Realistic Data-Driven Traffic Flow Animation Using Texture Synthesis

Qianwen Chao, Zhigang Deng, *Senior Member, IEEE*, Jiaping Ren,
Qianqian Ye and Xiaogang Jin, *Member, IEEE*

Abstract—We present a novel data-driven approach to populate virtual road networks with realistic traffic flows. Specifically, given a limited set of vehicle trajectories as the input samples, our approach first synthesizes a large set of vehicle trajectories. By taking the spatio-temporal information of traffic flows as a 2D texture, the generation of new traffic flows can be formulated as a texture synthesis process, which is solved by minimizing a newly developed *traffic texture energy*. The synthesized output captures the spatio-temporal dynamics of the input traffic flows, and the vehicle interactions in it strictly follow traffic rules. After that, we position the synthesized vehicle trajectory data to virtual road networks using a cage-based registration scheme, where a few traffic-specific constraints are enforced to maintain each vehicle's original spatial location and synchronize its motion in concert with its neighboring vehicles. Our approach is intuitive to control and scalable to the complexity of virtual road networks. We validated our approach through many experiments and paired comparison user studies.

Index Terms—traffic flow animation, crowd simulation, data-driven method, texture synthesis.

1 Introduction

In recent years, virtual environments (e.g., cities, road networks, and buildings) have been increasingly used in a variety of applications including virtual reality, digital tourism, urban planning, training, evacuation simulation, and so on. However, virtual environments alone, without populated crowds and traffic flows, would be unrealistic and thus often less useful for many applications.

A number of simulation based methods, including macroscopic (e.g., [1], [2], [3], [4]) and microscopic control models (e.g., [5], [6], [7]), have been developed to generate realistic traffic flows on road networks. However, they suffer from the following two inherent limitations:

- Compared to the ground truth data, the *simulated* traffic flows (Fig. 1(a)) by them are too *regular*, lacking of motion varieties that are essential characteristics of real-world traffic flows (Fig. 1(b)), due to the simplicity of their employed heuristic rules.
- They can only offer the user *indirect* control through the tedious non-trivial tuning of low-level model parameters in a trial-and-error manner, which is unintuitive and inefficient from the user's perspective. The main reason is that, it is technically impossible for the user to know (predict) beforehand what new traffic flows would approximately look like when adjusting those low-level model parameters.

Meanwhile, recently a few data-driven approaches have been proposed to populate virtual environments with human crowds or manipulate multi-character animation in crowds [8], [9], [10], [11]. These methods directly manipulate agents' trajectories to fit both user requirements and the topological constraints of the virtual environment. They provide more direct user control than the aforementioned, purely simulation based methods, because ultimately the synthesized results depend on input examples, which can be easily controlled by users. However, they cannot be straightforwardly extended to populate virtual road networks with realistic traffic flows, due to the following main difference: In a human crowd, typically it is assumed that agents can move along any trajectories with various speeds as long as collision avoidances are handled. By contrast, in addition to collision avoidances, vehicles on the road also need to obey strict traffic rules, such as safe lane changing, acceleration/deceleration, and road regulations. As a result, straightforwardly extending existing data-driven crowd populating approaches to traffic flows would lead to less realistic results.

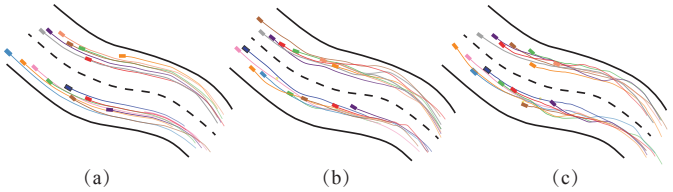


Fig. 1. (a) An example of synthetic two-lanes traffic flows by the well-known Intelligent Driver Model (IDM) based simulation method. (b) Ground-truth traffic flows recorded in real world. (c) The corresponding synthesized traffic flows by our approach. The trajectories of different vehicles are plotted in different colors.

In a sharp departure from widely-used, simulation-based traffic flow generation methods due to their aforementioned limitations, in this paper we present a new data-driven traffic flow simulation approach for virtual road net-

- *Q. Chao is with Computer Science Department, Xidian University, Xi'an, 710071, P. R. China.*
- *J. Ren, Q. Ye and X. Jin are with State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310058, P. R. China.*
- *Z. Deng is with Computer Science Department, University of Houston, Houston, TX, USA.*
- *X. Jin is the corresponding author. E-mail: jin@cad.zju.edu.cn*

(a) (b) (c)

Fig. 2. Examples of traffic flows synthesized by our approach. (a) The synthesized traffic flows on a curvy road. (b) The synthesized traffic flows on a traffic-light controlled road. (c) The synthesized traffic flows on an urban highway network.

works. Specifically, given a limited set of vehicle trajectories as input samples, we first synthesize a large set of vehicle trajectories through the combination of texture synthesis with microscopic traffic behavior rules. Besides minimizing the local repetitions of vehicle behaviors, the synthesized trajectories of vehicles can not only strictly follow traffic rules but also break the limits of vehicle number, lane number, and frame number in the input samples. After that, we fit the synthesized vehicle trajectory data to virtual road networks using an adaptive cage-based registration scheme, where a few traffic-specific constraints are enforced to maintain each vehicle's original spatial location and synchronize its motion in concert with its neighboring vehicles.

Through many experiments as well as paired comparison user studies, we demonstrate that our approach can effectively generate more realistic traffic flows on a variety of virtual road networks than existing simulation based methods. Fig. 2 shows several traffic flow examples generated by our method including a curvy road, a traffic-light controlled road, and a complex urban highway network.

The main contributions of this work can be described as follows.

- It introduces a novel data-driven scheme to generate a large set of new vehicle trajectories through the non-trivial fusion of texture synthesis and traffic behavior rules.
- It introduces a new cage-based traffic flow registration algorithm to precisely snap synthesized vehicle trajectories to various virtual road networks, where traffic-specific constraints are introduced to maintain the vehicles' original spatial location and synchronize their motions with neighboring vehicles.
- To the best of our knowledge, our approach is the first example-based traffic flow populating solution for virtual road networks, which is completely different from widely-used, simulation-based traffic flow generation methods. Besides generating more natural traffic flows than existing simulation-based methods, our method can inspire future research efforts along this new direction to simulate realistic traffics in various complex settings.

2 Related Work

Traffic simulation: Traffic simulation has become an increasingly popular and effective tool for analyzing a wide variety of dynamical problems about traffic flows, which cannot be accurately modeled using analytical methods. Based on the level of simulation details, existing methods can be roughly categorized into two types: *macroscopic* and *microscopic* models.

A macroscopic model describes vehicles' behaviors and interactions at a low level of details, in which a traffic stream is represented by a continuum in terms of characteristics including speed, flow and density [1], [2], [3], [4]. By contrast, a microscopic model treats each vehicle as a discrete autonomous agent with specific governing rules, and it can typically produce vehicle motions at a high level of details [5], [6], [7], [12]. Among existing microscopic models, the Intelligent Driver Model (IDM) [6] for acceleration/deceleration decision and Kesting et al.'s lane-changing model [13] are two notable examples.

Data-driven traffic visualization techniques have received noticeable attentions recently, including the reconstruction of traffic flows from spatio-temporal data acquired by existing in-road sensors [14], [15] and the generation of traffic flows by learning individual-specific driving characteristics [16]. As the latest development of data-driven traffic animation methods, Wilkie et al. [15] estimate the states of input traffic flows and then drive an agent-based traffic simulator to produce traffic animations that statistically match the given sparse traffic conditions.

While the above methods can generate plausible traffic flows to a certain extent, it is difficult to ensure the realism of the generated traffic flows due to the use of heuristically designed driving rules. By contrast, our work generates more vivid traffic animations directly from the real-world data without any simulator involved.

Crowd manipulation: Interactive manipulation techniques for editing multiple character motions has been studied with a varying level of control specifications. Kown et al. [17] use a graph structure to model the spatio-temporal group behavior of pedestrians and employ mesh editing algorithms to manipulate the animation interactively. Kim et al. [18] and Ho et al. [19] extend this manipulation method to handle a wide range of group behaviors with complex interactions. Recently, Kim et al. [11] edit large-scale crowd animations by extending cage-based deformation and as-rigid-as possible deformation, and demonstrated convincing results. However, these methods are only applicable to animations with a limited number of agents and with a relatively short length; they still need non-trivial manual efforts to generate more complex crowd animations at a

larger scale.

Crowd patches: The crowd patches technique [8] partially overcomes the limitations of the above motion editing approaches by automatically assembling periodic pieces of given crowd animations, in order to create infinitely large, yet repetitive, animated crowds. It tiles crowd patches to populate large-scale virtual environments on-the-fly. Kim et al. [9] extract patchable trajectories of multiple characters from a limited set of input animations and tile patches seamlessly to create a dense crowd of characters interacting with each other. Jordao et al. [10] introduced a crowd sculpting method to interactively design populated environments with an extension of crowd patches. However, the most notable limitation of patch-based methods is the lacking of motion varieties. For instance, if the user's point of view remains static for a while, the periodicity of synthesized crowd animations could be visually noticeable. In addition, certain strict principles must be defined to ensure a seamless transition between patches.

Texture synthesis: Various texture-based flow visualization techniques [20], [21], [22], [23], [24], [25] have been designed to provide a flexible, dense visual representation for a flow field with high spatio-temporal coherencies. In addition, texture synthesis has been extensively used in computer graphics and vision applications, including texture mapping [26], image completion and restoration [27], and pattern synthesis [28]. For a comprehensive survey on texture synthesis techniques, please refer to the recent work by Wei et al. [29]. Despite the success of applying texture synthesis on 2D, 3D, and spatio-temporal data, without considerable efforts it would be difficult to straightforwardly extend existing texture synthesis techniques for traffic flow synthesis, because vehicles on the road need to strictly obey various traffic rules, including collision avoidance, lane changing, acceleration/deceleration, etc.

3 Approach Overview

Given a virtual road network, and a limited set of vehicle trajectory samples (recorded from real-world traffics or even synthetic traffic data), our approach first synthesizes new vehicle trajectories segment by segment. Then, the synthesized vehicle trajectory segments are automatically concatenated together, and further geometrically registered with the road network through cage-based registration.

The input vehicle trajectory set contains a variety of traffic flow segments in terms of the number of lanes and flow density. Each segment contains vehicles' trajectories in a time period, denoted as $\{\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_i, ..., \boldsymbol{P}_M\}$, where $\boldsymbol{P}_i = \{\boldsymbol{p}_i^1, \boldsymbol{p}_i^2, ..., \boldsymbol{p}_i^j, ..., \boldsymbol{p}_i^{N_i}\}$. Here, $\boldsymbol{P}_i$ is the trajectory of the i-th vehicle, and $\boldsymbol{p}_i^j \in \mathbb{R}^2$ is the 2D position of the i-th vehicle at the j-th frame. M and N_i are the total number of vehicles and the total number of motion frames of the i-th vehicle, respectively. At the preprocessing step, we cluster these input trajectories into groups based on their average flow velocities. In this way, during our traffic flow synthesis step (§4), we can select different traffic flow groups as the input traffic flow samples, based on the specific traffic density and speed requirement of the road segment.

In real world traffic flows, each vehicle's motion depends on its surrounding vehicles, which is similar to the locality characteristic of texture synthesis. Therefore, we propose a novel concept of Traffic Texture, in which the spatio-temporal information of a traffic flow segment is conceptually regarded as a 2D texture. It is noteworthy that our defined traffic texture is different from traditional 2D texture that is based on fixed grids. Each texel in traffic texture encodes a vehicle's state at a certain frame including its velocity, position and dynamic relationships with its neighboring vehicles (refer to Fig. 3 for illustration). Its neighboring vehicle IDs are not fixed since the neighboring vehicles could change over time.

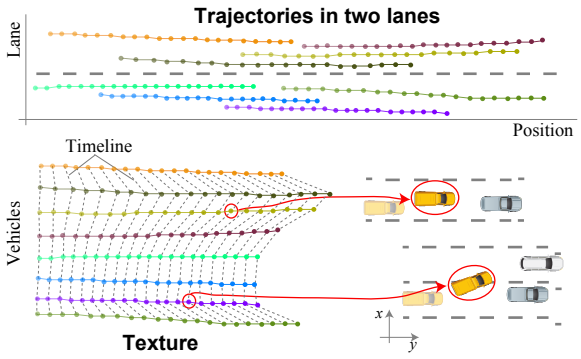


Fig. 3. Texture analogy of a set of two-lanes vehicle trajectories. The spatio-temporal information of the trajectory set can be conceptually viewed as a 2D texture, and each traffic texel encodes a vehicle's states at a certain frame, including its movement information and position relationship with its neighboring vehicles.

Based on the above analogy, we propose a new traffic flow synthesis method by combining texture synthesis with traffic behavior rules. We define a *traffic texture energy metric* according to traffic behavior rules to measure the similarity between the synthesized traffic flows and given traffic flow samples. Each vehicle's velocity in the synthesized traffic flows is determined by finding the best matched texel in the input traffic flow samples. The information of the leader vehicle is also introduced during the synthesis process to ensure collision avoidances in the synthetic traffic flows, which will be elaborated in §4.

After a new traffic flow segment is synthesized, we automatically fit it to the virtual road network using a cage-based registration scheme as follows: we first introduce a traffic-specific cage form to encapsulate the synthetic traffic flows by considering vehicles' strict behavior rules. Then, we accurately register it with the virtual roads based on as-rigid-as-possible deformation with several soft and hard constraints, which will be elaborated in §5.

4 Creating New Traffic Flows

The goal of this step is to produce various new vehicle trajectories for new road networks, which strictly follow traffic rules and break the limits of vehicle number, lane number, and frame number in the input samples. The generated new traffic flows are different from, yet similar to by nature, the input traffic flow samples. To show the varieties in space and time, we synthesize new traffic flows using different input samples, segment by segment. With different initialization settings (§4.1), our method can generate both temporal segments and spatial segments. The length of each

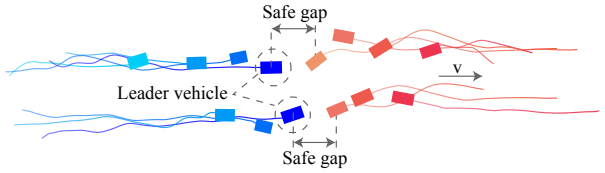


Fig. 4. The initialization of the leader vehicle's states in the synthesized traffic flows (blue) is based on the states of the last vehicle in the previous segment of traffic flows (red).

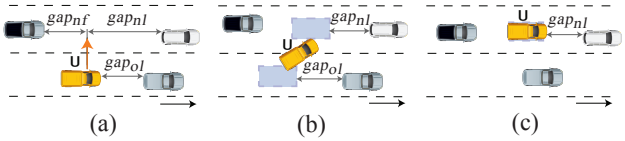


Fig. 5. An illustration of lane-changing process of vehicle *U* and its position relationship with its surrounding vehicles. (a) At the beginning of the lane changing. (b) In the middle of the lane changing. (c) At the end of the lane changing.

segment can be specified by users, which is suggested to be consistent with the length of the input traffic flow sample.

In this work, to generate a new traffic flow segment, we extend the classical pixel-based texture synthesis methodology [30], [31], [32], [33] that creates a new texel sequentially by finding and copying the texel with the most similar neighborhood in the input texture samples. Formally, let T_s denote the input traffic flow samples and T denote the new synthesized traffic flows, we define a traffic texture energy metric to measure the similarity between the synthesized traffic flows and the input traffic flow samples as follows: For a vehicle i at frame j in T, its surrounding states are compared with the neighborhood states of all the vehicles in T_s, and the velocity of the vehicle with the minimal traffic texture energy will be chosen to update $\boldsymbol{p}_i^j$ accordingly. With the growing of texels, we can update all the vehicles' velocities at all the frames in T and generate new traffic flows. Repeating this search-and-update process can iteratively refine the initial estimate of the new traffic flows, with the decreasing of the total traffic texture energy. This iterative update process will be elaborated in §4.2.

4.1 Initialization

In order to bootstrap our traffic flow synthesis process, we employ an initialization process, described below. Since we synthesize new traffic flows segment by segment, we first need to specify both the number of vehicles and the total number of frames in each segment. Different settings of the initial states of the leader vehicles can lead to the synthesized traffic flows as a temporal segment or a spatial segment. Since a synthesized temporal segment is continuous in time domain, its leader vehicles at the first frame should maintain safe gaps with the endmost vehicles at the last frame of the previous traffic flow segment, while a spatial segment should maintain a safe gap between vehicles at every frame. Fig. 4 illustrates the relationship between the synthesized traffic flow segment (blue) and its adjacent traffic flow segment (red).

Then, our initialization algorithm will automatically determine other vehicles' states based on traffic car-following rules [5] and vehicle kinematics. This is done by assigning a random velocity value from the input sample set T_s while maintaining a safe gap between vehicles.

This initialization process makes our method totally different from the patch-based populating method [8]. Our method can generate realistic traffic flows for the virtual roads without any trajectory stitching traces, since additional stitching algorithms are not needed to merge two trajectory flow segments into a longer one.

The lane changing behavior is detected and marked when initializing the position and velocity of a vehicle. According to the safety criteria in the lane-change model by Kesting et al. [13], lane changing takes place if the distances between a vehicle and its neighboring vehicles are large enough for a safe lane changing. As illustrated in Fig. 5(a), we use a simplified criterion for the vehicle *U*'s safe lane changing, that is, the distances between *U* and *U*'s neighboring vehicles gap_{ol}, gap_{nl} and gap_{nf} must be larger than a minimal safe gap. The gap between two vehicles can be straightforwardly computed using their positions.

4.2 Iterative Update Process

The initial estimate of the synthesized traffic flows is iteratively refined, guided by our introduced texture similarity measurement—*traffic texture energy*. We define the traffic texture energy with respect to a local neighborhood similarity in a Markov Random Field. At each iteration, the velocity of a vehicle is updated by finding the neighborhood in the input trajectory samples that is most similar to its neighborhood in the synthesized traffic flows.

To update the velocity $v_{i,j}$ of a vehicle i at frame j, we design specific forms of traffic texture energy for the following three conditions: the velocity in the forward direction $v_{i,j}^y$, the velocity perpendicular to the forward direction $v_{i,j}^x$, and the lane changing scenario. We handle the lane-changing behavior separately, since it is more complex than acceleration/deceleration decision-making in a single lane.

In the simplest case, the vehicle i is not marked as a lane changing vehicle during initialization. Its movement at frame j is only related to its leader vehicle's states at the current frame j and its previous states at frame j-1, which together form its neighborhood in our synthesis algorithm.

Traffic texture energy for updating $v_{i,j}^y$: For a texel candidate in T_s, assuming the vehicle r at frame k, its traffic texture energy E^y over its neighborhood is defined as follows:

$$E^y = \omega_v E_v^y + \omega_b E_b^y + \omega_g E_g^y + \omega_s E_s^y, \tag{1}$$

where the *velocity similarity term* E_v^y measures the current velocity similarity between the vehicles r and i, the *velocity consistency term* E_b^y measures the similarity between the two vehicles' velocities at their respective previous frames, the *follower-leader gap energy term* E_g^y measures the similarity between their gaps with their own leader vehicles, and the *collision avoidances term* E_s^y is introduced to preserve the vehicle i's safe gap with its leader vehicle in T. ω_v, ω_b, ω_g, and ω_s are normalized weighting parameters.

The velocity similarity term E_v^y and the follower-leader gap energy term E_g^y are designed to ensure the synthesized traffic flows have a similar motion pattern with the input traffic flow samples, defined in Eq. 2 and Eq. 3, respectively.

$$E_v^y = \left\| v_{i,j}^y - v_{r,k}^y \right\|_2, \tag{2}$$

$$E_g^y = \left\| d_{i,j} - d_{r,k} \right\|_2. \tag{3}$$

In the above equations, $v_{i,j}^y$ and $v_{r,k}^y$ denote the forward velocity of the vehicle i at frame j in the synthesized traffic flows T, and that of the vehicle r at frame k in the input samples T_s, respectively; $d_{i,j}$ and $d_{r,k}$ denote the distance gap between the vehicle i and its leader vehicle at frame j in T, and the distance gap between the vehicle r and its leader vehicle at frame k in T_s, respectively.

The velocity consistency term, E_b^y, is introduced to ensure the motion continuity, defined in Eq. 4.

$$E_b^y = \left\| v_{i,j-1}^y - v_{r,k-1}^y \right\|_2, \tag{4}$$

where $v_{i,j-1}^y$ and $v_{r,k-1}^y$ are the forward velocities of the vehicles i and r at their previous frames, respectively.

Collision avoidances are realized through the introduction of E_s^y. Suppose $v_{r,k}^y$ is assigned to $v_{i,j}^y$ for updating the position of the vehicle i in T, we need to make sure that the vehicle i keeps a safe distance with its leader vehicle in frame j, which leads to the following formula for E_s^y:

$$E_s^y = \left\| y_{i,j}^{leader} - (y_{i,j-1} + v_{r,k}^y \Delta t) - d_0 \right\|_2, \tag{5}$$

where $y_{i,j}^{leader}$ is the position of i's leader vehicle in the forward direction at frame j, $y_{i,j-1}$ is the position of the vehicle i in the forward direction at previous frame, Δt is the time step, and d_0 is the minimal safe gap between vehicles.

Traffic texture energy for updating $v_{i,j}^x$: Vehicles often need to make sideways movement (perpendicular to the forward direction) for better observing surrounding traffic conditions and avoiding obstacles. Previous traffic simulation methods often model vehicle movement in this direction by simply adding random fluctuations, which is ad hoc by nature. To make the synthesized traffic flows more resemble real-life traffic flows, we search for the most similar neighborhood in T_s to update $v_{i,j}^x$. The corresponding texture energy E^x is defined as follows:

$$E^x = \omega_v E_v^x + \omega_b E_b^x + \omega_l E_l^x, \tag{6}$$

where ω_v, ω_b and ω_l are user-specified weights for different terms, E_v^x and E_b^x are defined in a similar way as E_v^y and E_b^y, respectively. The *lane keeping term* E_l^x is introduced to measure the energy arising from the the vehicle i's deviation from the lane center, since vehicles must keep driving inside the lane marks according to traffic laws and regulations.

Assume $v_{r,k}^x$ is assigned to $v_{i,j}^x$ to update vehicle i's position in the sideways direction, E_l^x is defined as follows:

$$E_l^x = \left\| x_{i,j-1} + v_{r,k}^x \Delta t - p_{ml} \right\|_2, \tag{7}$$

where $x_{i,j-1}$ is the position of the vehicle i in the sideways direction at frame j-1, p_{ml} denotes the center position of the lane. Here we assume drivers typically prefer to drive their vehicles close to the lane center as much as possible.

Lane changing conditions: As illustrated in Fig. 5, a vehicle's lane changing process can be divided into three phases:

(1) The vehicle has the intention to switch to an adjacent lane, but it still stays in the current lane at that time (Fig. 5(a)).
(2) The vehicle is in the middle of two involved lanes (i.e., the old lane and the new lane) for lane changing (Fig. 5(b)).
(3) The vehicle has arrived the target lane, but the lane changing process has not finished, which is illustrated in Fig. 5(c).

When updating $v_{i,j}^y$, we need to determine which phase the vehicle is at, since vehicle movements at different phases are different. This also leads to different texture neighborhood compositions and different traffic texture energy components. In this work, we assume that the vehicle's movement in the forward direction at the phases (1) and (3) is the same as driving in a single lane. So, accordingly, the definition of traffic texture energy for updating $v_{i,j}^y$ can be the same as E^y in Eq. 1.

At the above phase (2), however, the vehicle is between the two lanes. Besides the influence of the vehicles in its old lane, its own movement in the forward direction is also affected by the leader vehicle in the new lane. When computing the texture energy for updating $v_{i,j}^y$, we add an additional energy term E_n^y into Eq. 1, to ensure the safety of lane changing. So, the traffic texture energy E_y for the phase (2) has a new form shown in Eq. 8:

$$E^y = \omega_v E_v^y + \omega_b E_b^y + \omega_g E_g^y + \omega_s E_s^y + \omega_n E_n^y, \tag{8}$$

$$E_n^y = \left\| y_{i,j}^{t\_leader} - (y_{i,j-1} + v_{r,k}^y \Delta t) - d_0 \right\|_2, \tag{9}$$

where ω_n is the weight for E_n^y. E_n^y is computed using Eq. 9, which is similar to Eq. 5. Here, $y_{i,j}^{t\_leader}$ is the position of i's leader vehicle in the new lane in the forward direction.

Since there does not exist the restriction of keeping the vehicle's sideways movement in a single lane during the lane changing process, the neighborhood for updating $v_{i,j}^x$ is only composed of its previous states. Therefore, the corresponding traffic texture energy E^x can be simply defined as:

$$E^x = \omega_v E_v^x + \omega_b E_b^x. \tag{10}$$

Using the above defined texture energy functions, we can find the best matched texel in the input traffic flow samples to update each vehicle's states in the synthesized traffic flows. This procedure is performed iteratively to decrease the total traffic texture energy until a user-specified maximum of iterations are reached. Moreover, by specifying the leader vehicles' states or the leader-follower relationships, our method can generate some emergent patterns of traffic flows such as jams and unstable, stop-and-go patterns of traffic flows (Fig. 6), or the evolving of traffic flows when meeting traffic-light signals (Fig. 2(b)). At the end, we can generate a variety of realistic, natural traffic flows based on input traffic samples.

Fig. 6. An example of synthesized traffic flows by our method to show jams and unstable, stop-and-go patterns of traffic flows.

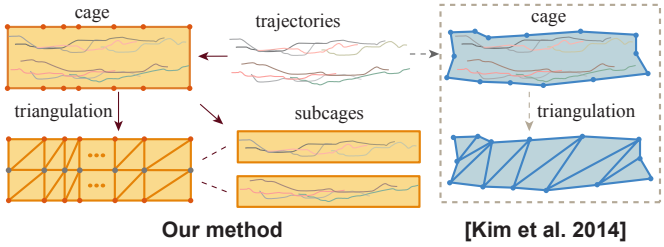


Fig. 7. Cage construction using Kim et al's method [11] (the blue part) and our method (the yellow part). In our method, a cage is divided into several subcages for computing the new positions of vehicle trajectories after deformation. The number of the subcages equals to the number of lanes in the input vehicle trajectories.

5 Cage-based Traffic Flow Registration

After a new set of vehicle trajectories are synthesized, then we need to fit them to virtual roads properly. To tackle this, we introduce a cage-based automatic registration scheme to accurately place the synthesized vehicle trajectories onto the target roads.

Inspired by the recent crowd editing method [11], we also employ a cage to enclose the vehicle trajectories. By deforming the cage to match with the shape of the road, the trajectories of multiple vehicles can be deformed and fitted to the road in a coordinated manner. However, if we directly use an arbitrarily concave polygon as the cage form, as in the work of [11], for traffic flow registration, it would be difficult to deform and strictly align such an irregularly shaped cage to the regularly shaped road (refer to Fig. 7 for an illustration). To ensure traffic flows strictly reside within the road boundaries, we define a specific cage structure. Moreover, we design a few traffic-specific constraints for cage deformation to preserve the characteristics of the input traffic flows as much as possible.

5.1 Cage Construction and Representation

For each set of input vehicle trajectories, its target road information can be easily obtained, including the road structure (expressed as the boundary marker points), the number of lanes, and the road length/width. The outer boundaries of the cage can be constructed based on the road information, by establishing the mapping between vehicle trajectories and their current road. As shown in Fig. 7, we construct a cage as follows: First, the four corner points of the cage are fixed to exactly match with the width and position of the road. Then, based on these corner points, more cage vertices are created via sampling according to the marker points on the road boundaries to ensure smooth cage deformation. Specifically, after the outer boundaries of the cage are determined, its interior region is tessellated using constrained Delaunay triangulation [34]. We add splitting points (gray points in Fig. 7) between lanes in the interior region of the cage, in order to ensure vehicle trajectories always stay within their own lanes, and the lanes always maintain their relative widths, without local extrusions during the deformation.

Vehicle trajectories inside the cage are represented by mean value coordinates (MVCs) [35] with respect to the cage vertices. The computed MVCs are fixed during the deformation process. However, using a single cage to deform the trajectories of all the vehicles cannot robustly keep each trajectory in its original lane during the deformation, which may cause vehicle collisions and lane departure in animation. So, we introduce a subcage for the vehicle trajectories in each lane. Fig. 7 shows two subcages for a set of two-lanes vehicle trajectories.

Given the cyclically-defined subcage boundary vertices $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_m\}$, the MVCs of $\boldsymbol{p}_k$ (a point on a vehicle trajectory), $\{\lambda_k^1, \lambda_k^2, ..., \lambda_k^m\}$, are defined as follows:

$$\lambda_k^i = \frac{\omega_i}{\sum_{j=1}^{m} \omega_j}, \quad \omega_i = \frac{tan\,(\phi_{i-1}/2) + tan\,(\phi_i/2)}{\|p_k - v_i\|}, \tag{11}$$

where ϕ_i is angle $\angle \boldsymbol{v}_{i+1}, \boldsymbol{p}_k, \boldsymbol{v}_i$ in the cage, and m is the total number of cage vertices.

Then, the location of the point can be described as a weighted linear combination of the subcage boundary vertices. When the cage vertices are changed, the updated position of $\boldsymbol{p}_k$, denoted as $\hat{\boldsymbol{p}}_k$, is computed as follows:

$$\hat{\boldsymbol{p}}_k = \sum_{i=1}^{m} \lambda_k^i \hat{\boldsymbol{v}}_i, \tag{12}$$

where $\{\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2, ..., \hat{\boldsymbol{v}}_m\}$ are the updated subcage boundary vertices.

5.2 Traffic Positioning via Cage Deformation

Different from the interactive manipulation of human crowds, traffic positioning challenges us to automatically deform the cage to match with an arbitrary road section. We use a cage-based automatic registration scheme to achieve this. Specifically, we formulate the deformation of the triangular cage mesh as an as-rigid-as-possible mesh deformation [36]. In this deformation process, several vertices on the cage boundaries are first specified to match with certain sampling points on the road. These vertices are used as the control points for deformation, and then the other vertices on the cage mesh are manipulated based on the as-rigid-as-possible deformation formula. Moreover, a set of constraints are enforced in the deformation formula, which can be divided into two types: soft and hard constraints. A soft constraint is included as a term in the deformation energy function, while a hard constraint needs to be exactly satisfied.

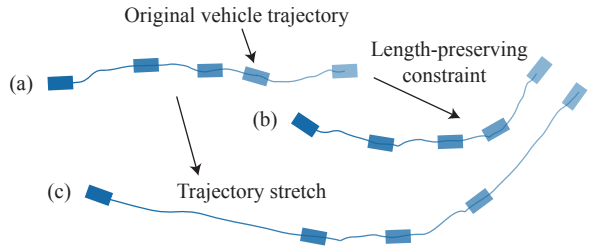


Fig. 8. Cage-based traffic flow registration with the soft constraints alone may lead to the stretching of vehicle motion. (a) The original vehicle motion. (b) The deformed result by our approach with the hard length-preserving constraint. (c) The deformed motion with the soft constraints alone.

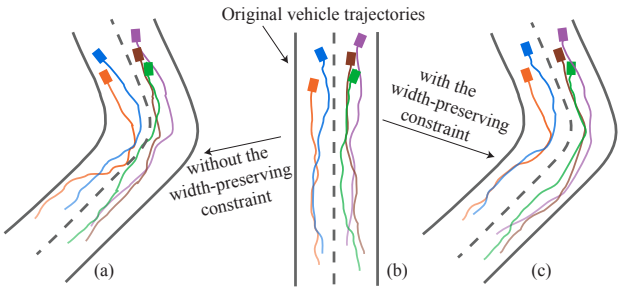


Fig. 9. Without the width-preserving constraint, the deformed vehicle trajectories may not always stay in the same lanes as the original vehicle trajectories. (a) The deformed trajectories without the width-preserving constraint. (b) The original vehicle trajectories. (c) The deformed vehicle trajectories by our approach with the width-preserving constraint.

Soft constraints: Soft constraints are mainly designed to ensure that the cage deformation is as rigid as possible and the control points are matched with the sample points on the road as closely as possible, which leads to two corresponding energy items: the cage deformation energy term, E_D, to measure the cage mesh distortion, and the control point error term, E_C, to describe the matching error between the control points and the sampling points on the road. These two soft constraints can be satisfied by solving an energy minimization problem, in which the objective function is written as $E_D + \omega_c E_C$. Here ω_c is the weight for E_c.

E_D is designed to ensure the cage mesh deformation as rigid as possible, which sums up the local deformation energy over each triangle in the cage mesh to compute the global energy [11], [37]:

$$E_D = \sum_{t=1}^{T} \sum_{i=0}^{2} cot(\theta_i^t) \left\| (\hat{\boldsymbol{v}}_i^t - \hat{\boldsymbol{v}}_{i+1}^t) - R_t(\boldsymbol{v}_i^t - \boldsymbol{v}_{i+1}^t) \right\|^2, \tag{13}$$

where T is the total number of triangles in the cage mesh, $\boldsymbol{v}_i^t$, $\boldsymbol{v}_{i+1}^t$, $\hat{\boldsymbol{v}}_i^t$ and $\hat{\boldsymbol{v}}_{i+1}^t$ are the locations of the i-th and (i+1)-th vertices in t-th triangle before and after deformation, respectively, and R_t is the estimated rotation matrix between the original t-th triangle and the deformed t-th triangle. Here, $\cot(\theta_i^t)$ is the per-edge weight for compensating non-uniform triangle shape, in which θ_i^t is the angle opposite to the edge $(\boldsymbol{v}_i^t, \boldsymbol{v}_{i+1}^t)$ in the t-th triangle.

E_C is defined as follows:

$$E_C = \sum_{c=1}^{C} \left\| \hat{\boldsymbol{v}}_c - \boldsymbol{r}_c \right\|^2, \tag{14}$$

where C is the total number of control points, $\hat{\boldsymbol{v}}_c$ and $\boldsymbol{r}_c$ are the deformed position of the c-th control point and the position of the corresponding road sampling point.

Hard constraints: Using the as-rigid-as-possible mesh deformation with the above soft constraints alone may lead to undesired results in the deformed traffic flow animation, such as the stretching of vehicle motion (Fig. 8) and the mixing of the trajectories in different lanes (Fig. 9). So, it is necessary to preserve certain features of the vehicle trajectories during the deformation. In this work, we introduce two traffic-specific hard constraints for this purpose.

First, we add a *length-preserving constraint* to preserve the cage's original length, because cage stretching or squeezing may directly lead to the undesired change of the vehicles' speeds or even vehicle collisions. The increased (or decreased) length of deformed trajectories makes the vehicles to move faster (or slower) than their original speeds (refer to Fig. 8). For example, the vehicles may have to move faster at some time intervals on the deformed trajectories while move slower on other intervals, or their velocities may be even much faster than the allowed maximum speed of the road. Such an irregular speed change is usually undesired in traffic flow animation. Our length-preserving constraint ensures the cage will not be stretched or squeezed during the deformation, and the vehicles in the deformed traffic flows move with their original speeds. This constraint can be expressed as follows:

$$\sum_{i=1}^{k-1} (\|\hat{\boldsymbol{v}}_{i+1} - \hat{\boldsymbol{v}}_i\| - \|\boldsymbol{v}_{i+1} - \boldsymbol{v}_i\|) = 0, \tag{15}$$

where k is the number of vertices on one side of the cage boundaries. Since the cage shape is symmetrical, we can easily infer $k = m/2$.

Second, we introduce a *width-preserving constraint* to ensure the deformed vehicle trajectories stay in the same lanes as the original vehicle trajectories. Even though our subcage mechanism is designed to maintain the vehicles' relative formation and separate the trajectories into different lanes, the deformed vehicle trajectories can still lead to potential collisions or an insufficient clearance between vehicles, as shown in Fig. 9. Hence, the width-preserving constraint, defined below, is introduced to prevent such artifacts.

$$\|\hat{\boldsymbol{v}}_{i+k} - \hat{\boldsymbol{v}}_i\| - \|\boldsymbol{v}_{i+k} - \boldsymbol{v}_i\| = 0, \quad i = 1, 2, ..., k. \tag{16}$$

We employ an alternating least-square optimization algorithm [37] to minimize $E_D + \omega_c E_C$ with the above two hard constraints, where $\hat{\boldsymbol{V}}$ and $\{R_t\}$ are unknowns. As an example, our cage-based traffic registration method can work well for large-scale deformations such as a curvy road in Fig. 2(a).

6 Experimental Results and Evaluations

To demonstrate our method, we genetrated different density traffic flows on a few virtual road networks. Fig. 2 shows snapshots of our synthesized traffic flows in different types

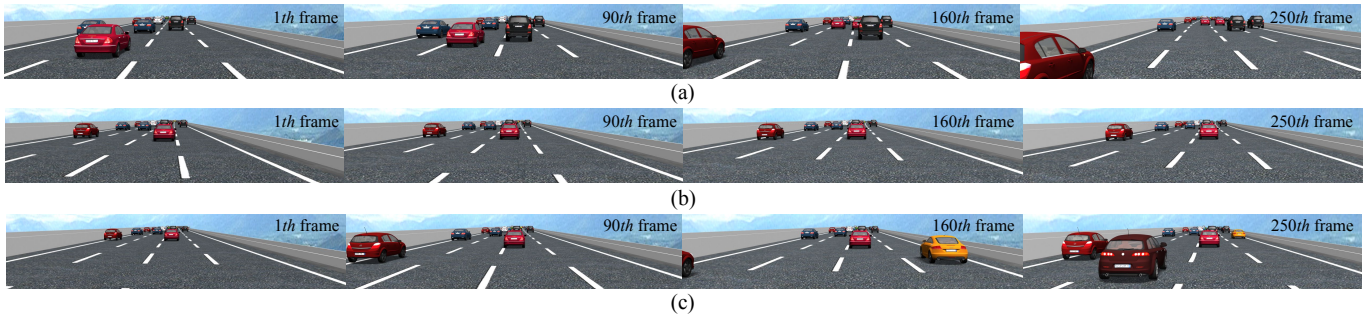


Fig. 10. Snapshots of the generated traffic flows on virtual road networks from the driver's view: (a) with the ground-truth data, (b) using the IDM model [7], and (c) using our method.

of road networks. Fig. 10 shows the visual comparisons among our method, the ground-truth data, and the IDM-based simulation method [7] from the driver's view (animation comparisons are enclosed in the supplemental video). We can see that the vehicles simulated by our method show a variety of acceleration/deceleration patterns similar to those in the ground-truth data, while the vehicles simulated by the IDM-based method tend to keep similar gaps with their neighboring vehicles.

The input vehicle trajectory set can be any vehicle trajectory data, either acquired via various sensors or video-based tracking, or even vehicle trajectory data synthesized by other methods. In our experiments, the input samples were extracted from the Next Generation Simulation (NGSIM) program [38], where the vehicle trajectories are acquired from multiple cameras installed along the road. Therefore, in this work we consider the NGSIM traffic flow data as the ground-truth data. The original trajectory data in the NGSIM dataset exhibits certain noise artifacts. Therefore, we pre-processed all the trajectories before using a filter [39]. Note that, since low-frequency noise or left/right bias cannot be perfectly separated from irregular vehicle movements via filtering, they may still be kept in the filtered dataset. To this end, in our experiments, the used traffic flow samples were spatio-temporal segments with the duration in the range of 20 to 300 frames, with the number of vehicles in the range of 5 to 210, and with the number of lanes in the range of 2 to 4. As shown in the demo video, our synthesized results reproduced not only the original vehicle movement patterns in the input sample data but also some non-ideal behaviors (such as sudden start/stop and left/right bias) that are also manifested in the filtered ground-truth data.

Parameter	Value	Description
ω_v	3.2	weight for E_v^y and E_v^x
ω_b	1.6	weights for E_b^y and E_b^x
ω_g	0.81	weight for E_g^y
ω_s	[0.56, $+\infty$)	weight for E_s^y
ω_n	[0.56, $+\infty$)	weight for E_n^y
ω_l	60	weight for E_l^x
d_0	3	the minimal safe gap between vehicles

TABLE 1
Parameter values used in our experiments

6.1 Performance and Convergence

Table 1 summarizes key parameter values used in our experiments. It is noteworthy that the values of ω_s and ω_n depend on the distance gap between the vehicle and its leader vehicle. A smaller gap implies larger weight values for these two parameters; and vice versa. If the gap is smaller than the minimal safe gap d_0, the two parameters are set to positive infinity.

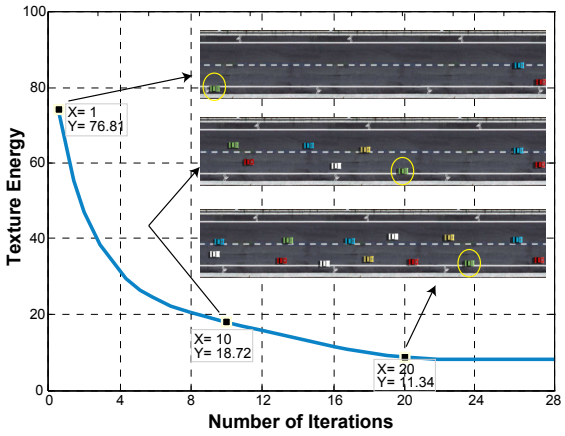


Fig. 11. Convergence of our traffic flow synthesis algorithm. The blue line shows how the traffic texture energy of the synthesized results is changed over iterations. The green vehicle within the yellow ellipses have better car-following and lane-keeping behaviors when more iterations are executed.

Runtime performance: Table 2 shows the runtime performances of our method for synthesizing a set of new vehicle trajectories and registering it to virtual roads. The runtime performance of our synthesis method highly depends on several parameter settings, including the vehicle number and frame number in the synthesized traffic flows, the vehicle number and frame number of the input trajectory sample set and the iteration level of the algorithm. The runtime of our vehicle trajectory synthesis process has an approximately linear relation to each of these parameters. The runtime of our cage-based registration process mainly depends on the subdivision level of the cage mesh, which is also shown in Table 2. All the reported times were obtained on a 64bit desktop machine with a 3.30GHz Inter Xeon CPU E3-1230 v3 processor and 8GB memory.

synthesized trajectories			input trajectory samples			iteration number	synthesis time (second)	cage triangles number	registration time (second)	total time (second)
vehicle number	frame number	lane number	vehicle number	frame number	lane number					
8	50	2	8	20	2	1	0.001	64	0.008	0.009
11	100	2	9	50	2	10	0.08	132	0.02	0.10
50	400	3	18	100	2	20	13.07	798	0.56	13.63
67	700	4	39	200	3	30	210.82	1864	2.31	213.13

TABLE 2
Runtime statistics of our method

Convergence of our synthesis algorithm: In our experiments, all the generated traffic flows were converged within 30 iterations. The convergence is reflected by the decreasing of the traffic texture energy to a steady state. In Fig. 11, we plot how the traffic texture energy of the synthesized traffic flows is decreased over iterations when we applied our method to populate a simple straight road with two lanes. As shown in this figure, the traffic texture energy is quickly decreased when more iterations are executed. In this figure, we also show the snapshots of the synthesized traffic flows at iteration #1, iteration #10, and iteration #20. As more iterations are executed, we can see the synthesized traffic flows have more realistic car-following and lane-keeping behaviors.

6.2 Comparison and Validation

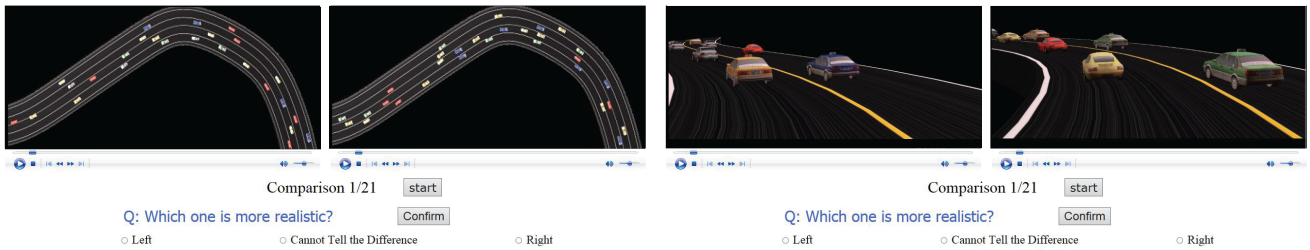
In order to evaluate the effectiveness of our approach, for the same set of test virtual roads, we generated traffic flows by three different methods: (1) the ground-truth (i.e., the NGSIM traffic flow data), (2) our method, and (3) one of the latest developments of the IDM model [7].

The reasons we choose to compare our method with the IDM-based simulation method, instead of other clearly related previous works in traffic simulation including traffic simulation models [4], [7], [40] and traffic reconstruction method [15] are: In our comparison user study, the comparison with simulation is aimed to compare the synthesized microscopic traffic details such as the acceleration/deceleration and lane-changing behaviors. Sewall et al.'s continuum traffic [4] cannot generate detailed motions of vehicles. The hybrid simulation method [40] controls each vehicle's detailed motion by an agent-based car-following method [6] that is an extension of IDM. In addition, Wilkie et al's traffic flow reconstruction work [15] estimates the full states of the traffic flows from sparse sensor measurements, which is quite different from our traffic populating method. Therefore, we choose Shen et al.'s work [7], one of the latest developments of IDM, for our comparison user study. For the sake of a fair comparison, the initialization of the IDM-based method [7] is the same as that of our method. The used parameter values in the IDM-based method, including the average velocity, the maximum acceleration, the minimum deceleration, and the minimum gap, were calibrated by first fitting the IDM model to the NGSIM traffic flow data, and then they were further diversified by adding random variations. It is noteworthy that the simulation results by our implementation are consistent with those provided by the authors of the IDM model [41].

Paired comparison user studies: For each test virtual road, we generated three different traffic flow animations using the above three different approaches. Then, we performed paired comparison user studies to evaluate the *comparative realism* between any two of the three methods. We chose the paired comparison methodology [42] for our user studies due to its proven effectiveness. Its basic idea is, instead of explicitly rating visual stimuli, participants are asked to select the perceptually better one between two visual stimuli (a pair). As noted by Ma and Deng [43], in a paired comparison study, the participants can avoid to make forced, inaccurate perception decisions, e.g., assign a subjective and quantitative rating to each stimulus. Instead, they just need to pick the perceptually better one between two visual stimuli (a pair), which increases the accuracy and robustness of the experiment outcomes.

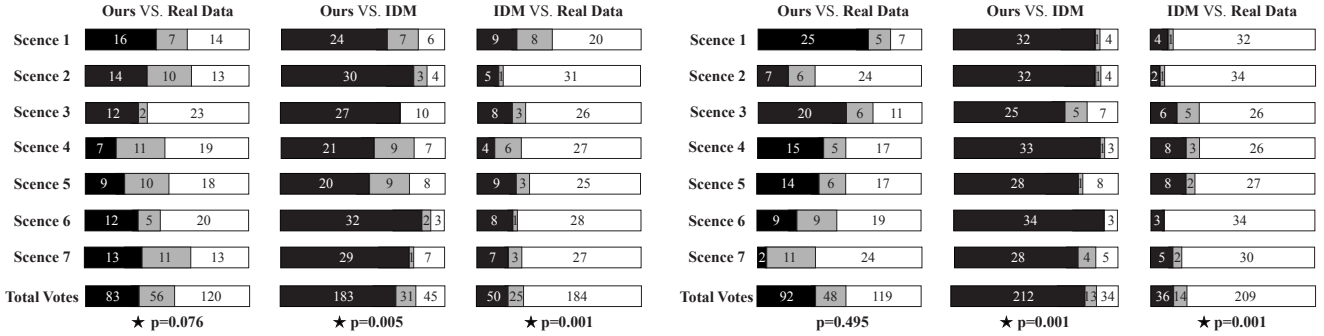
We chose 7 test virtual road scenes for our studies. For each test scene, we generated 3 traffic simulations using the above three different methods. Then, for each traffic simulation, we rendered it using two different camera views: the *bird's eye view*, and the *driver's view*. In the bird's eye view, we render the traffic animation from the view of a bird; while at the driver's view, we put a virtual camera at a fixed position in a moving vehicle (i.e., approximating the driver's view) and render the corresponding traffic animation. The main reason for including the driver's view in our studies, besides the conventional bird's eye view, is that, most people are much more familiar with real-world traffic flows from the driver's view, other than from the bird's eye view. Therefore, we argue that conducting a paired comparison study with the driver's view would produce more sound experiment outcomes. To the end, we produced a total of 42 traffic animation clips as the stimuli in our studies (7 scenes $\times$ 3 methods $\times$ 2 views). Based on the 42 animation clips, we further formed 42 pairs for our comparison studies. Specifically, for each test scene, we obtained 3 pairs (ground-truth versus our method, our method versus IDM, and ground-truth versus IDM) for the bird's eye view study (a snapshot of the bird's eye view study is shown in Fig. 12(a)), and similar 3 pairs for the driver's view study (a snapshot of the driver's view study is shown in Fig. 12(b)).

We recruited 37 participants to participate in our paired comparison studies. All the participants are graduate students in a university, whose ages are in the range of 20 to 30, with normal visions. They were asked to select the more realistic one between the two animation clips in a pair. Besides, the participants were allowed to select the undecided option if they cannot decide which clip is perceptually better. To counterbalance the order of the visual stimuli, the comparison pairs were displayed in a random order for each participant. The participants can choose to play the two animation clips in a pair one after another (do not need to play them simultaneously), and they can view



(a) (b)

Fig. 12. Snapshots of the bird's eye view study (a) and the driver's view study (b).



(a) (b)

Fig. 13. The experiment outcomes of the bird's eye view study (a) and the driver's view study (b). Black boxes at the left side and white boxes at the right side indicate the total number of times when the participants voted the results by the corresponding method. Gray boxes in the middle indicate "undecided choices" (i.e, perceptually equivalent). The symbol $\star$ indicates the computed statistical significance according to a two-tailed independent one-sample t-test with p-value < 0.05.

the clips for unlimited times before make their decisions.

(1) The bird's eye view study: The experiment outcomes of the bird's eye view study are shown in Fig. 13(a). Besides the votes for each test scene, we also show the information of the total votes in this figure. As clearly seen in this figure, our approach gained significant more votes than the IDM method, while the ground-truth method (real data) outperformed both our method and the IDM method, not surprisingly. To quantify the statistical significance of the voting outcomes, we also performed a two-tailed independent one-sample *t*-test and computed the corresponding *p*-values (reported in Fig. 13(a)). In this study, our method is statistically significantly better than the IDM method even if the significance level is set to 0.01, while the ground-truth method cannot claim its statistical significance over our method, with the *p*-value = 0.076.

(2) The driver's view study: The experiment outcomes of the driver's view study are shown in Fig. 13(b). The outcomes are consistent with those of the bird's eye view study. However, in the driver's view study, the difference between our method and the IDM method is further widened (e.g., the *p*-value goes down from 0.005 to 0.001), as shown in Fig. 13(b). Meanwhile, the statistical difference between the ground-truth method and our method is less obvious, with the *p*-value going up from 0.076 to 0.495. These results show that from the driver's view, our method can synthesize traffic flows that are highly similar to the ground-truth data, and it is statistically significantly better than the IDM method. To understand the outcome difference between the bird's eye view study and the driver's view study, we conducted informal post-study interviews with a few selected participants. Majority of their feedbacks point to that, in the driver's view study, it is easier for them to visually notice those less regular movements of vehicles; they typically judged the vehicles with less regular movements more realistic.

Quantitative validation: In addition to the above paired comparison studies, we also conducted a quantitative validation among our method, the NGSIM dataset, and the IDM-based method. For the results by each method, we calculate the following two measurement distributions: i) the distribution of the vehicle velocities, and ii) the distribution of the gaps between vehicles. The comparison results are illustrated in Fig. 14. As clearly indicated in this figure, compared to the IDM-based method, the results generated by our method have more similar distributions with the NGSIM dataset (i.e., real-data in this figure) in terms of both velocity and the gap between vehicles. This indicates that the synthesized traffic flows by our method can capture more statistical characteristics of the real-world data than the traditional IDM-based method.

7 Conclusion

We introduce an effective and scalable method to generate realistic traffic flows on virtual roads, given a limited set of traffic flow samples. To validate the effectiveness of our approach, we conducted two paired comparison user studies to validate the effectiveness of our approach. The experiment outcomes show that our approach is able to

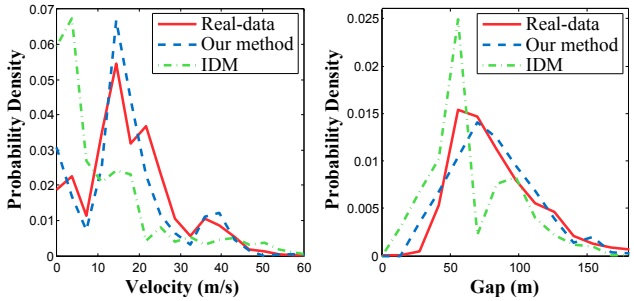


Fig. 14. Discrete probability density distributions of the vehicle velocities and the gaps between vehicles among our method, the NGSIM dataset (i.e., the real-data), and the IDM-based method.

produce statistically significantly better results than the state of the art microscopic simulation based method (i.e., the IDM model), and the results by our approach is reasonably close to the ground-truth traffic flow data (i.e., the NGSIM traffic data).

Collision avoidance has always been a challenging problem in crowd simulations. Since microscopic traffic rules (e.g., a safe gap between vehicles) are incorporated into our traffic texture energy metric, our method can guarantee vehicles' safe driving without collisions. Moreover, in our vehicle trajectory registration process, our introduced subcage mechanism and the two traffic-specific hard constraints for deformation also prevent the potential collisions between vehicles in the registered traffic flows.

The traffic behavior can be very complex when meeting road splitting and merging. Handling road splitting and merging is one of the limitations of our current work. As an alternative solution, a hybrid strategy can be adopted for handling road splitting and merging, that is, combining our data-driven method with agent-based techniques such as IDM. Indeed, similar hybrid strategies have been successfully used in Wilkie et al.'s traffic flow reconstruction work [15].

The process of creating new vehicle trajectories in our method can be implemented as an offline pre-computation task given the basic information of the target virtual roads, or can be called online before our cage-based automatic traffic flow registration process. However, it still cannot populate large-scale road environments at an interactive rate on an off-the-shelf computer, which is one of the limitations of our approach. In our vehicle trajectory synthesis process, significant computational time is spent for searching all of the input trajectory flow samples for each vehicle's state update during each iteration, which is less efficient. In the future, we plan to investigate GPU-accelerated or parallel computing schemes, or fast texture synthesis algorithms to speed up our current approach.

Another limitation of our current approach is that, in the real-word drivers may have different driving behaviors on roads with different shapes, such as making a deceleration decision at the corners of the road; however, our cage-based registration process does not consider such factors, and it just deforms the vehicle trajectory's shape while preserving its original motion pattern as much as possible. In addition, our approach cannot handle the potential interaction between vehicles and other moving objects (e.g., pedestrians, moving obstacles) on virtual roads.

In the future, we plan to extend our method to handle a large variety of traffic behaviors, such as the dynamic perception of surrounding environments and taking individual drivers into consideration (that is, add new terms to characterize different drivers to our synthesis formula). Adding driver characterization to our current synthesis framework is challenging. The main reason is, in order to add new terms for driver characterization, we first need to somehow extract the driver factors from the ground-truth traffic flow data, which is not currently available in many recorded traffic flow datasets, unfortunately. We are also interested in integrating our approach with various crowd editing tools and algorithms (e.g., [11]). In this way, users can flexibly refine the quality of traffic flows directly synthesized by our approach.

ACKNOWLEDGMENTS

Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant no. 61272298). Zhigang Deng was supported by the Joint Research Fund for Overseas Chinese, Hong Kong, and Macao Young Scientists of the National Natural Science Foundation of China (Grant No. 61328204) and the National Science Foundation of United States (Grant no. 1524782).

REFERENCES

[1] M. J. Lighthill and G. B. Whitham, "On kinematic waves. ii. a theory of traffic flow on long crowded roads," in *Proceedings of the Royal Society of London. Series A.* Mathematical and Physical Sciences, 1955, pp. 317–345.

[2] A. Aw and M. Rascle, "Resurrection of second order models of traffic flow," *SIAM Journal of Applied Math*, vol. 60, no. 3, pp. 916–938, 2000.

[3] H. M. Zhang, "A non-equilibrium traffic model devoid of gas-like behavior," *Transportation Research Part B*, vol. 36, no. 3, pp. 275–290, 2002.

[4] J. Sewall, D. Wilkie, P. Merrell, and M. C. Lin, "Continuum traffic simulation," *CGF*, vol. 29, no. 2, pp. 439–448, 2010.

[5] D. L. Gerlough, "Simulation of freeway traffic on a general-purpose discrete variable computer," PhD thesis, UCLA, 1955.

[6] M. Treiber and D. Helbing, "Microsimulations of freeway traffic including control measures," *Automatisierungstechnik*, vol. 49, pp. 478–484, 2001.

[7] J. Shen and X. Jin, "Detailed traffic animation for urban road networks," *Graphical Models*, vol. 74, no. 5, pp. 265–282, 2012.

[8] B. Yersin, J. Maïm, J. Pettré, and D. Thalmann, "Crowd patches: Populating large-scale virtual environments for real-time applications," in *I3D'09*. ACM, 2009, pp. 207–214.

[9] M. Kim, Y. Hwang, K. Hyun, and J. Lee, "Tiling motion patches," in *SCA'12*. Eurographics Association, 2012, pp. 117–126.

[10] K. Jordao, J. Pettré, M. Christie, and M.-P. Cani, "Crowd sculpting: A space-time sculpting method for populating virtual environments," *CGF*, vol. 33, no. 2, pp. 351–360, 2014.

[11] J. Kim, Y. Seol, T. Kwon, and J. Lee, "Interactive manipulation of large-scale crowd animation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, p. 83, 2014.

[12] X. Lu, W. Chen, M. Xu, Z. Wang, Z. Deng, and Y. Ye, "Aa-fvdm: An accident-avoidance full velocity difference model for animating realistic street-level traffic in rural scenes," *Computer Animation and Virtual Worlds*, vol. 25, no. 1, pp. 83–97, 2014.

[13] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1999, no. 1, pp. 86–94, 2007.

[14] J. van den Berg, J. Sewall, M. Lin, and D. Manocha, "Virtualized traffic: Reconstructing traffic flows from discrete spatio-temporal data," *TVCG*, vol. 17, no. 1, pp. 26–37, 2010.

[15] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 89:1–89:10, 2013.
[16] Q. Chao, J. Shen, and X. Jin, "Video-based personalized traffic learning," *Graphical Models*, vol. 75, no. 6, pp. 305 – 317, 2013.
[17] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi, "Group motion editing," in *ACM SIGGRAPH 2008 Papers*, 2008, pp. 80:1–80:8. [Online]. Available: http://doi.acm.org/10.1145/1399504.1360679
[18] M. Kim, K. Hyun, J. Kim, and J. Lee, "Synchronized multi-character motion editing," in *ACM SIGGRAPH 2009 Papers*, 2009, pp. 79:1–79:9. [Online]. Available: http://doi.acm.org/10.1145/1576246.1531385
[19] E. S. L. Ho, T. Komura, and C.-L. Tai, "Spatial relationship preserving character motion adaptation," in *ACM SIGGRAPH 2010 Papers*. ACM, 2010, pp. 33:1–33:8.
[20] J. J. van Wijk, "Image based flow visualization," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 745–754, Jul. 2002.
[21] R. S. Laramee, H. Hauser, H. Doleisch, B. Vrolijk, F. H. Post, and D. Weiskopf, "The state of the art in flow visualization: Dense and texture-based techniques," *CGF*, vol. 23, no. 2, pp. 203–221, 2004.
[22] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 795–802, Jul. 2005.
[23] A. W. Bargteil, F. Sin, J. E. Michaels, T. G. Goktekin, and J. F. O'Brien, "A texture synthesis method for liquid animations," in *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 345–351.
[24] V. Kwatra, D. Adalsteinsson, T. Kim, N. Kwatra, M. Carlson, and M. C. Lin, "Texturing fluids," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 13, no. 5, pp. 939–952, 2007.
[25] R. Narain, V. Kwatra, H.-P. Lee, T. Kim, M. Carlson, and M. C. Lin, "Feature-guided dynamic texture synthesis on continuous flows," in *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, ser. EGSR'07. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 361–370.
[26] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped textures," in *Proc. of SIGGRAPH'00*, 2000, pp. 465–470. [Online]. Available: http://dx.doi.org/10.1145/344779.344987
[27] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 303–312, Jul. 2003. [Online]. Available: http://doi.acm.org/10.1145/882262.882267
[28] Y.-T. Yeh, K. Breeden, L. Yang, M. Fisher, and P. Hanrahan, "Synthesis of tiled patterns using factor graphs," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 1, p. 3, 2013.
[29] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk, "State of the art in example-based texture synthesis," in *Eurographics 2009, State of the Art Report, EG-STAR*. Eurographics Association, 2009, pp. 93–117.
[30] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV'99*, vol. 2, 1999, pp. 1033–1038.
[31] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. of SIGGRAPH'00*. ACM Press, 2000, pp. 479–488.
[32] S. Lefebvre and H. Hoppe, "Appearance-space texture synthesis," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 541–548, 2006.
[33] Y. Zhao, X. Jin, Y. Xu, H. Zhao, M. Ai, and K. Zhou, "Parallel style-aware image cloning for artworks," *TVCG*, vol. 21, no. 2, pp. 229–240, 2015.
[34] J. Shewchuk, "Triangle: Engineering a 2d quality mesh generator and delaunay triangulator," in *Applied Computational Geometry Towards Geometric Engineering*. Springer, 1996, pp. 203–222. [Online]. Available: http://dx.doi.org/10.1007/BFb0014497
[35] M. S. Floater, "Mean value coordinates," *Computer aided geometric design*, vol. 20, no. 1, pp. 19–27, 2003.
[36] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 1134–1141, 2005.
[37] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *SGP'07*, 2007, pp. 109–116.
[38] "Next generation simulation," http://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm., 2013.
[39] C. Thiemann, M. Treiber, and A. Kesting, "Estimating acceleration and lane-changing dynamics from next generation simulation trajectory data," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2088, pp. 90–101, 2008.
[40] J. Sewall, D. Wilkie, and M. C. Lin, "Interactive hybrid simulation of large-scale traffic," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 135, 2011.
[41] "Traffic-simulation.de: Ring road," http://www.traffic-simulation.de/, 2011.
[42] M. G. Kendall and B. B. Smith, "On the method of paired comparisons," *Biometrika*, pp. 324–345, 1940.
[43] X. Ma and Z. Deng, "Natural eye motion synthesis by modeling gaze-head coupling," in *IEEE VR*, 2009, pp. 143–150.

Qianwen Chao is a Lecturer of Computer Science at Xidian University (China). She earned her Ph.D degree in Computer Science from the State Key Laboratory of CAD&CG, Zhejiang University in 2016. Prior that, She received her B.Sc. degree in computer science in 2011 from Xidian University. Her main research interests include crowd animation, cloth animation and swarm micro-robotics.

Zhigang Deng is a Full Professor of Computer Science at University of Houston.His research interests include computer graphics, computer animation, virtual human modeling and animation, and human computer interaction. He earned his Ph.D. in Computer Science at the Department of Computer Science at the University of Southern California in 2006. Prior that, he also completed B.S. degree in Mathematics from Xiamen University (China), and M.S. in Computer Science from Peking University (China). Besides the CASA 2014 general co-chair and SCA 2015 general co-chair, he currently serves as an Associate Editor of Computer Graphics Forum, and Computer Animation and Virtual Worlds Journal.

Jiaping Ren is a Ph.D candidate of the State Key Lab of CAD&CG, Zhejiang University, China. She received her B.Sc. degree in science in 2013 from Zhejiang University of Technology, China. Her main research interests include crowd animation and insect swarm animation.

Qianqian Ye is a graduate student of the State Key Lab of CAD&CG, Zhejiang University, China. She received her B.Sc. degree in digital media technology in 2014 from Zhejiang University. Her main research interests include crowd animation and insect swarm animation.

Xiaogang Jin received the B.Sc. degree in computer science and the M.Sc. and Ph.D degrees in applied mathematics from Zhejiang University, P. R. China, in 1989, 1992, and 1995, respectively. He is a professor in the State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include digital geometry processing, geometric modeling, 3D printing, virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling, and image processing. He received an ACM Recognition of Service Award in 2015. He is a member of the IEEE and ACM.