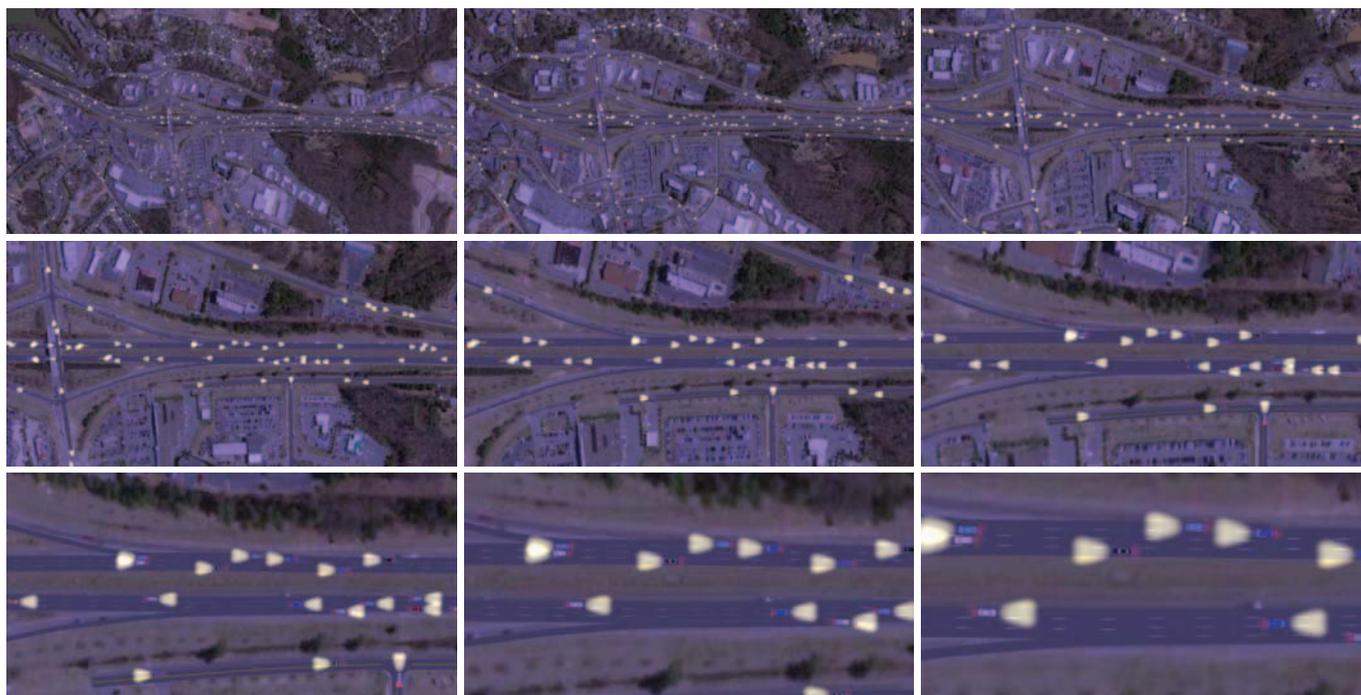
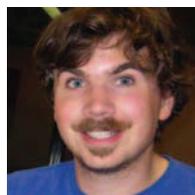


Fig. 10. A series of images showing ramps smoothly connected to a highway, at a variety of scales

- [17] H. Wang, J. Kearney, and K. Atkinson, "Arc-length parameterized spline curves for real-time simulation," in *Proc. 5th International Conference on Curves and Surfaces*. Citeseer, 2002, pp. 387–396.
- [18] J. van den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Diego, CA, November 2007, pp. 4253–4258.
- [19] D. Nieuwenhuisen, A. Kamphuis, M. Mooijekind, and M. Overmars, "Automatic construction of roadmaps for path planning in games," in *Proceedings of the International Conference on Computer Games, Artificial Intelligence, Design and Education*, 2004, pp. 285–292.
- [20] I. Balaban, "An optimal algorithm for finding segments intersections," in *Proceedings of the eleventh annual symposium on Computational geometry*. ACM, 1995, p. 219.
- [21] R. McMaster, "Automated line generalization," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 24, no. 2, pp. 74–111, 1987.
- [22] J. Ware, C. Jones, and N. Thomas, "Automated map generalization with multiple operators: a simulated annealing approach," *International Journal of Geographical Information Science*, vol. 17, no. 8, pp. 743–769, 2003.
- [23] D. Burghardt, "Controlled line smoothing by snakes," *GeoInformatica*, vol. 9, no. 3, pp. 237–252, 2005.
- [24] M. Harrower and M. Bloch, "MapShaper.org: A map generalization web service," *Computer Graphics and Applications, IEEE*, vol. 26, no. 4, pp. 22–27, 2006.
- [25] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
- [26] J. Sewall, "Efficient, scalable traffic and compressible fluid simulations using hyperbolic models," Ph.D. dissertation, University of North Carolina at Chapel Hill, 2010.



David Wilkie received a BS in computer science from Drexel University. He is currently in the doctoral program at the University of North Carolina at Chapel Hill, Department of Computer Science. He is a research assistant in the GAMMA group, and prior to that he was a research assistant in the GICL laboratory at Drexel University. David's research interests include motion planning, geometric modeling, and intelligent transportation systems.



Jason Sewall received his BA in Mathematics and BS in Computer Science from the University of Maine and his MS and PhD from the University of North Carolina at Chapel Hill. While at UNC-Chapel Hill, he worked on physically-based modeling problems under the supervision of Prof. Ming Lin. Jason joined the Parallel Computing Lab at Intel in 2010 as a Research Scientist. His research interests include graphics, physically-based modeling, scientific computations, parallel computing, and algorithms.



Ming C. Lin received her B.S., M.S., and Ph.D. degrees in Electrical Engineering and Computer Science from the University of California, Berkeley. She is current the John R. & Louise S. Parker Distinguished Professor of Computer Science at the University of North Carolina (UNC), Chapel Hill. She has received several honors and awards, including the NSF Young Faculty Career Award in 1995, UNC Hettleman Award for Scholarly Achievements in 2002, Beverly W. Long Distinguished Term Professor 2007-2010,

Carolina Women's Center Faculty Scholar in 2008, Carolina's WOWS Scholar 2009-2011, IEEE VGTC VR Technical Achievement Award 2010, and 8 best paper awards. Her research interests include computer graphics, visualization, robotics, and human-computer interaction in general with a focus on physically-based modeling, real-time interaction with virtual environments, geometric modeling, and GPU-Computing. She has (co-) authored more than 220 refereed publications and co-edited/authored four books. She is the Editor-in-Chief of IEEE Transactions on Visualization and Computer Graphics and a member of six editorial boards of scientific journals. She has served as a program/paper committee member for over 100 leading conferences and co-chaired over 20 international conferences and workshops. She also has served on the Steering Committees of international conferences and advisory boards for government agencies and industry.