EUROGRAPHICS 2014 / B. Lévy and J. Kautz
(Guest Editors)

Volume 33 (2014), Number 2

Designing Large-Scale Interactive Traffic Animations for Urban Modeling

I. Garcia-Dorado   D. G. Aliaga   S. V. Ukkusuri

Purdue University, IN, USA

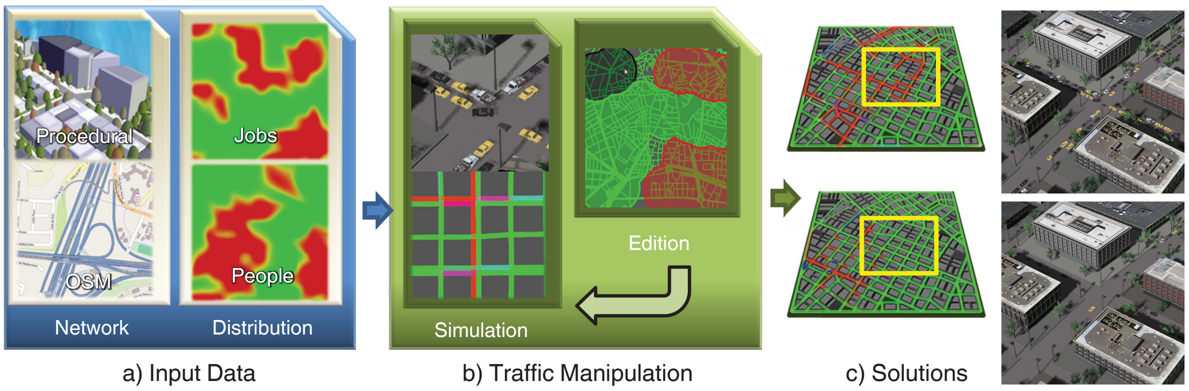

Figure 1: ***Pipeline.*** *Our approach enables a designer to specify a vehicular traffic behavior and the system will compute what realistic 3D urban model yields that behavior. a) The user creates/load a road network and defines the job and people distribution; b) inputs are used to simulate traffic and the user draws a desired new traffic behavior (or traffic optimization). Our system iteratively simulates and alters the model so as to find c) solutions that meet the desired goals and/or costs.*

Abstract

Designing and optimizing traffic behavior and animation is a challenging problem of interest to virtual environment content generation and to urban planning and design. While some traffic simulation methods have appeared in computer graphics, most related systems focus on the design of buildings, roads, or cities but without explicitly considering urban traffic. To our knowledge, our work provides the first interactive approach which enables a designer to specify a desired vehicular traffic behavior (e.g., road occupancy, travel time, emissions, etc.) and the system will automatically compute what realistic 3D urban model (e.g., an interconnected network of roads, parcels, and buildings) yields the specified behavior. Our system both altered and improved traffic behavior in novel procedurally-generated cities and in road networks of existing cities. Our urban models contain up to 360 km of roads, 300,000 vehicles, and typically cover four hours of simulated peak traffic time. The typical editing session time to "paint" a new traffic pattern and to compute the new/changed urban model is two to five minutes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—I.3.6 [Computer Graphics]: Methodology and Techniques—

1. Introduction

Interactive modeling of urban spaces, with high realism and accurate behavior, is a fundamental challenge in computer graphics. Vehicular traffic is a ubiquitous dynamic activity in real-world cities which makes its simulation a necessity for realistic interactive urban environments. Moreover, with more than half of the world population living in cities, there is considerable interest in large-scale traffic simulation, design, and visualization. Virtual environment applications with a growing need for realistic vehicle traffic include virtual tourism, games and films, navigation services, traffic monitoring, eco-routing, and urban planning and re-design.

Considerable effort has been devoted to urban modeling. Vanegas et al. [VAW*10] and Musialski et al. [MWA*13] provide surveys of urban procedural modeling and urban 3D reconstruction. Schreckenberg and Sharma [SS11] and Pelechano et al. [PAB08] provide excellent crowd simulation and animation surveys. Traffic simulation and animation for computer graphics has received some recent attention (e.g., [GVK06,SvdBLM11,SWML10,SWL11,WSL13]) but has also only been investigated in a forward fashion (i.e., simulate traffic for a given road network).

The challenges for our work are i) simulating realistic traffic flows at interactive rates, and ii) controlling traffic in an easy and intuitive manner. Traffic is well recognized to be difficult to simulate and control due to its highly nonlinear behavior, inherent complexity, and emergent behavior. For example, a local change to the network might have an adverse effect elsewhere in the network (e.g., blocking an important avenue in one neighborhood might cause a long traffic delay in another part of the city due to traffic redirection, or increasing the speed limit of a road segment might seem like it will improve travel time, but it might in fact attract more vehicles and ultimately slow down transit in the area). Trial-and-error and keyframe-based control techniques might work for a small number of intersections but not for interactively designing large-scale traffic animations.

We present a novel methodology for automatically creating a 3D urban model (e.g., an interconnected network of roads, parcels, and buildings) that exhibits a desired and realistic vehicular traffic behavior (Figure 1). Our method provides an interactive virtual paintbrush tool whereby the user can specify i) the desired traffic for a new urban model (e.g., for games and films, navigation services, or virtual environments in general), or ii) can improve or alter traffic in a provided urban model to assist traffic planners in obtaining desired values for standard metrics such as road occupancy (i.e., percentage of the segment that is occupied), travel time, or CO emission level. Our solution includes a novel traffic microsimulation engine and an algorithm to manipulate traffic behavior. To the best of our knowledge, our framework is the *first interactive method to automatically generate a realistic 3D urban model that yields a specified traffic behavior*.

Our traffic microsimulation engine yields both the detailed per-vehicle data needed for traffic animation and the fast performance needed for our design strategy. The system we create achieves its significant speedup by extending microsimulation with a novel traffic atlas concept, a new approximate solution to a time-dependent shortest path problem, and an efficient adaptation of car-following, lane changing, and gap-acceptance models.

Our traffic manipulation strategy explores which set of urban model changes brings the simulated traffic behavior closer to the interactively specified behavior. While there are several ways to explore such a large solution space, our method is based on Markov Chain Monte Carlo (MCMC) and inspired by recent inverse procedural modeling of 2D input [SBM*10], 3D shapes [TLL*11], or cities [VGDA*12].

Our framework has created 3D urban models containing up to 360 km of roads, 300,000 vehicles, and typically spanning four hours of simulated time. Our system is estimated to be 77 to 81 times faster for large road networks as compared to other traffic simulators (e.g., [SUM, SWML10]), and it provides per-vehicle (disaggregated) information. Our system has altered traffic behavior in novel cities as well as in road networks from OpenStreetMap [OSM] for Boston, Manhattan, and Madrid. A typical total interactive design time is two to five minutes.

Our main contributions include

- a fast traffic microsimulation engine including per-vehicle simulation, lane changing, car following, and intersection modeling (i.e., traffic lights, stop signs); our engine exploits our traffic atlas concept and our efficient approximation of time-dependent shortest path;
- a traffic manipulation framework that enables specifying a desired traffic behavior (and thus animation) and automatically generating the underlying urban model including road, parcel, and building geometries; and
- a road optimization method that reduces travel time or CO emission, for example, with a minimum change (i.e., cost) to the original road network.

2. Previous Work

2.1. Procedural Road Modeling

Procedural road modeling has received significant interest (e.g., [VGDA*12, CEW*08, GPGB11]). However traffic behavior is not simulated during design. Weber et al. [WMWG09] simulate traffic flow to estimate road widths and to improve a land use simulation. Their simulation only performs a stochastic sampling of a subset of all trips and estimates road occupancy. A detailed traffic microsimulation is not used.

2.2. Traffic Simulation

Traffic simulation/flow models can be categorized into the following three broad categories.

- *Microscopic models* simulate traffic interaction at an individual vehicle level including quantifying driver behavior, vehicle spacing, headway, speeds and lane changing (e.g., [MAT]SIM, MITSIM [YK96], [SUM]O, and [VIS]SUM). The key drawbacks are (i) agent-level calibration and (ii) large computational time.
- *Macroscopic models* provide aggregated representations of traffic (e.g., [LMS07, Ngo11]). Traffic is modeled as a continuum based on hydrodynamic kinematic wave equations using the fundamental relationships of speed, flow, and occupancy [Pay71]. These models are faster but lack realism (and data) of individual vehicle behavior.
- *Mesoscopic models* simulate individual vehicles, but vehicle movement (or flow) is governed by macroscopic relationships rather than detailed per-car models.

All these simulation models are "forward generating" in the sense that traffic is simulated for a given road network. In contrast, we seek an interactive method which finds city-scale urban geometry yielding a desired traffic behavior.

2.3. Traffic Design and Animation

Network (re)design to satisfy a set of traffic objectives is studied in the transportation community (e.g., [SUM09]). Often solutions are formulated as bi-level leader-follower games typically known as Stackelberg games [SBUH11]. However, it is well known that these types of problems are

NP-hard. Approximations are typically analytical optimizations at non real-time speeds. Recently, within the computer graphics community, Go et al. [GVK06] animate vehicles using control and motion planning. Van den Berg et al. [SvdBLM11] reconstruct and visualize continuous traffic flows from discrete data provided by traffic sensors. Sewall et al. [SWML10] extend a continuum based (i.e., macroscopic) approach to include some lane and speed limit changes. Sewall et al. [SWL11] extend the method to include microsimulation in selected parts of the road network while obtaining roughly similar performance. Similar to Section 2.3, all these methods are also forward-generating. In contrast, our method supports our novel traffic design concept. Further, Sewall et al. [SWML10] claims a performance of about 100x over real time; our method yields a microsimulation at 9000x improvement over real-time.

3. Overview

As an overview, we describe the initial road network, summarize our interactive traffic editing system, and highlight how the road network can be automatically changed.

3.1. Initial Road Network

Our method uses a well-connected initial road network with lanes and intersections obtained from OSM or from an urban procedural modeling engine. In both cases, we store the road network as a directed graph. Each edge (i.e., road segment) stores the number of lanes, the lane directionalities, road geometry, to be computed probability, road width, and speed limit. Each node (i.e., intersection) stores whether there is a traffic signal, stop sign, or a right-of-way. Our system's urban procedural modeling engine is an enhanced version of the Vanegas et al. [VGDA*12] that was inspired by CityEngine [ESR]. Our engine uses high-level procedural parameters with a focus on road geometry. Our engine takes as input a terrain, a distribution of people and jobs, and several stylistic parameters. Then, it generates a road network (of highways, avenues, and collector roads), subdivides areas enclosed by roads into parcels, and places a building structure inside each parcel based on whether it is residential (i.e., only people are located there), commercial (i.e., only jobs are located there), or mixed-use. The building is one of several predefined types.

3.2. People and Jobs Distribution

Our system requires storing and modifying the distribution of people and jobs over the urban area. While we could store people/jobs as spatially-located agents, it would be hard (and inefficient) to relocate them during traffic manipulation. Instead, we store them as Gaussian probability distributions over a regular grid of cells – typically 200x200 meter cells. To sample the people or jobs distribution, we first randomly choose a cell proportional to its Gaussian probability distribution. Then, we find values within the cell by performing a 2D sampling of its Gaussian distribution.

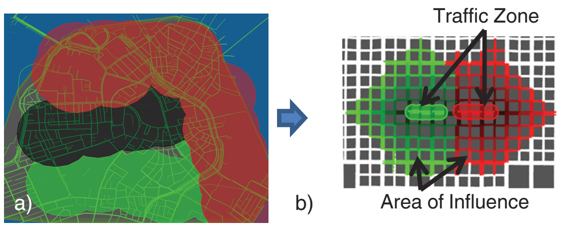

Figure 2: ***Traffic Zones.*** *a) The user can "paint" one or more traffic zones to specify a traffic behavior. b) Each traffic zone has an area of influence that may be altered during the traffic manipulation algorithm.*

3.3. Traffic Zone Specification

The designer uses a virtual paint brush to specify a set of K *traffic zones* $Z = \{Z_1, Z_2, \ldots, Z_K\}$ each with desired constraints or objectives for its contained traffic (Figure 2). Each traffic zone is modeled as a union and/or difference of user drawn circles (or polygons). Further, individual streets can be added/removed from traffic zones. The zone can be specified as constrained (black) or unconstrained. If unconstrained, the paint color indicates whether the traffic is to increase (red) or decrease (green).

3.4. Urban Model Solutions

Given an urban model, road network, and traffic zones, our traffic manipulation strategy explores a set of road network and model changes to the initial urban model so that it better satisfies the specified traffic behavior. Potential changes are selected from a set of topology-preserving (i.e., the connectivity of the roads does not change) and topology-changing (i.e., roads are added or deleted) road network change types. After a change, our traffic simulator is executed to predict the effect of said changes. Traffic editing can be done for:

- *design* - useful to devise a desired traffic behavior for procedural models or content generation; or
- *optimization* - useful to minimize cost once a goal traffic behavior has been sufficiently achieved.

4. Traffic Simulation

Traffic is simulated over many small time steps (e.g., $\Delta t \in [0.1, 0.5]$ secs). All our traffic-related models and parameters stem from well-known traffic simulation literature and are considered important in practice. First, our simulator executes per-vehicle trip planning. In each simulation step, a car's trip plan is used to update its position, velocity, and acceleration while inspecting the network, others cars, and intersections. Finally, we compute traffic performance metrics.

4.1. Trip Planning

If not provided as input, we augment the people/jobs distribution of the urban model with individualized trip plans consisting of a randomized schedule and desired route(s). Each person is assigned a home location and a job location. Starting at different times during the simulation period (e.g., 6 to 10 am), the vehicle departs its home location to reach the employment location and returns at a later time. To simulate trip chaining, for some vehicles additional random destinations are added during the simulation period. To compute each vehicle's route, we approximately solve a time-dependent shortest path (TDSP) problem. Normally, solving the TDSP problem requires computing the effective travel time of each lane segment during all time steps within the simulated period. The effective travel time is used to update the shortest paths for all vehicles. This process repeats until the effective travel time does not significantly change (i.e., equilibrium). This methodology is very time consuming and the update time is not bounded. In contrast, our approximate solution uses average travel time per lane (instead of a sampling of travel times during the simulated period), route-source grouping (Section 4.1.1), and progressively-decaying route updating (Section 4.1.2) which bounds the number of passes. The result is a fast approximate solution to TDSP (Section 6).

4.1.1. Route-Source Grouping

The first simplification assigns each vehicle to the intersection closest to their source position (e.g., home, office). Dijsktra's shortest path algorithm computes the shortest path from a vertex to all other vertices. Thus, rather than executing Dijkstra for each vehicle during each update pass, we execute it only for each graph vertex (i.e., intersection) having at least one vehicle assigned to it. Hence, the number of executions of Dijsktra's algorithm is proportional to the number of vertices times the number of update passes. In practice, it is even less since only a fraction of the vertices correspond to home or job locations. As an example, for a graph with 3000 intersections and 200,000 vehicles, we update all vehicles' shortest paths in only 0.09 milliseconds.

4.1.2. Progressively-Decaying Route Updating

The second simplification progressively reduces the number of vehicles that are updated during each pass of shortest path re-computation. After running the initial pass, although all vehicles are used in all passes we gradually reduce the percentage of the vehicles whose shortest paths are updated (e.g., 75%, 50%, 25%, 12%, etc.). This decay reflects that all vehicles do not necessarily follow the "best" shortest path and it also typically bounds the number of passes to 5 or less with reasonable trip planning.

4.2. Traffic Atlas

A given road network is converted into a compact 2D *traffic atlas* representing sampled road locations and an array of intersection records (Figure 3). One option to maintain the per-vehicle data for microsimulation is to maintain a set of lists. However, this requires sorting and queuing operations that are time consuming for crowded roads. Since vehicles can be treated fairly independently, efficient memory access and easy separation of tasks is crucial for a parallelized implementation, our approach is to store vehicle data in a compact

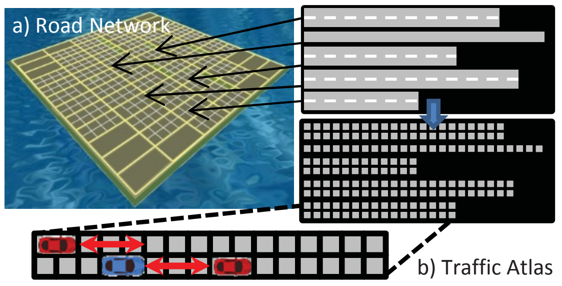

Figure 3: ***Traffic Atlas.*** *a) Each road lane is a row in the traffic atlas; b) the traffic atlas (top right) is sampled into delta segments (bottom right).*

but parallel-access friendly traffic atlas. Our traffic atlas, akin to a texture atlas, compactly stores sampled road segments. Each road segment (i.e., graph edge) is stored as a set of rows (one per lane) of bytes where each byte represents t_m meters of a lane. Each byte can at most be occupied by one vehicle. The byte stores the car's speed (in m/s) times three (e.g., 55 mph or 88 kmh corresponds to 24.4 meters/second and to the value 73). In practice, we found this quantization to be adequate. Intersections (i.e., graph nodes) are much sparser then sampled road segments and are stored separately. This data structure can efficiently store very large road networks (e.g., using $t_m = 1$ we can store 250,000 km of 4 lane roads in 1 GB of memory - this is roughly the length of all roads in Germany or Spain).

4.3. Simulation Steps

Given a set of per-vehicle trip plans and current traffic condition (traffic atlas), we update each vehicle's acceleration value and position, perform mandatory or discretionary lane changes, and update travel times.

4.3.1. Car-Following Model

Our simulation model is based on a discretized car-following principle: the current speed and acceleration depends on the distance to the following car (i.e., the next, or following, car in the direction of the current lane). As in Sewall et al. [SWL11], we use the Intelligent Driver Model [THH00]. The acceleration/breaking function has two main terms:

- *free flow*: the vehicle's acceleration to reach its desired speed in absence of others (i.e., the speed limit), and
- *following-car closeness*: this term represents the deceleration when it comes too close to the one in front of it.

Altogether, we can write a vehicle's acceleration as

$$\dot{v} = a\left(1-(v/v_o)^4-(s^*(v,\Delta v)/s)^2\right) \tag{1}$$

where a is the acceleration ability of the car, v is the current speed of the car, v_0 is the desired speed, s is the distance gap to the following car, and s^* is

$$s^*(v,\Delta v) = s_0 + Tv + v\Delta v/2\sqrt{ab} \tag{2}$$

where s_0 is the minimum following distance, T is the desired time headway, and b is a comfortable braking deceleration.

We follow the range guidelines from Treiber et al. [THH00] to assign random values to a, b, and T.

4.3.2. Lane-Changing Model

This model predicts when and where a vehicle will make a lane change based on two fundamental behaviors:

- *mandatory behavior*: the necessity of changing the lane in order to reach an exit or turn at an intersection.
- *discretionary behavior*: the desire to change lane in order to increase speed and bypass a vehicle.

Our model is based on a combination of the approaches of Yang and Koutsopoulos [YK96] and of Choudhury et al. [CTBA07]. A vehicle always enters a new road in discretionary behavior and with an exponential probability might change to a mandatory behavior. The probability to enter a mandatory behavior can be written as:

$$m_i = \begin{cases} e^{-(x_i-x_0)^2} & x_i > x_0 \\ 1 & x_i \leq x_0 \end{cases} \tag{3}$$

where m_i is the probability of vehicle i to be changed to mandatory behavior, x_i is the current distance from the vehicle to an exit/intersection, and x_0 is the distance of a critical location to the exit/intersection (e.g., last exit warning).

4.3.3. Gap-Acceptance Model

Once the vehicle has decided to change lanes, the maneuver is performed if the lead and lag gaps are acceptable. Using car speeds and distances, we compute the *critical lead gap* g^D_{ia} (i.e., minimum distance to the following car at which a lane change can be performed) and the *critical lag gap* g^D_{bi} (i.e., minimum distance to the lagging car at which a lane change can be performed):

$$g^D_{ia} = \max g^D_a, g^D_a + \alpha^D_{a1} v_i + \alpha^D_{a2}(v_i - v_a) + \varepsilon_{ia} \tag{4}$$

$$g^D_{bi} = \max g^D_b, g^D_b + \alpha^D_{b1} v_b + \alpha^D_{b2}(v_b - v_i) + \varepsilon_{bi} \tag{5}$$

where g^D_a is the desired lead gap for a lane change, g^D_b is the desired lag gap for a lane change, α is a system parameter (typically $\alpha = [0.05, 0.40]$) that controls the gap based on speed, v_i is the speed of the vehicle, v_a is the speed of the lead vehicle, v_b is the speed of the lag vehicle, and ε_{ia} and ε_{bi} are terms that add randomness to the behavior. We use typical values for these parameters obtained from [CTBA07]. If a vehicle's actual lead and desired gaps are within the critical and desired range and the lane changing mode is discretionary, a lane change occurs.

4.3.4. Simulation Update

During each simulation step, the traffic atlas, traffic lights, and vehicle information are updated. To avoid synchronization overhead and dependency on update execution order, we swap between two atlases by simply exchanging atlas pointers. Traffic lights are updated using round robin logic or using light phasings and timings based on real-world data. The simulator checks if a vehicle is waiting to start a new trip. If there is room in the first segment of the vehicle's route, the vehicle is positioned on the traffic atlas with $v = 0$. The position and velocity of this now active vehicle, and all other active vehicles, is updated.

For each active vehicle, the simulator performs several update steps. First, it checks the distance to the following car. If none is found on the current lane, the simulator inspects the traffic signaling at the following intersection. If the traffic light is red or it's the car's turn to stop at a stop sign, it corresponds to there being a following car at the intersection with $v = 0$. If the traffic light is green or it's not the car's turn to stop at the stop sign, our method finds the following car on the next lane segment of the vehicle's route. At a stop sign, the intersection tells the car when it can pass through. Second, given the distance to the following car (if any), the vehicle's acceleration, velocity, position and relevant traffic indicators (Section 4.4) are updated. Third, the simulator considers vehicle lane changes (Section 4.3.2). If a lane change is desired, the gap-acceptance model (Section 4.3.3) is evaluated. Fourth, if an intersection is reached and it is not the destination, then the vehicle attempts to move to the next lane segment. If there is no space in the lane, it remains at the current location (i.e., $v = 0$).

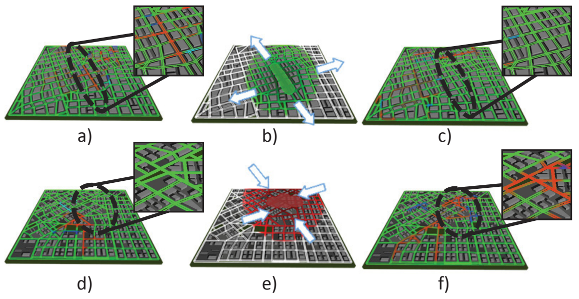

Figure 4: ***Lane-Changing Operations.*** *a) Initial network occupancy. b) The user "paints green" so as to reduce traffic. c) After MCMC, new road occupancy values closely match the painted traffic behavior. d-f) An analogous process but for increasing lane traffic.*

4.4. Traffic Indicators

Our simulator calculates several indicators which are important for decision-making and policy setting. We measure metrics such as average travel time, distance travelled, and emissions. For example, to report total or per-vehicle CO emission, our system uses the following equation [AU12]

$$\Omega = -0.064 + 0.0056 v_m + 0.00026(v_m - 50)^2 \tag{6}$$

where Ω is the emission rate (in grams of CO per vehicle per second) and v_m is speed of the vehicle in mph.

5. Traffic Manipulation

We describe our traffic manipulation methodology and its use of the traffic simulation engine. During our MCMC process, the probability of a road network change is computed as a product of a change-type probability and a location probability. Each proposed change is evaluated for acceptance and the search continues until satisfying an objective function (or reaching a maximum "cost").

When so desired, the aforementioned traffic painting can be used to constrain areas of the road network and ensure only local road network changes. However, due to the global and complex nature of traffic flow, a closer to optimum solution for a locally specified objective might actually involve a global set of changes – thus stability can be enforced but is not always beneficial.

5.1. Change-Type Probability

The following topology-preserving and topology-changing operations worked well with our examples. For a lane i or person/job grid cell k, we define four change types and their probabilities.

- *Lane direction change*. With probability d_i this topology-preserving change alters the directionality of a road segment's lane (Figure 4). To decrease traffic in a zone, we switch the lane direction to direct vehicles away from the zone middle. Conversely, to increase traffic, we swap the lane direction so that vehicles pour into the zone.
- *Number of lanes change*. With probability n_i this topology-preserving change alters the number of lanes per direction of a road segment. To decrease traffic in the zone, a lane is added to a road segment inside or outgoing from the traffic zone. Conversely, to increase traffic a lane is removed from a road segment.
- *People change*. With probability p_k this topology-changing operation relocates the people within the urban space and potentially triggers a significantly different procedural urban model (Figure 5). To reduce traffic, we need to know the cells that generated people who passed through the traffic zone and then relocate their distribution energy to elsewhere in the urban space. To increase traffic, we move the distribution energy from a random area to the most common people distribution area with vehicles passing through the traffic zone. Whenever possible, we transfer people from a zone wishing to decrease traffic to another zone seeking to increase traffic.
- *Job change*. With probability j_k this topology-changing operation is analogous to the previous category but moves job distributions instead.

These four types of changes may alter the cityscape. In particular, people/job distribution may alter buildings (e.g., dense areas will have smaller setbacks and bigger buildings) and the road network (e.g., MCMC adds roads to fill a dense area – even to OSM networks).

To avoid excessively altering lane directions/number of lanes, changes are done to a neighborhood of lanes. Further, to find origin-destination link pairs that pass through the traffic zone for people/jobs changes, we create a hash table to lookup the vehicles used by each edge. Then, we create a histogram of origin-destination link pairs of those roads and find which were the most common edges that made the vehicles cross the traffic zone.

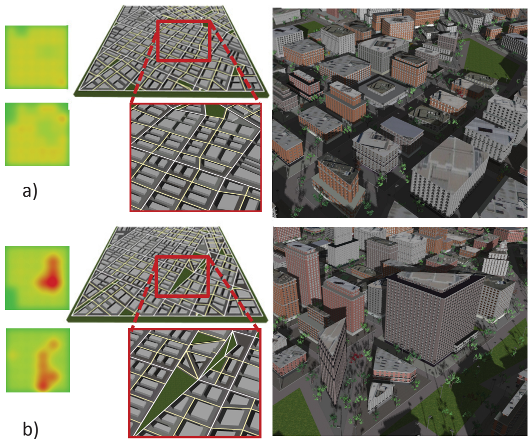

Figure 5: ***People/Jobs Changes.*** *Such changes can impact network topology. a) Input city. b) City with changed roads and buildings satisfying new traffic.*

5.2. Location Probability

The probability of changing all lanes, people, and jobs inside a zone is related to their location relative to the zone. The location probability x_i of a lane changing direction, or the corresponding road segment having a number of lanes change, is inversely proportional to the distance from the lane's midpoint to the exterior border of the traffic zone. Our function is setup so that a lane inside a zone has probability one and those outside the zone are computed using a Gaussian weighting function; i.e.,

$$x_i = \begin{cases} \frac{1}{w_Z}\phi\frac{t_i}{w_Z} & if\, outside\, zone \\ 1 & if\, inside\, zone \end{cases} \tag{7}$$

where t_i is the distance of lane i to the zone perimeter and w_Z is the width of a zone's area of influence on the surrounding urban space. In our system, t_i is computed as number of graph edges links (e.g., breadth first search).

The location probability q_k of a people change, or job change, is proportional to its usage as an origin, or destination, for a route passing through the traffic zone. Thus, its location probability does not depend on whether the cell is near the traffic zone but on whether the cell is used by a route that passes through, or near, the traffic zone. The aforementioned histogram of origin-destination pairs is sorted and top candidates are given the highest probability. Thus,

$$q_k = \begin{cases} \frac{1}{w_S}\phi\frac{s_k}{w_Z} & if\, outside\, zone \\ 1 & if\, inside\, zone \end{cases} \tag{8}$$

where s_k is the histogram count for cell k and w_S is a normalization constant. In order to instill some additional randomness in the solution exploration process for people and job changes, we add an additional probability r. The term r mimics behaviors in a city where people, or jobs, periodically change due to a variety of reasons. In predictive agent-based urban simulations (e.g., [Wad02]), such periodic random change of people or jobs is modeled by allowing a small percentage (e.g., 10%) of changes per year.

5.3. MCMC Search Strategies

Our search strategy to manipulate traffic is to alternate between updating the traffic values using our traffic simulator and performing an MCMC optimization step until the specified traffic behavior is met. Using the aforementioned change-type and location probabilities, our search process starts with an initial urban model configuration X_0 and seeks an improved urban model X_n that better satisfies an objective function; i.e., $F(X_n, Z_1, Z_2, \dots, Z_K) \to 0$. Our MCMC optimization uses n_T simultaneous threads and each thread performs up to n state changes. Each thread starts using a different random seed and a different temperature β with values that were empirically found to range from 4 to 256. Further, our optimization process includes two objective functions: goal-driven and cost-driven.

A candidate state change $\widehat{Y}$ is obtained by sampling over a subset of the possible lanes. The probability is equal to the product of the corresponding change-type probability and location probability: for each lane $d_i x_i$ or $n_i x_i$; whereas for each grid cell $p_k(q_k+r)$ or $j_k(q_k+r)$. Starting with the lanes nearest to or inside a zone and the people/job cells highest in the origin-destination histogram, we randomly select lane direction changes, number of lanes changes, people relocation, and/or jobs relocation. We grow the subset until H changes occur amongst all change types. The collection of lanes, people, and jobs changes define a candidate state change $\widehat{Y}$.

The candidate $\widehat{Y}$ is then evaluated for acceptance. The search process executes our traffic simulation and then re-evaluates the objective function. Our Metropolis ratio computes the acceptance probability of going from current state X_c for $c \in [1,n]$ to the candidate state $\widehat{Y}$ as

$$a\left(X_c \to \widehat{Y}\right) = \min\left\{1, \frac{e^{-\beta F(\widehat{Y},\{Z_1,Z_2,\dots,Z_K\})}}{e^{-\beta F(X_C,\{Z_1,Z_2,\dots,Z_K\})}}\right\} \quad (9)$$

where β is the thread's temperature. We use the Metropolis ratio because our probability distributions are Gaussians and our proposal function is symmetric. The rejection probability is $\left(1 - a(X_c \to \widehat{Y})\right)$; i.e., $X_{c+1} = X_c$. Once $c = n$, we have reached the final state and the best solution over all threads and states is selected.

5.3.1. Goal-Driven Optimization

Our goal-driven optimization drives the simulated traffic behavior to the road occupancy specified by the traffic zone set. Road segment s's occupancy is defined as

$$u_s = \frac{c_s}{L_s n_s / b_v} \quad (10)$$

where c_s is the average of the number of vehicles on the road segment over the simulated period and the denominator is the maximum number of vehicles per road segment (also known as the "jam density"). The maximum vehicles per road segment is computed using L_s= length of the road segment, n_s= number of lanes in the road segment, and b_v= minimum distance between vehicles (e.g., 5m). The objective function can be then written

$$F(X_n, \{Z_1, Z_2, \dots, Z_K\}) = \sum_S \|u_s - \widehat{u_s}\| \quad (11)$$

where S is the set of roads inside the traffic zones and $\widehat{u_s}$ is the desired per road segment occupancy value.

5.3.2. Cost-Driven Optimization

Our second strategy minimizes the cost of the network changes once traffic behavior is sufficiently similar to the one specified by the traffic zones. For traffic design, this enables finding a low-cost solution satisfying the behavior. To measure cost C, we quantify the cost of all changes:

$$C = (w_L/N_L)\sum_{N_L}\|e_i - \bar{e_i}\| + (w_S/N_S)\sum_{N_S}\|n_s - \bar{n_s}\| + (w_P/N_P)\sum_{N_P}\|P - \bar{P}\| + (w_J/N_J)\sum_{N_J}\|J - \bar{J}\| \quad (12)$$

where (w_L, w_S, w_P, w_J) and (N_L, N_S, N_P, N_J) are the cost weights and number of entries for lane direction changes, number of lane changes, people moving changes, and jobs moving changes, respectively; $e_i = \{0,1\}$ refers to the current lane direction and $\bar{e_i}$ is the initial lane direction; similarly for number of lanes n_s, 2D distribution of people P, and 2D distribution of jobs J.

A proposed state change is considered to be accepted only if Equation (11) is beneath a threshold value. Then $F(X_n, \{Z_1, Z_2, \dots, Z_K\}) = C$ is used as the objective function. While this might not yield the quickest convergence it works well in practice. This score function can defined as the average travel time, CO emissions, or distance to desire traffic pattern.

5.4. Search Initialization

To improve performance when the desired traffic behavior is significantly different from the current one, we make an initial guess of the desired distribution of people and jobs. This task is related to the networking problem of computing a traffic matrix which specifies the amount of traffic between source-destination pairs (i.e., person-job pairs). We adapt the tomo-gravity model [ZRDG03] which is one such well-known method to infer the traffic matrix from the link loads (i.e., road segments). Once such initial locations are computed, we use MCMC optimization to refine the result.

6. Implementation and Results

We used our framework to create and edit a variety of city-scale 3D models obtained from OSM or from our procedural engine having up to 360 km of roads. The simulation is implemented both on the CPU and on the graphics card (CUDA). All rendering is done using our custom shader (see video). All editing is interactive, and results appear while

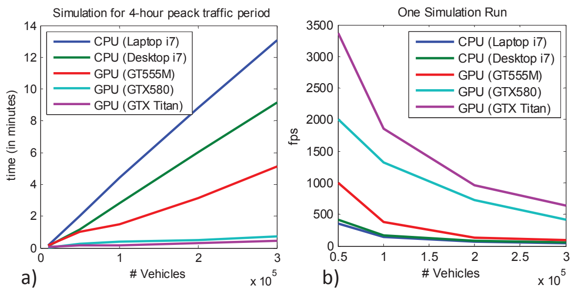

Figure 6: ***Performance.*** *a) Times (in minutes) to perform a 4-hour simulation b) The number of simulation steps per second.*

editing. All example sessions were completed in less than 10 minutes (and typically in less than 5).

Performance. We present a performance summary using up to 300k vehicles on 200 km of roads (with no rendering overhead). Performance is shown for CPU and GPU versions (Figure 6a). A four-hour peak traffic period simulation (i.e., 6am-10am) of 300k cars takes as little as 24 seconds, including trip planning, simulation, and estimating occupancy and indicators. Figure 6b shows that with 300k cars our system can compute 636 simulation steps per second.

We compare performance to those of Sewall et al.'s [SWML10] macrosimulation engine and to SUMO's open source microsimulation engine. SUMO's performance, as well as ours, is dependent on the number of vehicles and on the simulation time. Sewall et al.'s [SWML10] method is claimed to be linearly dependent on road network size and on the simulation time. Their largest reported network has 140,000 cars and only 10 km of roads. Their fastest system (8 cores) was reported as 50 seconds which was 54 times faster than their SUMO performance. For a network with the same number of cars but with 200 km of roads, our approach only takes 13 seconds which is 81 times faster than our SUMO performance. The amount of simulated time is not reported for Sewall et al. [SWML10] thus a direct comparison is hard to make. However, if we linearly extrapolate their performance from 10 km to 200 km of roads, our method would be 77 times faster than Sewall et al. Moreover, our method yields disaggregated per-vehicle data. Finally, we use SUMO to evaluate simulation accuracy (Figure 7). Our method produces occupancy values that, on average, are within 6% of SUMO-computed values.

Analysis. Figure 8 compares the behavior of our two search strategies for 30k cars (Sections 5.3.1 and 5.3.2). Note that due to the non-linear nature of traffic, solution process is not monotonic. We define traffic zones that reduce traffic through one of the main arterial roads. Further, we set the cost function to be the total number of changed lanes. On the left, the goal-oriented optimization minimizes the objective function without any constraints. The found low-score solution requires almost 350 lane changes. On the right, the cost-driven optimization finds solutions with even lower score

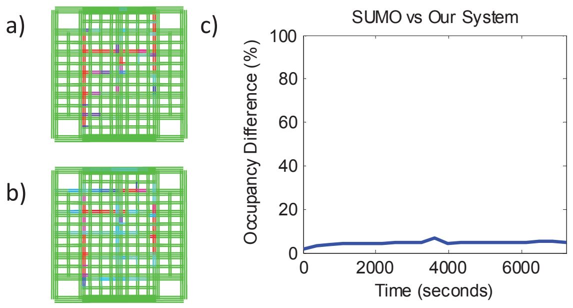

Figure 7: ***Occupancy Comparison - SUMO vs. Our System.*** *Traffic flow measured by our system (a) and SUMO (b); red = complete utilization, green = empty street; c) Error of occupancy measurements over time.*

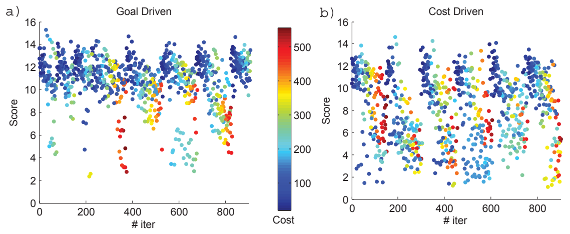

Figure 8: ***Search Strategy Comparison.*** *a) As the iterations pass, goal-driven optimization is able to find solutions of a lower score, but their cost is unbounded. b) In contrast, a cost-driven optimization attempts to reach near the desired score and then it continues but tries to minimize cost.*

and with much lower cost (i.e., just 50 lane changes!). This occurs because the second acceptance formula is more relaxed and thus other state changes can occur.

Example Designs. Figures 9-11 show the results from several example design sessions. Figure 9 demonstrates a local inverse design using downtown Boston (9a) with 1393 streets, 4767 intersections, and a total road length of over 290km. The people and job distribution has been extracted from a GIS. The user first runs the simulation with 110k vehicles. Results show significant traffic crossing the Boston Commonwealth Park and the nearby government buildings. The user then draws two areas to reduce the road occupancy and defines the maximum occupancy to be 20% of those roads. Next, the simulation is run in three different scenarios. In the first scenario, the user only allows changes in lane directions (9f). This restriction sparks changing 48 lane directions to reach the desire behavior. Then, the user only allows changes in job distribution (9g); this could be useful to improve traffic with the cost of a company office reallocation. However, it causes 37% of the jobs to relocate. Finally, the user enables lane direction change and job re-distribution resulting in just 16 lane changes and 3% of the jobs moving (9h). Moreover, we reduce the occupancy to just 9% as indicated by the specified traffic zone behavior.

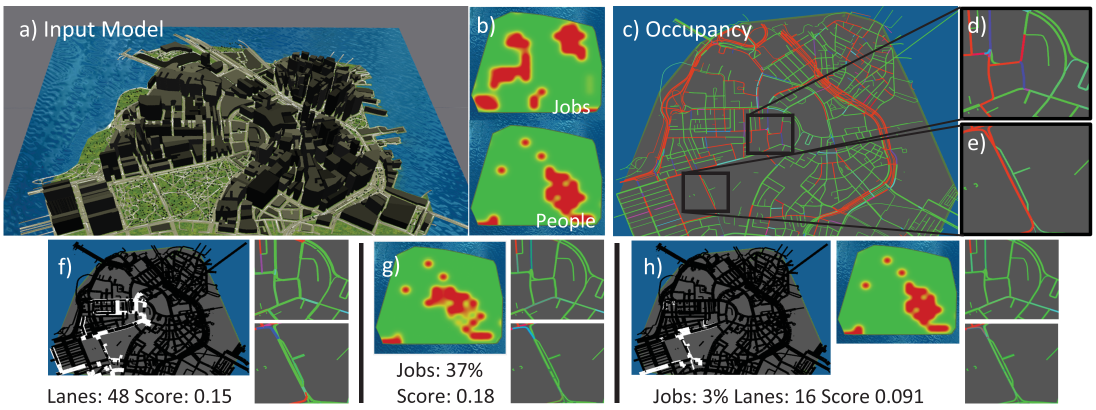

Figure 9: ***Local design.*** *a) Fragment of central Boston. b) Job and people distribution from GIS sources. c) Initial road occupancy as per our traffic simulation. d-e) Close-ups. Three solution options: f) solution by only changing lane directions, g) similar as previous but only jobs distribution changes, and h) using both change types (best solution).*

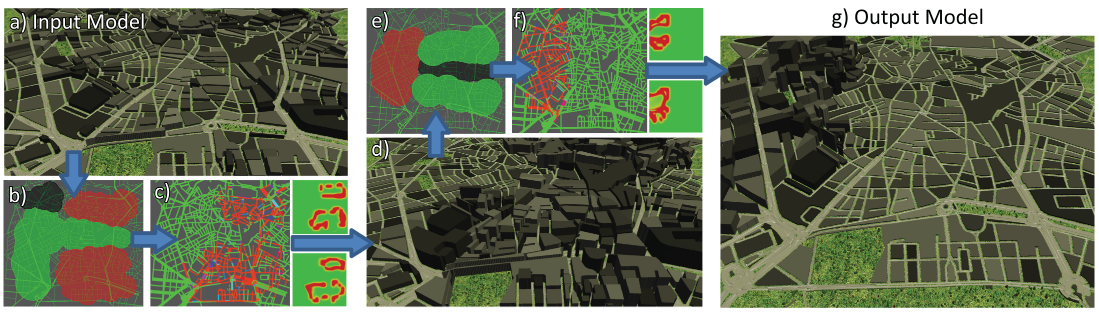

Figure 10: ***Global Design.*** *a) Fragment of Madrid. b) User draws the desired traffic and c-d) our system first uses the tomo-gravity model and then MCMC refines it. e-g) Another editing iteration produces the final output.*

Figure 10 represents an interactive session with a global design objective. The user loads the map of Madrid, Spain from OSM (10a). It contains 2288 streets, 6141 intersections, and more than 330km of roads. The user defines a desired traffic behavior for the entire city and for 220,000 vehicles (10b). Our system finds a solution that produces the given traffic zones with just a few iterations (10c-d). This is possible thanks to the tomo-gravity model (this is the only result that uses Section 5.4 initialization). Afterwards, the user wanted an alternate traffic behavior of the city (10e). The system was also able to find the model that yields such a traffic behavior (10f-g). This session took under 5 minutes.

Figure 11 shows a global traffic optimization. We procedurally generate a fragment of New York City using GIS data, producing over 900 streets, 620 intersections, 25k cars, and 360 km of roads. The initial urban model yields an average travel time of 60 minutes and 1012 gr of CO per person (11a). The user defines three desired travel time values (i.e., 50 minutes, 40 minutes and 30 minutes). Our system finds that it is possible to improve the traffic to a 50 minute travel time by changing 52 lane directions (11b). However, in order to achieve a 40 minute travel time, more radical changes are required –16% of the jobs and 31% people should be relocated (11c). Finally, to achieve a 30 minute average travel time even more changes are required (11d). Each of these sessions took 10 minutes or less.

Limitations. Our system is not without limitations. It is possible to specify a target traffic behavior for which the system is over-constrained and no solution can be found. Also, we cannot specify behaviors for specific people and/or jobs. While we can indicate a desired behavior for a lane, our approach is generally more tuned for medium to large-scale network modifications (i.e., our simulator is not as accurate in traffic prediction for very small network changes).

7. Conclusions

We presented an approach to interactively "paint" a desired vehicular traffic behavior and animation and then the system automatically computes a realistic 3D urban model yielding the specified behavior. We used our system to control traffic behaviors such as road occupancy, travel time, and CO emission. Our framework includes a novel traffic microsimulation approach which yields the high performance needed for our interactive design tool. Our traffic manipulation strategy adapts a MCMC method to explore the solution space by performing a set of topology-preserving and topology-changing road network changes. In addition, we define a

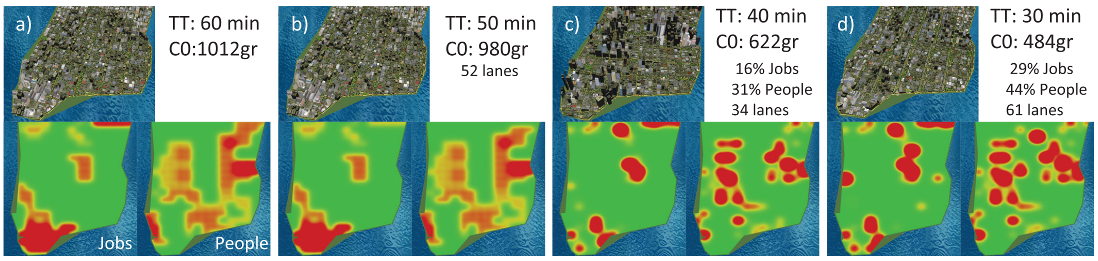

Figure 11: ***Global Optimization.*** *a) Fragment of New York. It has an average travel time (TT) of 60min and CO emission of 1012gr. Our system finds that b) by just changing lanes it is able to achieve the 50min goal. c-d) To reach 40min and 30min, it is necessary to change people, jobs, and lanes.*

novel road network optimization strategy that reduces travel time or CO emission, for example, with a minimum cost.

As future work, there are several avenues. First, we will provide better controls to avoid excessive changes in direction and in number of lanes. Second, we will explore additional topology-preserving changes, such as altering intersection type and speed limit. Third, our simulator will be improved to support more complex traffic lights, on/off ramps, random driver behavior, and more.

Acknowledgment. Partially funded by NSF 1250232, 0964302, and 1302172. We thank NVidia's collaboration.

References

[AU12] Aziz H. M. A., Ukkusuri S. V.: Integration of environmental objectives in a system optimal dynamic traffic assignment model. *Computer-Aided Civil and Infrastructure Engineering 27*, 7 (2012), 494–511. 5

[CEW*08] Chen G., Esch G., Wonka P., Müller P., Zhang E.: Interactive procedural street modeling. *ACM Trans. Graph. 27*, 3 (2008), 103–112. 2

[CTBA07] Choudhury C., Toledo T., Ben-Akiva M.: Modeling cooperative lane changing and forced merging behavior. *Transp. Research Board* (2007). 5

[ESR] ESRI. *esri.com/software/cityengine*. 3

[GPGB11] Galin E., Peytavie A., Guérin E., Benes B.: Authoring Hierarchical Road Networks. *Comput. Graph. Forum 30*, 7 (2011), 2021–2030. 2

[GVK06] Go J., Vu T. D., Kuffner J. J.: Autonomous behaviors for interactive vehicle animations. *Graphical Models 68*, 2 (2006), 90 – 112. 1, 3

[LMS07] Lebacque J., Mammar S., Salem H.: Generic second order traffic flow modeling. *Transp. and Traffic Theory* (2007), 755–776. 2

[MAT] Multi-Agent Transp. Sim. *matsim.org*. 2

[MWA*13] Musialski P., Wonka P., Aliaga D. G., Wimmer M., van Gool L., Purgathofer W.: A survey of urban reconstruction. *Comput. Graph. Forum 32*, 6 (2013), 146–177. 1

[Ngo11] Ngoduy D.: Multiclass first-order traffic model using stochastic fundamental diagrams. *Transportmetrica 7*, 2 (2011), 111–125. 2

[OSM] OSM. *openstreetmap.org*. 2

[PAB08] Pelechano N., Allbeck J. M., Badler N. I.: *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers, 2008. 1

[Pay71] Payne H. J.: *Models of freeway traffic and control*. Simulation Councils, 1971. 2

[SBM*10] Stava O., Benes B., Mech R., Aliaga D., Kristof P.: Inverse procedural modeling by automatic generation of l-systems. *Comput. Graph. Forum 29*, 2 (2010), 665–674. 2

[SBUH11] Stackelberg H. v., Bazin D., Urch L., Hill R. R.: *Market structure and equilibrium*. Springer, 2011. 2

[SS11] Schreckenberg M., Sharma S. D.: *Pedestrian and Evacuation Dynamics*. Springer, 2011. 1

[SUM] Simulation of Urban MObility. *sumo-sim.org*. 2

[SUM09] Sharma S., Ukkusuri S., Mathew T.: Pareto optimal multiobjective optimization for robust transportation network design problem. *Transp. Research Record: J. of the Transp. Research Board 2090* (2009), 95–104. 2

[SvdBLM11] Sewall J., van den Berg J., Lin M. C., Manocha D.: Virtualized traffic: Reconstructing traffic flows from discrete spatiotemporal data. *IEEE Trans. on Vis. and Comput. Graph. 17*, 1 (2011), 26–37. 1, 3

[SWL11] Sewall J., Wilkie D., Lin M. C.: Interactive hybrid simulation of large-scale traffic. *ACM Trans. Graph. 30*, 6 (2011), 135–146. 1, 3, 4

[SWML10] Sewall J., Wilkie D., Merrell P., Lin M. C.: Continuum traffic simulation. *Comput. Graph. Forum 29*, 2 (2010), 439–448. 1, 2, 3, 8

[THH00] Treiber M., Hennecke A., Helbing D.: Congested traffic states in empirical observations and microscopic simulations. *Phys. Rev. E 62* (2000), 1805–1824. 4, 5

[TLL*11] Talton J. O., Lou Y., Lesser S., Duke J., Měch R., Koltun V.: Metropolis procedural modeling. *ACM Trans. Graph. 30*, 2 (2011), 11–24. 2

[VAW*10] Vanegas C., Aliaga D. G., Wonka P., MÃijller P., Waddell P., Watson B.: Modelling the appearance and behaviour of urban spaces. *Comput. Graph. Forum 29*, 1 (2010), 25–42. 1

[VGDA*12] Vanegas C., Garcia-Dorado I., Aliaga D., Benes B., Waddell P.: Inverse design of urban procedural models. *ACM Trans. Graph. 31*, 6 (2012), 168–178. 2, 3

[VIS] PTV VISSIM. *vision-traffic.ptvgroup.com*. 2

[Wad02] Waddell P.: Urbansim: Modeling urban development for land use, transportation and environmental planning. *Journal of American Planning Assoc. 68* (2002), 297–314. 7

[WMWG09] Weber B., Müller P., Wonka P., Gross M.: Interactive geometric simulation of 4d cities. *Comput. Graph. Forum 28*, 2 (2009), 481–492. 2

[WSL13] Wilkie D., Sewall J., Lin M.: Flow reconstruction for data-driven traffic animation. *ACM Trans. Graph. 32*, 4 (2013), 89–98. 1

[YK96] Yang Q., Koutsopoulos H. N.: A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transp. Research Part C 4*, 3 (1996), 113 – 129. 2, 5

[ZRDG03] Zhang Y., Roughan M., Duffield N., Greenberg A.: Fast accurate computation of large-scale ip traffic matrices from link loads. *Proc. of Sigmetrics* (2003), 206–217. 7