A Deep Learning-Based Framework for Intersectional Traffic Simulation and Editing

Huikun Bi, Tianlu Mao, Zhaoqi Wang, and Zhigang Deng, *Senior Member, IEEE*

Abstract—Most of existing traffic simulation methods have been focused on simulating vehicles on freeways or city-scale urban networks. However, relatively little research has been done to simulate intersectional traffic to date despite its broad potential applications. In this paper, we propose a novel deep learning-based framework to simulate and edit intersectional traffic. Specifically, based on an in-house collected intersectional traffic dataset, we employ the combination of convolution network (CNN) and recurrent network (RNN) to learn the patterns of vehicle trajectories in intersectional traffic. Besides simulating novel intersectional traffic, our method can be used to edit existing intersectional traffic. Through many experiments as well as comparative user studies, we demonstrate that the results by our method are visually indistinguishable from ground truth, and our method can outperform existing methods.

Index Terms—Traffic simulation, crowd simulation, data-driven, deep learning, intersectional traffic

1 Introduction

Virtual traffic has been increasingly used in urban planning, computer games, urban network visualization, virtual reality, and auto-driving applications in recent years. In particular, with the rapid development of digital earth (e.g., Google Maps and Virtual Earth) and smartphone techniques, more and more real-world traffic data can be efficiently collected for traffic visualization and simulation applications [1].

Most of existing traffic simulation methods have been focused on simulating vehicles on freeways or city-scale urban networks, using either macroscopic models (e.g., [2], [3]) or microscopic models (e.g., [4], [5], [6], [7]), to generate vehicle trajectories. However, to date relatively little research has been done to simulate *intersectional* traffic. For instance, in the work of [8] and some well-known traffic simulators [9], [10], the simulation of intersectional traffic is over-simplified into a queue system, which clearly falls short of modeling complex real-world intersectional traffic. The vehicles simulated in SUMO [10] drive along well-defined lanes (Fig. 1d) and follow car-following rules. Compared to ground truth intersectional traffic (Fig. 1a), the trajectories simulated by SUMO (Fig. 1b) are too regular, lacking of flexibility and diversity.

Intersectional traffic simulation often needs to handle heterogeneous vehicles mixed with pedestrians. Some human trajectory prediction methods (e.g., [11], [12], [13], [14], [15]) in crowded space treat each object as a point in the model. Various deep learning models learn human-human interactions based on hidden states in the network and assume that each object has similar behavior patterns. However, the dynamic behaviors of vehicles and those of pedestrians are significantly different, in particular, when crossing intersections. Therefore, the above methods cannot be directly applied for intersectional traffic simulations without considerable efforts.

On the one hand, road intersections play an important role in real-world traffic phenomena; on the other hand, the difficulty of simulating realistic intersectional traffic has been well recognized in the community. Arguably, the main reasons for such a challenge are: (i) Due to the lack of clearly defined lanes, vehicles in intersectional traffic have more flexible trajectories than those driving along well-defined, unidirectional lanes. (ii) Intersectional traffic simulations involve many more dynamic environment factors, including heterogeneous vehicles mixed with pedestrians, than freeway traffic simulations.

Inspired by the above challenges, we propose a novel deep learning-based framework to simulate and edit intersectional traffic (refer to Fig. 1c). Specifically, we first construct a high-quality intersectional traffic dataset. Then, our intersectional traffic simulation framework consists of two main stages. (i) At the first stage, we process the inter'-sectional traffic data and convert it into a compact yet effective representation for deep learning, including the introduction of a novel grid coordinate system to encode dynamic intersectional environment and vehicle-vehicle interactions. With respect to the complexity of the involved

- *H. Bi is with the University of Chinese Academy of Sciences, Huairou, Beijing 100049, China, the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Haidian, Beijing 100190, China, and also with the Computer Graphics and Interactive Media Lab, University of Houston, Houston, TX 77204. E-mail: xiaobi361@gmail.com.*
- *T. Mao and Z. Wang are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Haidian, Beijing 100190, China. E-mail: {ltm, zqwang}@ict.ac.cn.*
- *Z. Deng is with the Computer Science Department, University of Houston, Houston, TX 77004. E-mail: zdeng4@uh.edu.*

Manuscript received 1 Sept. 2018; revised 18 Dec. 2018; accepted 21 Dec. 2018. Date of publication 3 Jan. 2019; date of current version 1 June 2020. (Corresponding author: Zhigang Deng.)
Recommended for acceptance by J. Lee.
Digital Object Identifier no. 10.1109/TVCG.2018.2889834

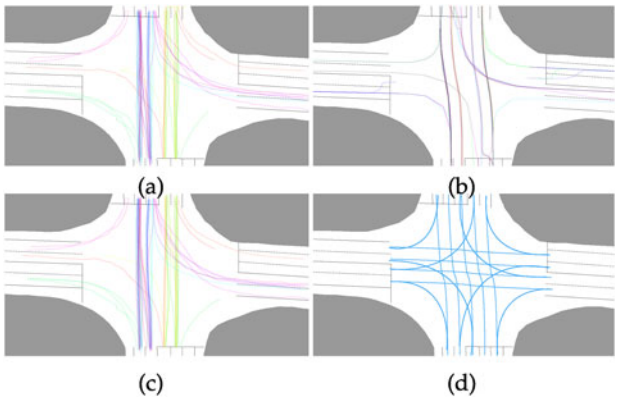


Fig. 1. (a) The ground truth data of traffic trajectories in an intersection. (b) An example of intersectional traffic simulation by the well-known traffic simulator SUMO [10]. (c) The traffic trajectories generated by our approach. (d) Pre-defined lanes driving along by vehicles in SUMO. Different trajectories are plotted with different colors.

vehicles, our representation can handle heterogeneous vehicles mixed with pedestrians. (ii) At the second stage, we employ the combination of convolution neural network (CNN) and recurrent neural network (RNN) to learn the Forward Difference (FD) velocities of vehicle trajectories in intersectional traffic. The trajectories learned from our model are more realistic and collision-free. Based on the predicted FD velocities of each vehicle, we further use a geometric method to obtain its Central Difference (CD) velocities.

Through many experiments as well as comparative user studies, we demonstrate that our approach can generate more realistic traffic in intersections than existing traffic simulators. Besides simulating novel intersectional traffic, we also demonstrate how our method can be used to edit existing intersectional traffic, including traffic simulation in intersections with new terrain and the trajectories of selected vehicles. Fig. 2 shows some example results by our approach: a 3D intersectional traffic simulation with a street view and an edited traffic animation with a bird's-eye view.

To the best of our knowledge, this work is the first reported system to simulate intersectional traffic without lanes and handle vehicle trajectories mixed with pedestrians in intersectional areas. The specific contributions of this work can be summarized as follows:

- A new data representation tailored for deep learning is introduced to encode heterogeneous vehicles mixed with pedestrians in intersectional traffic, which embeds vehicle-vehicle interactions.
- A deep learning-based framework is proposed to learn vehicle trajectory patterns in intersections.
- A new user editing technique is introduced to effectively edit specific vehicles in intersections.

2 Related Work

In this section, we first survey the existing rule-based and data-driven traffic simulation methods. Next, we review recent research efforts on intersectional traffic simulation. Finally, we survey human trajectory prediction methods with deep learning networks.

2.1 Traffic Simulation

Due to the ubiquity of traffic in the real world, many traffic simulation techniques have been proposed during the past decades. Generally speaking, there are three types of traffic simulations based on the level of simulation details: *microscopic*, *mesoscopic*, and *macroscopic*. Also known as agent-based methods, microscopic simulation treats each vehicle as a discrete autonomous agent with pre-defined rules [16], [17], [18], [19]. In particular, the Intelligent Driver Model (IDM) [6] and lane-changing model [20] are two notable methods. Existing microscopic methods have been recently improved to generate more flexible continuous traffic flow [5], [21]. Mesoscopic methods use Boltzmann-type mesoscale equations to simulate traffic dynamics [18], where traffic flow is viewed as continuum dynamics like fluid or gas. Therefore, nonlinear scalar conservation law or other second-order systems of equations derived from the equations of gas dynamics can be used to describe the regulations of vehicles [2], [17], [22], [23], [24]. A recent extension of macroscopic methods [3] can animate detailed traffic flow. However, these techniques have been primarily focused on traffic simulation on freeways. As a result, most of the traffic simulations studied by them are the decision-making of acceleration/deceleration in car-following or lane-changing scenarios.

Ignoring context-adaptive characteristics during driving, the rule-based methods mentioned above often fall short of simulating realistic traffic due to the inherent simplicity of pre-defined rules. To address the limitation, researchers have explored data-driven techniques in traffic simulation in recent years [25], [26], [27], [28], [29], [30]. With the concept of "virtualized traffic", Sewall et al. [31] reconstruct traffic flow with individual-specific driving characteristics learned

Fig. 2. Example results by our approach: 3D intersectional traffic simulation with a street view (left) and the edited traffic with a bird's-eye view (right).

from spatio-temporal data acquired with sensors. Wilkie et al. [4] use an agent-based traffic simulator to generate traffic animations to match the preset sparse traffic conditions statistically. A video-based approach by learning the specific driving characteristics of drivers was proposed in [32]. The works of [33], [34] reconstruct and visualize city-scale traffic animations through statistical learning. In the work of [35], a data-driven framework is introduced to perform the decision-making and execution processes for lane-changing in traffic simulations. In addition, some recent efforts have been conducted to simulate mixed traffic (i.e., mixing vehicles/motorcycles or pedestrians) [36], [37].

Compared with traffic simulations on freeways, very few studies on intersectional traffic simulation have been conducted partially due to the lack of any publicly available intersectional traffic datasets. Specifically, in the few existing works of [8], [9], [10], vehicle trajectories through intersectional areas need to be pre-defined, and signal-controlled queues are built for vehicles to drive along with different directions. Not surprisingly, they fall short of generating realistic intersectional traffic due to their fundamental dependence on empirically pre-defined rules.

2.2 Crowd Editing

In recent years many approaches have been proposed to interactively edit crowds or the motions of multi-characters. Kwon et al. [38] use a graph structure to edit crowd groups, where the vertices denote the positions of individuals at specific frames, and the edges encode neighborhood formations and moving trajectories. Later, researchers further extended this technique to interactively manipulate the synchronized motions of multi-characters in more complex scenarios [39], [40]. Kulpa et al. [41] proposed the imperceptible relaxation of collision avoidance constraints for interactive virtual crowd editing. Kim et al. [42] employ a cage-based method for large-scale crowd editing, where animators are allowed to edit existing crowd animations intuitively with real-time performance while maintaining complex interactions between individuals. A novel local interaction algorithm with a new context-aware, probabilistic motion prediction model for crowd simulators is proposed in [43]. Karamouzas et al. [44] proposed a simple and effective optimization-based integration scheme for implicit integration and apply this to crowd simulations. Whether and how the above crowd editing methods can be robustly generalized to traffic animation editing (in particular, intersectional traffic animation) has not been experimented and validated.

2.3 Human Trajectory Prediction

Predicting human motion behavior is a critical yet challenging task in the domains of video understanding and autonomous driving. The key problem that human trajectory prediction in a crowded space needs to handle is to analyze human-human interactions. The works of [45], [46] use Gaussian Processes and Bayesian nonparametric approaches, respectively, to learn the motion trajectories of pedestrians in the video. These learned motion trajectories avoid obstacles but ignore human-human interactions. By inferring traversable regions using semantic scene information, Kitani et al. [47] forecast the future trajectories of pedestrians. Alahi et al. [12] proposed a social pooling layer to encode human-human interactions to predict human trajectories. This work has been extended in [48], [49] to include static obstacles and dynamic agents. But only the dynamic agents in the neighborhood are considered. Fernando et al. [50] use the Long Sort Term Memory (LSTM) framework to predict human trajectories and consider all agents in the environment. Gupta et al. [14] proposed *social-GAN*, a recurrent sequence-to-sequence model, to predict the trajectories of pedestrians. In their method, a novel pooling mechanism is introduced in the network to aggregate information across people. However, all the above human trajectory prediction works treat each subject as a point and assume that each subject during prediction has similar behavior patterns. By contrast, the behaviors of vehicles in an intersection react not only to heterogeneous vehicles but also to pedestrians. As a result, these methods cannot be directly applied to intersectional traffic simulations.

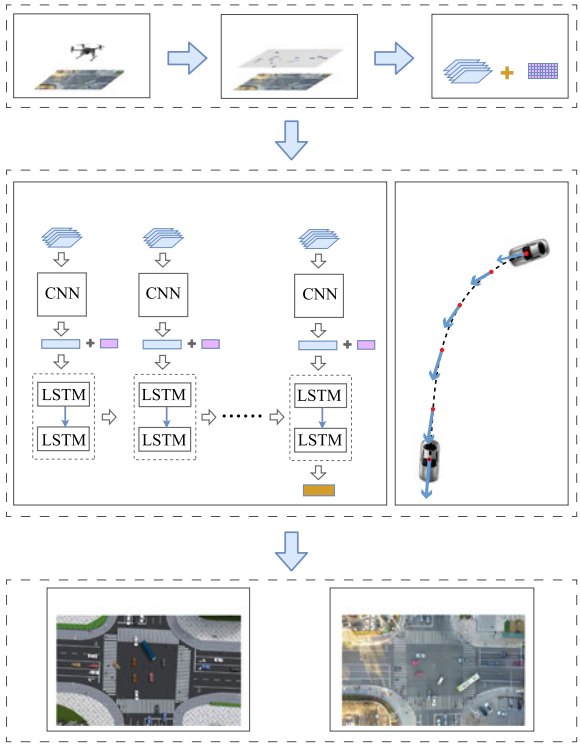


Fig. 3. Schematic illustration of our deep learning based framework for intersectional traffic simulation and editing.

3 Our Approach Overview

Our framework consists of three main steps: data acquisition and processing, trajectory learning, and simulation process described below. Fig. 3 shows the schematic view of our framework.

Data Acquisition and Processing. After intersectional traffic video data acquisition and processing (detailed in Section 4.1), we further convert the data to a format particularly tailored for deep learning (Section 4.2). Different from traffic simulations on freeways, the vehicles in an intersection do not have strictly-defined lanes to follow and have a high readiness to respond to potential unanticipated events. The characterization of heterogeneous vehicles mixed with pedestrians also

makes various existing traffic data representations that are originally designed for freeway traffic simulations unsuitable for our work. In order to address this issue, we introduce a grid coordinate system called *grid map* to encode the relevant driving information of heterogeneous vehicles (Section 4.2). A window with five channels sliding on the grid map can generate an environment matrix for each vehicle. In addition, the destination of a vehicle and its current driving states together are called *vehicle identity* (Section 4.2), which can also affect its trajectory passing through an intersection. The environment matrices together with the vehicle identities are inputs to the following modules.

Trajectory Learning. Based on the above inputs, we first build a CNN with three convolution layers and three pooling layers (Section 5.1) to reduce the dimension of the environment matrices. Then, with the environment feature vectors outputted from the CNN as one of the inputs, we train a two-stack-layers RNN, with LSTM as the memory cell to store the long term task-relevant data, to learn the FD velocities of vehicle trajectories (Section 5.2). Besides the FD velocities in trajectory, the CD velocities of each vehicle also need to be considered. Due to physical rules, each vehicle cannot drive towards any direction instantly. In this work, we choose to use the tangential vector of the vehicle's historical driving path to approximate its current CD velocities (detailed in Section 5.4).

Traffic Simulation and Editing. After the above deep learning model is learned (Section 5.3), we can simulate and edit intersectional traffic (Section 5.5). Given the initial states of an intersection, our framework can simulate the traffic until all the vehicles drive through the intersection. Also, by allowing users to edit vehicles (e.g., add new vehicles, remove existing vehicles, or modify the trajectories of some vehicles), our method can be directly used to generate new traffic simulations in the intersection. In this process, the trajectories of the edited vehicles and those of the neighboring vehicles are re-calculated by our method.

4 Traffic Video Data Acquisition

We recorded a video dataset containing vehicles' trajectories at a city intersection. The intersectional area was approximately 96 meters $\times$ 54 meters. We used a drone to hover at 72 meters above the intersection, as statically as possible, to record vehicles and pedestrians passing through the area from a top-down view. A total of 2,611 seconds of video data with a 30 fps frame rate was acquired.

4.1 Trajectory Data from Traffic Video

In order to obtain accurate vehicle trajectories from the acquired video data, we employed a visual tracking algorithm, the Multi-Domain Convolutional Neural Networks (MDNet) [51], to track the locations of vehicles and pedestrians in each video frame. We used the center of the tracking box of each vehicle or pedestrian as its location. After the automated video tracking, we also manually checked and corrected any mis-tracked places for every frame to ensure the data quality. The size of the vehicles that are specifically studied in this work is approximately 4.5 meters in length and 2 meters in width, since such vehicles (e.g., sedans and sports utility vehicles) widely exist in real-world daily traffic. Other vehicles including buses and articulated buses are not considered in this work due to their significantly different driving patterns in intersections. To this end, the obtained dataset contains 26,110 frames of vehicle trajectory data, down-sampled to 10 frames per second.

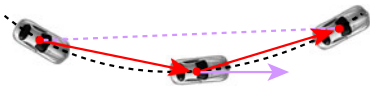


Fig. 4. Illustration of the FD velocity V and CD velocity Δ of a vehicle. The black dashed line is the path of vehicle at steps $t-1$, t, and $t+1$. The position of the vehicle at each step is denoted as $L=(L_x, L_y)$. So the FD velocity of the vehicle at t is discretely calculated as $V^t = L^{t+1} - L^t$ (red arrow). In order to generate smooth trajectories, we further use Gaussian smoothing to remove potential tracking noise and high frequency information. The position of the vehicle after smooth at each step is denoted as $\hat{L}=(\hat{L}_x, \hat{L}_y)$. And the CD velocity of the vehicle at t is $\Delta^t = \hat{L}^{t+1} - \hat{L}^{t-1}$ (purple arrow).

Also, traffic light plays a key role in traffic intersections. However, the goal of this work is to learn the driving patterns of vehicles with the assumptions of the no-lane-constraint and heterogeneous traffic agents. The traffic light can be treated as a separate constraint in our simulations. Following the traffic rules, vehicles at a red light are forced to stop. In order to ensure data quality, we chop the traffic video into fragments where vehicles passed through intersections with the same traffic light conditions. The trajectories of all the vehicles are labeled. And we only learn the trajectories of vehicles whose driving behaviors are only affected by the driving environment and under green traffic lights. Those vehicles affected by traffic lights or some other self-factors (e.g., the stopping of a taxi due to the loading/unloading of passengers) are not considered, and instead they are considered as the known inputs in our model.

To this end, in our processed dataset, we have 1,347 vehicles driving straight, 426 vehicles turning left, and 471 vehicles turning right. To counterbalance different driving directions, we rotated the data for the four possible incoming driving directions to obtain 4 times of data for our deep learning-based framework. Because of the different driving patterns of the vehicles passing through intersections and the non-uniform distribution of the data, we classify all the vehicle trajectories into three classes: drive straight, turn right, and turn left. For each vehicle trajectory, we use Gaussian smoothing to remove tracking noise and high frequency information.

4.2 Data Representation for Deep Learning

Based on the above vehicle trajectories, we need to first convert them to a data representation that is particularly suitable for deep learning. For traffic simulation on freeways, each vehicle needs to calculate its velocity and acceleration at each frame. Different from the vehicles that drive along explicitly pre-defined lanes, those vehicles passing through intersections typically do not drive on explicitly pre-defined lanes. Based on the discrete vehicle trajectory data, in this paper, we use the FD velocity (denoted as V) and the CD velocity (denoted as Δ) to describe the detailed movement of a vehicle (refer to Fig. 4). Here, the FD velocity (the red arrows in Fig. 4) is used to calculate the next position that

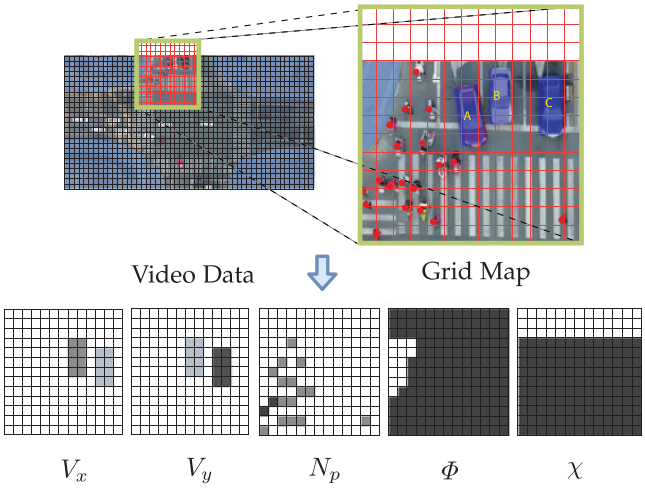


Fig. 5. Illustration of the Grid Map in this work. For O_A, a window with size 31 $\times$ 31 is used to describe the surrounding area. We build an environment matrix $M = (V_x, V_y, N_p, \Phi, \chi)$ including five channels. The grayscale of a grid in V_x (or V_y) visualizes v_x (or v_y) of O_B and O_C. The grayscale of a grid in N_p denotes the number of pedestrians and bicycles. The grids in Φ and χ represent the area into which O_A can drive and the visible area from the drone's perspective, respectively. The black color of a grid means its value is 1.

the vehicle drives towards based on the current position. The CD velocity (the purple arrows in Fig. 4) is used to represent the approximated tangential vector of the driving path at each step. With such representations, the FD velocity ensures the details of movement, such as collision avoidance. On the other hand, the CD velocity ensures the smoothness of the simulated trajectory.

Environment Matrix. Visual depth maps have been employed in various applications with deep learning techniques [52], [53], [54]. Analogously, the vehicles passing through an intersection rely on the visual perception of the objects (e.g., vehicles, pedestrians, and obstacles) in the neighborhood, such as their velocities and positions. Inspired by the concept of visual depth maps and the characteristics of vehicle-environment interactions, we introduce a grid coordinate system, called *grid map*, in intersections to encode the interactions between heterogeneous vehicles mixed with pedestrians. As illustrated in Fig. 5, a trajectory in the grid map is in XY-plane, and we map all the trajectories to the corresponding grid map. With the aid of the grid map, searching for neighboring objects (i.e., vehicles, pedestrians, or obstacles) of a vehicle can be efficiently done. At each step, in order to describe the driving environment around a vehicle, an environment matrix $M = (V_x, V_y, N_p, \Phi, \chi)$ is built based on the grid map of the vehicle. The environment matrix consists of five channels, each of which has a resolution of 31 $\times$ 31 and occupies an area of approximately 31 $\times$ 31 meters. Each channel extends 15 grids along X-axis and Y-axis bilaterally from the grid where the vehicle's center is located. Specifically, we use O_A to denote vehicle A. Its first two channels V_x and V_y are the FD velocity along the lateral axis (X-axis) and the FD velocity along the vertical axis (Y-axis), respectively. N_p indicates the numbers of pedestrians and bicycles in each grid. Φ represents the road grid map around O_A. The value of a grid is set to 1 if the vehicle can drive into it. χ represents the visible area: the value of a grid is set to 1 if it is visible from the drone's perspective. Through the environment matrices, original vehicle-vehicle/pedestrian interactions and vehicle kinematics are together encoded.

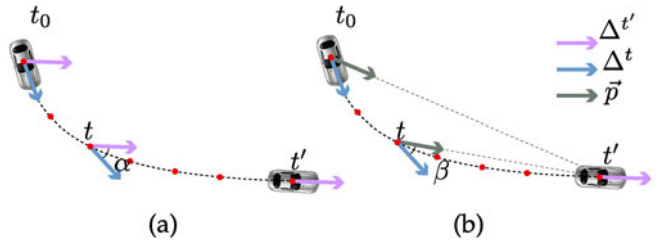


Fig. 6. Illustration of the CD velocity of a vehicle. (a) and (b) are the schematic view of α and β. The blue arrows denote the CD velocity at each step. t' is the last step along the path, and the purple arrows denote the CD velocity at t'. The green arrows $\vec{p}$ are the vectors from the position of the vehicle at a step t to its position at t'. α is the angle between the CD velocity at t and the CD velocity at t'. β is the angle between the CD velocity at t and vector $\vec{p}$

Vehicle Identity. When a vehicle drives through an intersection, the destination of its trajectory is its location at the last step. For O_A, we use vehicle identity $I = (v_x, v_y, cos\alpha, cos\beta)$ to represent the vehicle's driving information towards the destination at step t. v_x and v_y represent the FD velocities along the lateral axis (X-axis) and along the vertical axis (Y-axis) at step t, respectively. We also define α and β to describe the relationship between the CD velocity Δ and motion. As illustrated in Fig. 6, α is the angle between the CD velocity at t and the CD velocity at the final step t'. β is the angle between the CD velocity and the vector $\vec{p}$ (Fig. 6) from the position of the vehicle at t to the destination of the trajectory at t'.

Driving State. For O_A, its driving state $S^t = (M, I)$ at step t consists of its environment matrix M and vehicle identity I.

Vehicle Trajectory. The trajectory status of O_A at step t: $\Gamma^t = (V^t, \Delta^t)$, where V^t and Δ^t respectively denote the FD velocity and the CD velocity of the vehicle at step t. Here we define $V^t = (v_x^t, v_y^t)$ and $\Delta^t = (\Delta_x^t, \Delta_y^t)$. The trajectory status at t is decided by the historical driving states.

Therefore, the objective of our framework is to learn a mapping function f from the driving states to the vehicle trajectory, described as follows:

$$\Gamma^t = f(S^{t-1}, S^{t-2}, \ldots). \quad (1)$$

5 Trajectory Learning

Due to the different driving patterns of the vehicles passing through intersections, our framework handles the cases of driving straight, turning right, and turning left, separately. For each case, the architecture of the neural network is the same; only the training data are different.

In order to learn trajectories based on the driving state $S^t = (M, I)$ at step t, we build a network consisting of CNN-RNN: the CNN is used to extract features from the environment matrix M, and the RNN is designed to learn the FD velocities of the trajectories from sequence data. After training the networks, we use a geometric method to calculate the CD velocities of each vehicle.

5.1 CNN Network Structure

To construct the environment features for trajectory learning, we use a CNN to extract features (denoted as ϵ) from the environment matrix M. Specifically, the CNN consists of three convolutional layers and three pooling layers. Each

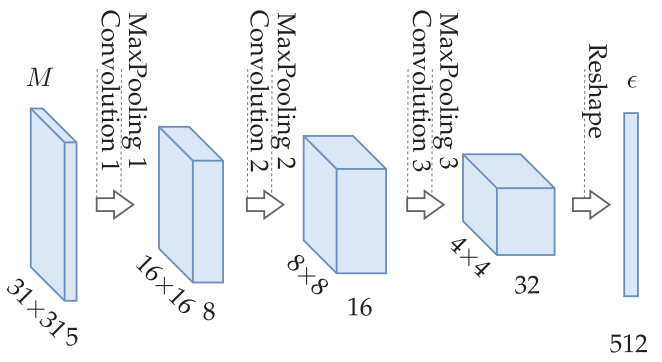


Fig. 7. Illustration of the used CNN architecture. Convolution 1 contains 8 filters of size 5×5. Convolution 2 contains 16 filters of size 5×5. Convolution 3 contains 32 filters of size 5×5. The input to Convolution 1 is M with size $31 \times 31 \times 5$. After the Convolution 1 and Max Pooling 1, its size becomes $16 \times 16 \times 8$. After the Convolution 2 and Max Pooling 2, its size becomes $8 \times 8 \times 16$, and after the Convolution 3 and Max Pooling 3, its size becomes $4 \times 4 \times 32$. Finally, we reshape the output of Convolution 3 and Max Pooling 3 to a 1×512 vector.

convolutional layer performs a one-dimensional convolution over the matrix independently. The input of the CNN is the environment matrix $M = (V_x, V_y, N_p, \Phi, \chi)$ and the output of the CNN is the environment features ϵ that contain the current driving environmental information.

Fig. 7 illustrates the architecture of the used CNN. At each step, the CNN repeatedly applies convolution and max pooling onto the environment matrix $M = (V_x, V_y, N_p, \Phi, \chi)$ from bottom to top. One convolutional layer and max pooling layer can be described as

$$\Phi(X) = \sigma_r(\Psi(X \circledast W^{\dagger} + b^{\dagger})), \tag{2}$$

where $W^{\dagger} \in \mathbb{R}^{l \times n}$ is the weights matrix initialized to some small random values, $b^{\dagger} \in \mathbb{R}^n$ is the bias initialized to zero, l and n represent the numbers of units in two adjacent layers, $\circledast$ is a convolution operation on X, and Ψ denotes a max pooling operation that is used to reduce the dimensionality of the output from the previous convolutional layer. Ψ takes the maximum of each adjacent pair grids. $\sigma_r(*)$ is a nonlinear operation $ReLu(*) = max(0, *)$. To this end, the environment matrix M is transformed and reduced to $4 \times 4 \times 32$, which is further reshaped to a 1×512 vector.

5.2 RNN Network Structure

To generate the FD velocities of the vehicle trajectory, we use a RNN on top of the combination of the environment features ϵ and vehicle identity I. As illustrated in Fig. 8, the used RNN has two stack layers. At time step t, the driving state $S^t = (M, I)$ after CNN is transformed to (ϵ, I). For a sequence of ϵ and a sequence of I, we can easily map two sequences and build a fully connected layer on top of it. We take the output from the fully connected layer as the input to the RNN.

We employ the LSTM [55] as the basic memory cell of each layer in RNN. Each cell is explicitly controlled via an input gate, an output gate, and a forget gate. The network employing memory cells overcomes the vanishing and exploding gradients in traditional RNNs. It also facilitates the long term storage of task-relevant data. Previously, LSTM has been successfully used for human motion prediction and natural language processing [56], [57], [58].

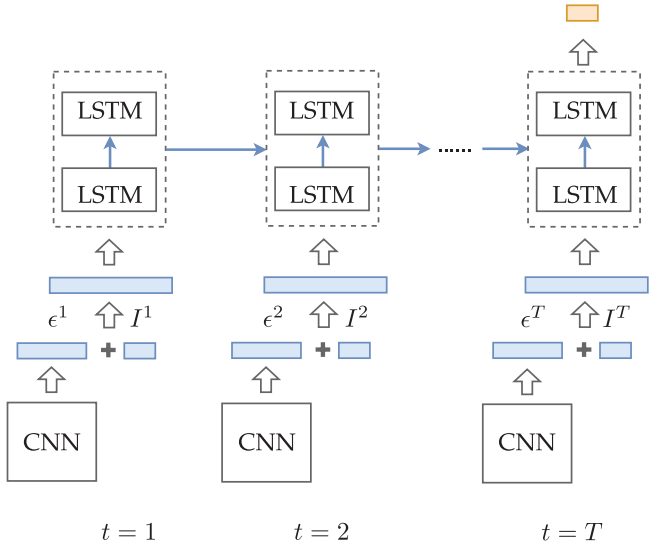


Fig. 8. Illustration of the used RNN architecture. At each step t, the shape of ϵ^t is the 1×512 vector from CNN. Together with I^t whose size is 1×4, a fully connected layer is built on top of them to generate the input of RNN. After a two-stack-layers RNN, the output from LSTM cell at $t = T$, is utilized as an input to a fully connected layer. After fully connected layer its size becomes 1×2, namely V^{T+1}.

At step t, we denote the input and output of a memory cell in one LSTM as $C_I^t \in \mathbb{R}^h$ and $C_O^t \in \mathbb{R}^h$. We use $X^t \in \mathbb{R}^d$ and $H^t \in \mathbb{R}^h$ to represent the input and output of one LSTM. All the equations we employ in a memory cell are as follows:

$$\text{For input gate}: G_I^t = \sigma_g(W_I X^t + U_I H^{t-1} + b_I), \tag{3}$$

$$\text{For forget gate}: G_F^t = \sigma_g(W_F X^t + U_F H^{t-1} + b_F), \tag{4}$$

$$\text{For output gate}: G_O^t = \sigma_g(W_O X^t + U_O H^{t-1} + b_O), \tag{5}$$

$$\text{For cell input}: C_I^t = \sigma_c(W_c X^t + U_c H^{t-1} + b_c), \tag{6}$$

$$\text{For cell output}: C_O^t = G_I^t \circ C_I^t + G_F^t \circ C_O^{t-1}, \tag{7}$$

$$\text{For output of LSTM}: H^t = G_O^t \circ \sigma_h(C_O^t). \tag{8}$$

Here $G_*^t \in \mathbb{R}^h$ (with $* \in \{I, F, O\}$), and $W_* \in \mathbb{R}^{h \times d}$ is a weights matrix that connects X^t to the $*$ gate and the cell input. $U_* \in \mathbb{R}^{h \times h}$ is a weights matrix that connects H^{t-1} to the $*$ gate and the cell input. $b_* \in \mathbb{R}^h$ is the bias of $*$ gate and the cell input. σ_g is a sigmoid function and both σ_h and σ_c represent hyperbolic tangent functions. The operator $\circ$ denotes the Hadamard product (entry-wise product). Specifically, the subscripts d and h refer to the number of input features and the number of hidden units, respectively. In our experiments, we choose $d = 64$ and $h = 84$.

At step $t = T$, the output of LSTM, H^T, is utilized as an input to a fully connected layer. The FD velocity V^{T+1} of the vehicle trajectory at step $T + 1$ can be calculated by

$$V^{T+1} = H^T W^{\circ} + b^{\circ}. \tag{9}$$

Here $W^{\circ} \in \mathbb{R}^{h \times s}$ is the weights matrix between the LSTM and the last fully connected layer, initialized to some small random values, $b^{\circ} \in \mathbb{R}^s$ is the bias of the last fully connected

layer, which is initialized to zero, and s denotes the dimension of V. We choose $T = 8$ and $s = 2$ in our experiments.

5.3 Training

We train the whole CNN-RNN together. The network learning process can be regarded as optimizing the following differentiable error function

$$E(W, b) = \frac{\Sigma_{k=1}^{N} E_k(W, b)}{N}, \tag{10}$$

where N represents the number of training data sequences, and W and b represent the weights matrix and bias in CNN-RNN, respectively. Our training criterion is to minimize the squared loss between the output V and the ground truth $\hat{V}$. Specifically, for an input sequence of driving states $S = (M, I)$, the loss function takes the following form:

$$E(W, b) = \frac{\Sigma_{k=1}^{N} E_k(W, b)}{N} = \frac{\Sigma_{k=1}^{N} \left\| V_k^T - \hat{V}_k^T \right\|^2}{N}. \tag{11}$$

Iteratively, we calculate the error gradient and update the network. We optimize the network using stochastic gradient descent (SGD). The parameters in the training process include batch size (512), learning rate (0.001) and momentum (0.9). To prevent over-fitting, we use a dropout strategy [59] with the dropout ratio = 0.8. The optimization process continues until it converges or after 20,000 epochs. For each driving pattern of the vehicles passing through intersections, the vehicle trajectory sequence data are randomly divided into three groups: 70 percent for training, 15 percent for validation, and 15 percent for test. We train our networks on an off-the-shelf desktop computer with 2.7 GHz Intel Core I5-6400K CPU and a GeForce 1,060 GPU (6 GB memory). Our framework was implemented on the Tensorflow platform [60].

5.4 Generate the CD Velocities

Besides the above FD velocity V, we also need to calculate the CD velocity Δ for each vehicle. We use the tangent vector of the driving path to represent the current CD velocity. During our simulation, the position of a vehicle at $t+1$ is still unknown. Therefore, we assume that the CD velocities of the vehicle at two adjacent steps are sufficiently close due to the motion continuity. We use the CD velocity at the previous step as the CD velocity at the current step.

We first use a Gaussian smoothing filter to process the historical position data and remove high frequency information. Then, $\Delta^{t-1} = (\Delta_x^{t-1}, \Delta_y^{t-1})$ is calculated using the following equation (also see Fig. 4)

$$\Delta_x^{t-1} = \hat{L}_x^t - \hat{L}_x^{t-2}, \tag{12}$$

$$\Delta_y^{t-1} = \hat{L}_y^t - \hat{L}_y^{t-2}, \tag{13}$$

where $\hat{L}_x^t$ and $\hat{L}_y^t$ denote the position of the vehicle at step t along the lateral axis (X-axis) and the vertical axis (Y-axis) processed with Gaussian smoothing, respectively.

It is noteworthy that our current method does not consider any physical constraints of vehicles. Therefore, we manually add some control policies to address this issue: The FD velocity of a vehicle trajectory needs to satisfy the physical constraints of the maximum acceleration, maximum deceleration, and the steering limitation of the vehicle. Our environment matrices could encode the vehicle-vehicle/pedestrian interactions in the local neighborhood, which contributes to avoid collisions. More collision avoidance examples are presented in Section 6.2. Furthermore, if there is an inevitable collision, we will randomly select and stop one of the involved vehicles.

5.5 Simulation

Once the above framework is trained, we can use it to simulate intersectional traffic. Based on a provided driving state sequence as the initial condition, our method can simulate the detailed motions of the vehicles passing through intersections.

Algorithm 1 details the process of intersectional traffic simulation using our method. Given an initial vehicle set $\mathfrak{D}$ including the initial states of each vehicle, we first create a grid map for each simulate step t. Here, the initial states of each vehicle include arrival time, arrival position, arrival CD velocity, arrival FD velocity, departure position, and departure CD velocity. So each vehicle can be easily classified based on its arrival and departure positions. For each vehicle in $\mathfrak{D}$, we collect the sequence of driving states $S = (M, I)$ at previous T steps. After feeding CNN with the sequence of M (denoted as κ_M), we map the sequence of ϵ as the output of CNN with the sequence of I (denoted as κ_I). Then, we feed the mapping sequence to RNN and calculate the FD velocity V of the vehicle trajectory, which is further refined using the rules of maximum acceleration and maximum deceleration. Finally, we approximate the CD velocity Δ and write the trajectory $\Gamma = (V, \Delta)$ into the current grid map. Iteratively, we can simulate the whole scenario until all the vehicles in $\mathfrak{D}$ have driven through the intersection. Note that the step used in the simulation is the same as the sampling interval in our data (0.1 s).

Algorithm 1. Intersectional Traffic Simulation Process

Input: $\mathfrak{D}$;
create Grid Maps Π
$t \leftarrow T + 1$
While $\mathfrak{D}$ is not empty **do**
  for $O_i \in \mathfrak{D}$ **do**
    for $i = 0; i < T; i++$ **do**
      $\pi \leftarrow \Pi[t - i]$
      Search π for $M = \left(V_x, V_y, N_p, \Phi, \chi\right)$
      Search π for $I = \left(v_x, v_y, cos\alpha, cos\beta\right)$
      $\kappa_M[T - i] \leftarrow M$
      $\kappa_I[T - i] \leftarrow I$
    end for
    choose a CNN-RNN network class τ
    $V \leftarrow$ feed the CNN-RNN network τ with κ_M and κ_I
    $V \leftarrow V$ after control policy is applied
    $\Delta \leftarrow$ CD velocity
    store V and Δ in π
    $t = t + 1$
    if O_i arrive destination **then**
      delete O_i from $\mathfrak{D}$
    end if
  end for
end while

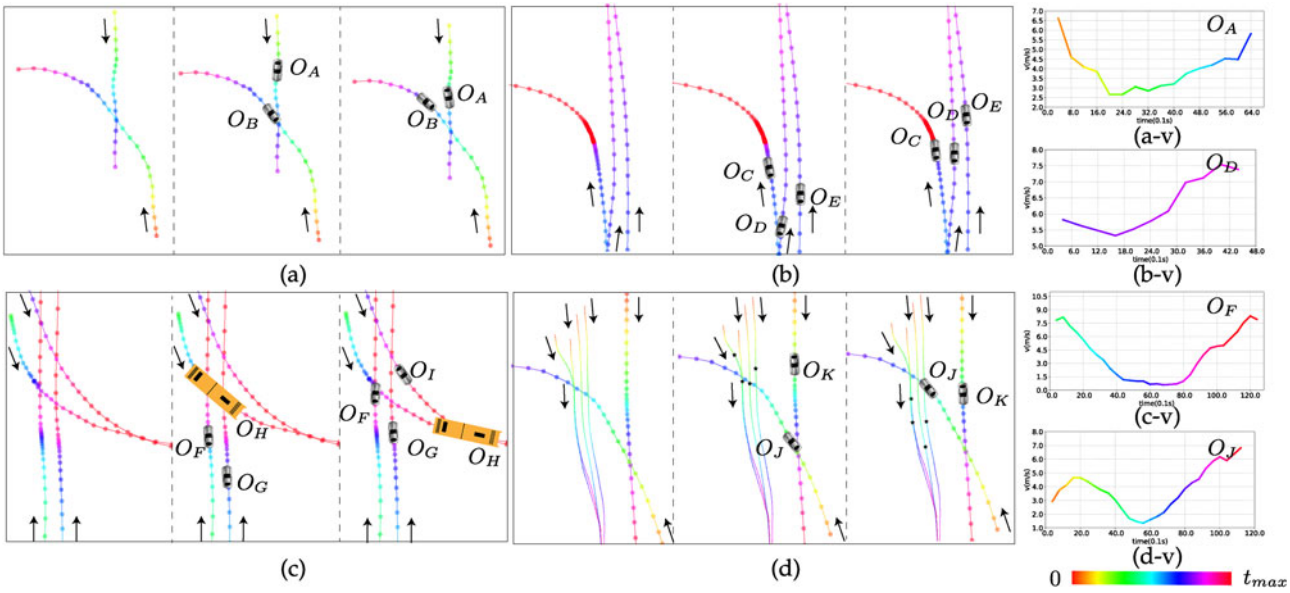


Fig. 9. Four collision avoidance examples by our method. The leftmost panel in each example shows the trajectories of all the vehicles. The middle and right panels show the relative positions of the vehicles at two specific steps. Each black arrow indicates the driving direction of a vehicle. Different color dots represent different steps. Dotted lines are the trajectories of vehicles and solid lines are the trajectories of pedestrians. O_H is a bus and $*$ denotes the specific step of a pedestrian. (a-v), (b-v), (c-v) and (d-v) plot the FD velocity curve of vehicles O_A, O_D, O_F and O_J, respectively.

6 Experimental Results and Analysis

In this section, we show some experimental results by our method. By specifying the trajectories of some pedestrians, a new intersectional traffic simulation result can be generated by our system. Furthermore, by specifying the terrain of the intersection and the initial states of vehicles, our method can also simulate traffic scenes with different terrains. The corresponding animation results are enclosed in the supplemental demo video, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2018.2889834.

Collision Avoidance. By encoding vehicle-vehicle/pedestrian interactions within the local neighborhood of each vehicle, our method can implicitly avoid most collisions between vehicles and between vehicles and pedestrians. Other collisions can be avoided through pre-defined physical rules. For example, in a typical simulation consisting of 77 vehicles and 727 frames, among a total of 7,547 time steps that need to be predicted, only 401 (5.3 percent) time steps are corrected with pre-defined physical rules. We show some examples in Fig. 9. The vehicle O_A in Fig. 9a drives straight and decelerates to avoid collisions. The vehicle O_D in Fig. 9b drives in the same lane as O_C. O_D initially drives with a relatively low FD velocity to avoid potential collisions with O_C. The vehicle O_F in Fig. 9c also decelerates to avoid collisions with the bus O_H and the coming vehicle O_I from the opposite direction. The vehicle O_J in Fig. 9d decelerates to avoid collisions with a group of pedestrians, and then accelerates towards the destination. These examples demonstrate that the synthesized vehicles by our method can effectively avoid collisions with other vehicles or pedestrians.

6.1 Runtime Performance

Table 1 shows the runtime performances of our method and the average runtime of generating the trajectory for a vehicle at one step. The majority of the computational time is used to search the grid map for M and I in order to learn V, and to calculate the CD velocity Δ based on the historical trajectories. We also utilized the mean absolute percentage error (MAPE) and root mean squared error (MSE) to measure the performance of our method.

In order to decrease the errors caused by non-uniform data distribution, we classified all vehicle trajectories into three categories and trained a different model for each category to predict the FD velocity. We also compared our individually-trained models with a global model that is trained using all vehicle trajectories. The comparison results are shown in Table 1. Clearly, the prediction errors of v_x and v_y by the individually-trained models are generally smaller than the global model. This is because the individually-trained models can better reduce the impact of non-uniform data distribution.

TABLE 1
Runtime Performances of Our Method

	RMSE				MAPE				TestData		Runtime	
	individual		integrated		individual		integrated		Sample Number	Frame Number	Training Time (hour)	Average Simulation Time (per vehicle per step, second)
	v_x	v_y	v_x	v_y	v_x	v_y	v_x	v_y				
Drive Straight	**0.0309**	**0.0289**	0.0417	0.0398	**0.0599**	**0.0673**	0.0709	0.0994	3667	77469	10.547	1.535
Turn Left	**0.0293**	**0.0304**	0.0394	0.0425	**0.0551**	**0.0679**	0.0693	0.1112	1500	47475	6.988	1.632
Turn Right	**0.0334**	**0.0348**	0.0478	0.0480	**0.0704**	**0.0881**	0.0892	0.1114	1633	30724	5.863	1.647

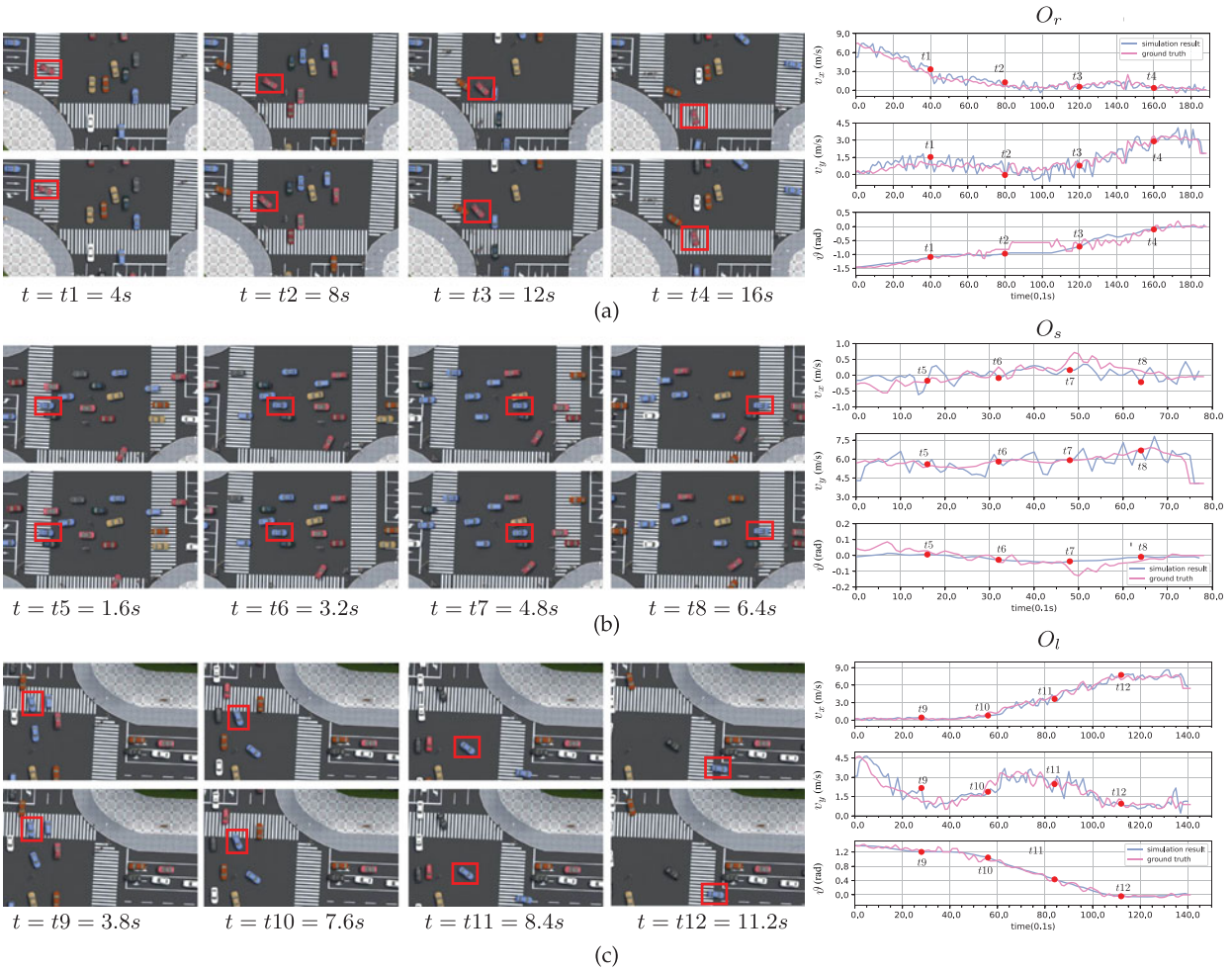


Fig. 10. Comparisons of the traffic trajectories simulated by our method and ground truth. (a), (b) and (c) separately show the trajectory of a vehicle that is turning right, driving straight, and turning left, denoted as O_r, O_s, O_l, respectively. In each paired comparison (i.e., (a), (b), or (c)), the top row shows the simulation result and the bottom row shows the ground truth trajectory. Also, the diagrams at the right compare the trajectories in v_x, v_y, ϑ between the simulation result and the ground truth.

As shown in Table 1, the computational efficiency of our approach is relatively low, mainly due to (1) the un-optimized and un-parallelized implementation of our approach, and (2) the relatively high computational cost of our approach. The major computational cost per vehicle includes GridMap construction and trajectory prediction for subsequent time steps, and the GridMap construction step is more computationally expensive than the trajectory prediction step. In order to predict detailed vehicle movements, we used a 31×31 GridMap, which has a relatively high granularity.

6.2 Comparisons with Ground Truth

In order to evaluate the effectiveness of our method, we compared the vehicle trajectories simulated by our method and the ground truth. Specifically, given the trajectory data of pedestrians and vehicles of other types (if any) as well as the initial driving states of vehicles of interest, our method can automatically generate intersectional traffic trajectories.

The trajectories of all three different driving patterns in intersections were compared. The vehicles turning right, driving straight, and turning left are denoted as O_r, O_s, O_l, respectively. Without loss of generality, we illustrate four specific steps in Fig. 10. We mainly compared the trajectories in v_x, v_y and ϑ. Here, We defined ϑ as

$$\vartheta = sign(\Delta^t \times \Delta^{t'}) \times \alpha. \tag{14}$$

ϑ is used to compare the simulated CD velocities of a vehicle with the ground truth. $\Delta^{t'}$ is its FD velocity at the last step. With the driving towards the destination in an intersection, ϑ is expected to gradually approach 0. Therefore, $sign(\Delta^t \times \Delta^{t'})$ can reflect its relative relationship with the destination at each step.

As shown in Fig. 10, compared with the ground truth, the FD velocities along X-axis and Y-axis (v_x, v_y) by our method are roughly close to the ground truth. Based on the driving environment at the current step, the vehicle can adjust its FD velocity in a timely manner. The simulated CD velocities also show similar patterns as the ground truth. For animation comparisons, please refer to the supplemental demo video, available online. It is noteworthy that our framework can only learn the trajectories of the vehicles whose behaviors are only affected by the driving environment and under green traffic lights. In order to better visually compare our method with the ground truth, the traffic light condition is set to be the same as the ground truth. The vehicles at a red light will be forced to stop. For example, from 0′49″ to 1′30″ in the supplemental demo video, available online, only the vehicles driving from the north/south, and the white

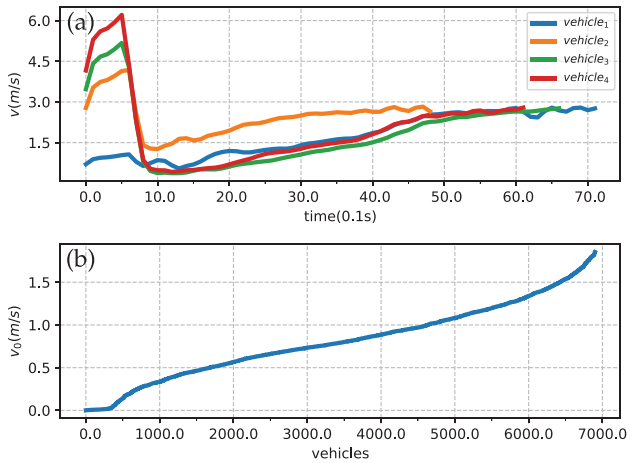


Fig. 11. (a) The predicted FD velocities with different arrival FD velocities in the same driving environment. (b) The statistical results of arrival FD velocities in the initial states of all the vehicles in the training dataset.

vehicle from the east at 0′51″ are simulated by our method. In addition, the trajectories of pedestrians, buses, articulated buses are taken from the original traffic video data and used as the known inputs to the grid map in our model for better visual comparisons.

Validation of the FD Velocities. We use the validation of the FD velocities to show the generalization of our model. Specifically, the arrival FD velocities in the initial states of all the vehicles are statistically shown in Fig. 11b. Clearly, all the arrival FD velocities are smaller than 2m/s. Without loss of generality, we choose $vehicle_1$ whose arrival FD velocity $v_0 = 0.703m/s$. Here, we separately set the arrival FD velocities $v_0' = 4v_0$, $v_0'' = 5v_0$, $v_0''' = 6v_0$ for $vehicle_2$, $vehicle_3$, $vehicle_4$, respectively. As shown in Fig. 11a, we compare the predicted FD velocities under the same driving environment. Although the arrival FD velocities in the initial states are beyond the range of our dataset, the FD velocities predicted by our method are still reasonable. This is because the predicted FD velocities partially depend on the grid map that captures the surrounding driving environment. For animation results of the validation of the FD velocities, please refer to the supplemental demo video, available online.

Length of the Generated Trajectory. Furthermore, we use average displacement error (ADE) to evaluate the generated trajectories with different lengths (T). ADE is the mean squared error between the points of a predicted trajectory and the points of the corresponding ground-truth trajectory. We conducted experiments for $T = 6, 8, 10, 12, 14$ separately. As shown in Fig. 12, for those vehicles that turn left or drive straight, the ADE performances are improved with the increase of the trajectory length, since more

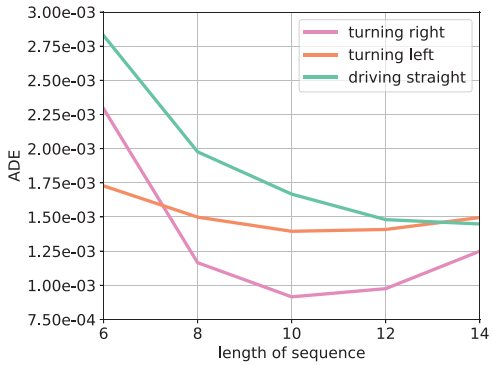


Fig. 12. ADE of the resulting vehicle trajectories by training our model with different lengths (T).

historical trajectory points can make better contributions to the trajectory prediction. For the case of the right turn, only when $T <= 10$, the ADE performance continues to be improved with the increase of the length. One sound explanation is that the path of right turn is often substantially shorter than those of left turn and straight driving, and thus it has fewer historical trajectory points for prediction.

6.3 Comparisons with Baselines

We further evaluated our method by quantitatively comparing it with four selected baseline methods that can be adapted for intersectional traffic simulation, including the well-known *SUMO* simulator [10], where the trajectories of the vehicles in an intersection are simulated with a car-following model along pre-defined lanes; the basic Vanilla LSTM (*V-LSTM*) [61], where each vehicle trajectory is predicted independently, without considering interactions; the Social LSTM (*S-LSTM*) [12], where each object is modeled via an LSTM with a social pooling layer to encode interactions; and the Social GAN (*S-GAN*) [14], which is a GAN based encoder-decoder framework with a pooling mechanism to learn social norms. We describe our comparison results below.

Quantitative Metrics. In order to quantitatively compare different methods, the first thing is to select quantitative metrics. In this comparison, we selected the following two metrics: (1) *Average Displacement Error (ADE)* to measure the averaged trajectory difference, and (2) *Final Displacement Error (FDE)* to measure the distance between the predicted destination and the ground-truth destination at the end of a vehicle trajectory. Table 2 shows the comparison results for three types of driving behaviors in an intersection (i.e., drive straight, turn left, and turn right) in terms of ADE and FDE.

Not surprisingly, the vehicle trajectories simulated by SUMO had high ADE and FDE errors because the vehicles were simulated along pre-defined lanes with a car-following

TABLE 2
Quantitative Comparisons of Our Method and the Four Baseline Methods

Method	SUMO		V-LSTM		S-LSTM		S-GAN		Ours	
Metric	ADE ($\times 10^{-2}$)	FDE ($\times 10^{-2}$)	ADE ($\times 10^{-2}$)	FDE ($\times 10^{-2}$)	ADE ($\times 10^{-2}$)	FDE ($\times 10^{-2}$)	ADE ($\times 10^{-2}$)	FDE ($\times 10^{-2}$)	ADE ($\times 10^{-2}$)	FDE ($\times 10^{-2}$)
Drive Straight	1.7703	3.5574	5.3580	7.9337	0.6922	1.3213	0.4651.	0.8370	**0.2319**	**0.3796**
Turn Left	2.3106	5.4953	6.6002	10.1907	0.7795	1.5775	0.3659	0.6011	**0.1991**	**0.5149**
Turn Right	2.6391	4.4936	5.4620	8.0288	1.1154	2.1969	0.3310	0.5986	**0.3164**	**0.4144**
Average	2.2400	4.5154	5.8067	8.7177	0.8627	1.6986	0.3873	0.6789	**0.2491**	**0.4363**

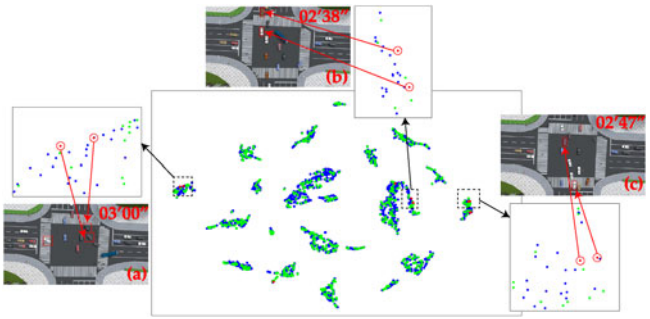


Fig. 13. The middle panel visualizes the high dimensional initial state data with t-SNE [62]. The blue scatter points represent the training dataset. The green scatter points represent the validation dataset and the test dataset. The red points represent the initial states of the corresponding edited vehicles in V#1. (a)(b)(c) Show some edited vehicles. The right top of (a)(b)(c) displays the corresponding time in the supplemental demo video, available online.

model, which significantly simplifies the real-world intersectional traffic. The vehicles driving straight and turning left/right in the SUMO model use pre-defined lanes to avoid collisions. If any potential collisions are in the range, the vehicles will decelerate or completely stop in current lanes, which is basically simplified to a queue system. Similarly, the V-LSTM method also generated relatively high prediction errors since it cannot effectively encode vehicle-vehicle/pedestrian interactions. The trajectory of each vehicle is predicted based on its own historical positions.

Both S-LSTM and S-GAN performed much better than SUMO and V-LSTM. To predict more complex trajectories, S-LSTM and S-GAN can capture certain interactions between vehicles and pedestrians. Our method performed better than both S-LSTM and S-GAN significantly for all three driving behaviors (turn left/right, and drive straight) in terms of both ADE and FDE because both S-LSTM and S-GAN assume all vehicles and pedestrians have similar behavior patterns. Both of them also ignore vehicle kinematics. By contrast, our method considers vehicle-vehicle/pedestrian interactions within the local neighborhood. Also, we input the relative-to-destination position of each vehicle to the network at each step, which helps to reduce the errors. By learning trajectories and the implicit kinematics of each vehicle driving in intersections from data, our method can produce more accurate trajectory predictions.

6.4 Intersectional Traffic Editing

Our method can also be straightforwardly used to edit existing intersectional traffic by modifying some vehicles' trajectories, adding new vehicles, or deleting some existing vehicles. Taking the modification of existing vehicles' trajectories as an example, the trajectory of a vehicle is decided by M and I. Given a new (user-specified) destination for a vehicle, at step t the angle between the vehicle's CD velocity and its CD velocity at the final step, α, will be changed accordingly. Also, β, the angle between the vehicle's CD velocity and vector $\vec{p}$ (refer to Fig. 6) will be modified according to different destinations. Simultaneously, the trajectories of the vehicles with new destinations will also affect the adjustments of the neighboring vehicles.

We selected two traffic video clips as test data (denoted as V#1 and V#2, respectively). V#1 has a total of 73 vehicles and

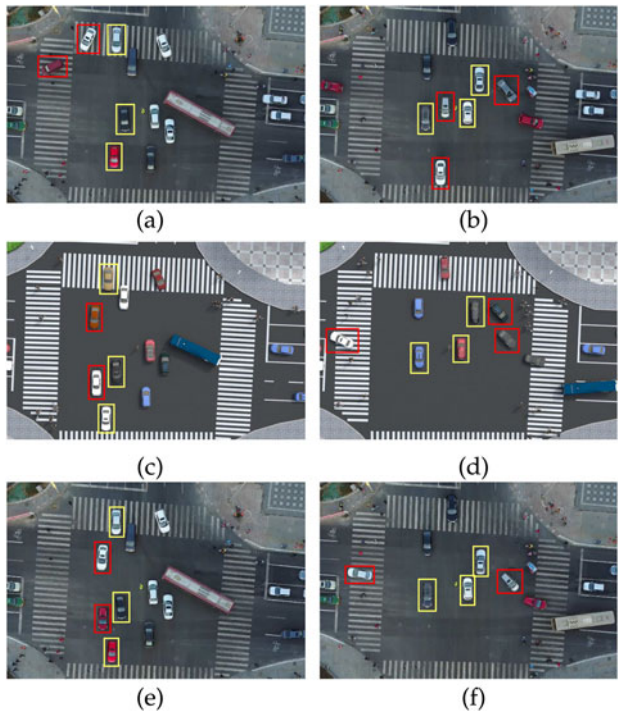


Fig. 14. Comparisons between the ground truth (top row) and the traffic editing results in 3D (middle row) and 2D scenes (bottom row). (a)(c)(e) and (b)(d)(f) are traffic snapshots at two selected steps randomly in V#1. The red rectangular boxes highlight the modified destinations of vehicles. The yellow rectangular boxes highlight the neighboring vehicles assigned with new trajectories.

V#2 has a total of 79 vehicles. To test our method, we modified randomly selected 11 vehicles' destinations in each of them. Since our method can automatically adjust the trajectories of neighboring vehicles, a total of 23 vehicles in V#1 were assigned with new trajectories and a total of 16 vehicles in V#2 were assigned with new trajectories. Some 2D and 3D edited animation results of V#1 are shown in Fig. 14. Based on the definition of the initial states of each vehicle in Section 5.5, we use t-SNE [62] to visualize the high dimensional initial states data in Fig. 13. Although some initial states do not exist in our dataset, our framework can predict trajectories according to the current driving environment. Refer to the supplemental demo video, available online, for the edited intersectional traffic results. Similar to the aforementioned intersectional traffic simulation, the same traffic light condition as the original traffic video is used for better visual comparisons between our edited result and the original video. The trajectories of pedestrians, buses, and articulated buses are taken from the original traffic video data and used as the known inputs to the grid map. For example, from 2'31" to 3'14" in the supplemental demo video, available online, only the vehicles driving from the north/south, the red vehicle from the west at 3'06" and the blue vehicle from the west at 3'08" are simulated by our method. In the two examples, we only edited the specific vehicle trajectories, and the surrounding vehicles will also change their trajectories accordingly.

6.5 Paired Comparison User Study

In order to further evaluate the effectiveness of our method, we designed a paired comparison user study. V#1 and V#2 are randomly split into three scenes respectively (denoted

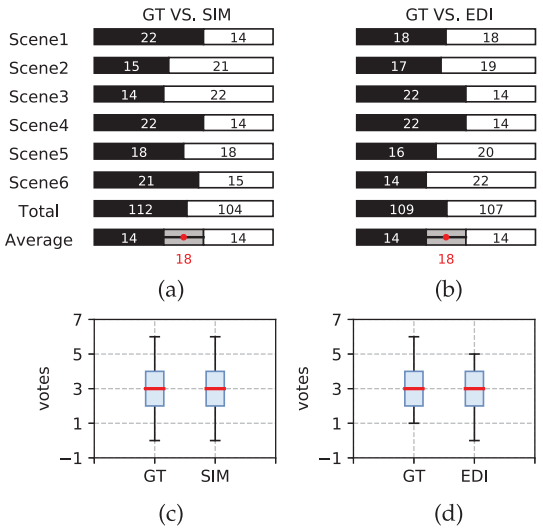


Fig. 15. The experimental results of our user study. In (a) and (b), the left black boxes and the right white boxes indicate the total number of participants who voted for the corresponding method: Ground Truth (GT), our Simulation results (SIM), and our Edited results (EDI). The error bars in the average boxes indicate the 95 percent CI of the total number of times when the participants voted the GT results. In (c) and (d), the box and whisker plots show the distribution of the number of scenes voted by each participant for the corresponding method.

as Scene1, Scene2, Scene3, Scene4, Scene5, and Scene6). For each scene, we generated 2 pairs of 3D traffic animation (i.e., ground truth versus traffic simulation result, and ground truth versus edited traffic result). To this end, we generated 18 intersectional traffic animations for this user study and constructed 12 comparison pairs.

We recruited 36 participants to participate in our paired comparison user study. All of them are graduate students in a university. Avoiding making forced and inaccurate perception votes, they are required to perceptually select which is the more realistic scene in each paired comparison, without limiting watching time. In order to balance carryover effects due to a small number of participants, we adopted the Williams design latin square [63] to display the pairs during the study. We use one-sample t-tests to determine the 95 percent Confidence Interval (CI) and paired-sample t-tests to compare the difference of the true mean of two sets of data with 95 percent confidence. The conventional significance for the entire analysis was determined at $\alpha = 0.05$, two tailed.

Fig. 15a shows the voting result of our user study for the 6 comparison pairs between the ground truth and the simulation results. A paired-sample t-test shows the difference of the true mean with 95 percent confidence between the ground truth and the simulation results is are not statistically significant ($p = 0.6656 > 0.05$). Fig. 15c shows the distribution of the number of scenes voted by each participant. The median of ground truth is measurably equal to that of simulation; the lower and upper quartile of ground truth and simulation are similar. This user study result indicates that the simulation results and ground truth are visually equivalent, to a large extent.

Similarly, Fig. 15b shows the voting result of our user study for 6 comparison pairs between ground truth and the edited traffic results. A paired-sample t-test shows the difference of the true mean with 95 percent confidence between ground truth and editing is not statistically significant ($p = 0.9049 > 0.05$). Fig. 15d shows the distribution of the number of scenes voted by each participant. The median of ground truth is measurably equal to that of the edited traffic results; the lower and upper quartile of ground truth and the edited results are highly similar. The user study result indicates that the edited traffic results and ground truth are perceptually indistinguishable.

7 Discussion and Conclusion

We present a new deep learning-based framework to simulate and edit traffic in intersections. A new data representation containing heterogeneous vehicles mixed with pedestrians is proposed and tailored for deep learning, which also captures the vehicle-vehicle/pedestrian interactions. Our representation has a higher granularity. Traffic is driven by the trajectories learned through CNN-RNN whose input includes the driving environment and the intentions of individual vehicles, defined as their environment matrices and vehicle identities, respectively. By specifying novel destinations and terrain, our method can be straightforwardly used to edit traffic in intersections. In order to validate our method, we compared our simulation and edited results with ground truths and found that our results are visually indistinguishable from the ground truth. Also, compared with existing human trajectory prediction models that could be adapted for intersectional traffic simulation, our method outperformed them in terms of quantitative metrics.

Our current method has a few limitations. First, the driving states of all previous steps are used as the inputs to generate the vehicle trajectories at the current step in the simulation process, which could lead to accumulated errors in long predicted sequences. Second, since as a purely data-driven method, our current method does not contain any physical implications; therefore, it may not be generalized sufficiently to handle certain complex intersectional traffic behaviors such as complex terrain in an intersection or crazy driving. In order to learn more accurate vehicle behavior patterns, we classify the vehicle trajectory data to driving straight, turning left, and turn right, and then learn corresponding different models to reduce the impact from the non-uniform distribution of the trajectory data. However, this may not be the optimal solution. Third, in our current method, pedestrians and other types of vehicles are simply treated as inputs during our simulation; any potential interactions between them are not considered.

There are a number of future directions to extend our current framework. Algorithmically, we would like to optimize and parallelize (via GPU or GPU+CPU) the current framework to reduce the simulation time and improve the accuracy of the simulated trajectories. Furthermore, a more generalized model to combine all driving behaviors including driving straight, turning left, and turning right, is necessary. Our current framework treats traffic light signals as a separate constraint. One potential solution is to extend the grid map representation to encode traffic light signals. Also, we plan to model the potential interaction among vehicles, pedestrians, and other traffic factors as future work. Lastly, we plan to build a complete traffic editing system to edit intersectional traffic with dynamically adjusting terrains.

ACKNOWLEDGMENTS

This work is in part supported by the National Key Research and Development Program of China (2017YFC0804900), the National Natural Science Foundation of China (61532002), the 13th Five-Year Common Technology pre Research Program (41402050301-170441402065), the Science and Technology Mobilization Program of Dongguan (KZ2017-06), and US NSF IIS 1524782. Huikun Bi is also supported by a CSC Fellowship.

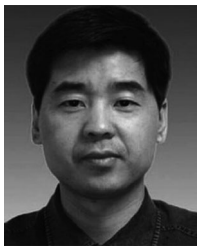
REFERENCES

[1] Uber, "Uber movement," 2017. [Online]. Available: https://movement.uber.com
[2] M. J. Lighthill and G. B. Whitham, "On kinematic waves. II. A theory of traffic flow on long crowded roads," *Proc. Roy. Soc. A: Math. Phys. Eng. Sci.*, vol. 229, no. 1178, pp. 317–345, 1955.
[3] J. Sewall, D. Wilkie, P. Merrell, and M. C. Lin, "Continuum traffic simulation," *Comput. Graph. Forum*, vol. 29, no. 2, pp. 439–448, 2010.
[4] 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, Jul. 2013.
[5] J. Shen and X. Jin, "Detailed traffic animation for urban road networks," *Graphical Models*, vol. 74, no. 5, pp. 265–282, 2012.
[6] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E*, vol. 62, no. 2, 2000, Art. no. 1805.
[7] A. Kesting and M. Treiber, "Calibrating car-following models by using trajectory data: Methodological study," *Transp. Res. Rec.: J. Transp. Res. Board*, vol. 2088, pp. 148–156, 2008.
[8] A. D'Ambrogio, G. Iazeolla, L. Pasini, and A. Pieroni, "Simulation model building of traffic intersections," *Simul. Model. Practice Theory*, vol. 17, no. 4, pp. 625–640, 2009.
[9] Mit intelligent transportation systems, 2011. [Online]. Available: its.mit.edu/
[10] K. Daniel, E. Jakob, B. Michael, and B. Laura, "Recent development and applications of sumo - simulation of urban mobility," *Int. J. Advances Syst. Meas.*, vol. 5, pp. 128–138, 2012.
[11] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2203–2210.
[12] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 961–971.
[13] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–7.
[14] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2255–2264.
[15] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5275–5284.
[16] D. L. Gerlough, *Simulation of Freeway Traffic on a General-Purpose Discrete Variable Computer*. Los Angeles, CA, USA: Univ. California, 1955.
[17] G. F. Newell, "Nonlinear effects in the dynamics of car following," *Operations Res.*, vol. 9, no. 2, pp. 209–229, 1961.
[18] K. Nagel and M. Schreckenberg, "A cellular automaton model for freeway traffic," *J. Physics I*, vol. 2, no. 2, pp. 2221–2229, 1992.
[19] M. Treiber and D. Helbing, "Microsimulations of freeway traffic including control measures," *Automatisierungstechnik*, vol. 49, pp. 478–484, 2002.
[20] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following model," *Transp. Res. Rec.*, vol. 1999, no. 1, pp. 86–94, 2007.
[21] T. Mao, H. Wang, Z. Deng, and Z. Wang, "An efficient lane model for complex traffic simulation," *Comput. Animation Virtual Worlds*, vol. 26, no. 3/4, pp. 397–403, 2015.
[22] V. Shvetsov and D. Helbing, "Macroscopic dynamics of multilane traffic," *Phys. Rev. E*, vol. 59, no. 6, 1999, Art. no. 6328.
[23] H. J. Payne, *Models of Freeway Traffic and Control*. Mathematical Models of Public Systems, Simulation Council, La Jolla, CA, 1971.
[24] G. B. Whitham, *Linear and Nonlinear Waves*. Hoboken, NJ, USA: Wiley, 2011.
[25] L. Chong, M. M. Abbas, and A. Medina, "Simulation of driver behavior with agent-based back-propagation neural network," *Transp. Res. Rec.: J. Transp. Res. Board*, vol. 2249, pp. 44–51, 2011.
[26] L. Chong, M. M. Abbas, A. MedinaFlintsch, and B. Higgs, "A rule-based neural network approach to model driver naturalistic behavior in traffic," *Transp. Res. Part C: Emerging Technol.*, vol. 32, pp. 207–223, 2013.
[27] Q. Meng and J. Weng, "Classification and regression tree approach for predicting drivers' merging behavior in short-term work zone merging areas," *J. Transp. Eng.*, vol. 138, no. 8, pp. 1062–1070, 2012.
[28] Y. Hou, P. Edara, and C. Sun, "Modeling mandatory lane changing using Bayes classifier and decision trees," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 2, pp. 647–655, Apr. 2014.
[29] 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," *Comput. Animation Virtual Worlds*, vol. 25, no. 1, pp. 83–97, 2014.
[30] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 2, pp. 1167–1178, Feb. 2018.
[31] J. Sewall, J. V. D. Berg, M. C. Lin, and D. Manocha, "Virtualized traffic: Reconstructing traffic flows from discrete spatiotemporal data," *IEEE Trans. Vis. Comput. Graph.*, vol. 17, no. 1, pp. 26–37, Jan. 2011.
[32] Q. Chao, J. Shen, and X. Jin, "Video-based personalized traffic learning," *Graphical Models*, vol. 75, no. 6, pp. 305–317, 2013.
[33] D. Wilkie, J. Sewall, W. Li, and M. C. Lin, "Virtualized traffic at metropolitan scales," *Frontiers Robot. AI*, vol. 2, no. 11, pp. 1–10, 2015.
[34] W. Li, D. Wolinski, and M. C. Lin, "City-scale traffic animation using statistical learning and metamodel-based optimization," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 200:1–200:12, Nov. 2017.
[35] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation*, 2016, pp. 149–158.
[36] Q. Chao, Z. Deng, and X. Jin, "Vehicle–pedestrian interaction for mixed traffic simulation," *Comput. Animation Virtual Worlds*, vol. 26, no. 3/4, pp. 405–412, 2015.
[37] W.-C. Lin, S.-K. Wong, C.-H. Li, and R. Tseng, "Generating believable mixed-traffic animation," *Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3171–3183, Nov. 2016.
[38] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi, "Group motion editing," *ACM Trans. Graph.*, vol. 27, no. 3, pp. 80:1–80:8, Aug. 2008.
[39] M. Kim, K. Hyun, J. Kim, and J. Lee, "Synchronized multi-character motion editing," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 79:1–79:9, Jul. 2009.
[40] E. S. L. Ho, T. Komura, and C.-L. Tai, "Spatial relationship preserving character motion adaptation," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 33:1–33:8, Jul. 2010.
[41] R. Kulpa, A.-H. Olivierxs, J. Ondřej, and J. Pettré, "Imperceptible relaxation of collision avoidance constraints in virtual crowds," *ACM Trans. Graph.*, vol. 30, no. 6, pp. 138:1–138:10, Dec. 2011.
[42] J. Kim, Y. Seol, T. Kwon, and J. Lee, "Interactive manipulation of large-scale crowd animation," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 83:1–83:10, Jul. 2014.
[43] D. Wolinski, M. C. Lin, and J. Pettré, "WarpDriver: Context-aware probabilistic motion prediction for crowd simulation," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 164:1–164:11, Nov. 2016.
[44] I. Karamouzas, N. Sohre, R. Narain, and S. J. Guy, "Implicit crowds: Optimization integrator for robust crowd simulation," *ACM Trans. Graph.*, vol. 36, no. 4, Jul. 2017, Art. no. 136.
[45] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1164–1171.
[46] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy, "A Bayesian nonparametric approach to modeling motion patterns," *Auton. Robots*, vol. 31, no. 4, 2011, Art. no. 383.
[47] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 201–214.
[48] D. Varshneya and G. Srinivasaraghavan, "Human trajectory prediction using spatially aware deep attention models," 2017, *arXiv:1705.09436*.
[49] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," in *Proc. 24th Int. Conf. Pattern Recog.*, 2018, pp. 1941–1946.

[50] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft+ hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," *Neural networks*, vol. 108, pp. 466–478, 2018.
[51] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 4293–4302.
[52] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2366–2374.
[53] Y. Kuznietsov, J. Stückler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6647–6655.
[54] L. He, G. Wang, and Z. Hu, "Learning depth from single images with deep neural network embedding focal length," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4676–4689, Sep. 2018.
[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[56] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised video summarization with adversarial LSTM networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2982–2991.
[57] J. Walker, K. Marino, A. Gupta, and M. Hebert, "The pose knows: Video forecasting by generating pose futures," in *Comput. Vis. (ICCV), 2017 IEEE Int. Conf.*, 2017, pp. 3352–3361.
[58] X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," *Proc. 54th Annual Meet. Assoc. Comput. Linguistics*, vol. 1, pp. 1064–1074, 2016.
[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[60] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/
[61] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, no. 5/6, pp. 602–610, 2005.
[62] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov., pp. 2579–2605, 2008.
[63] E. Williams, "Experimental designs balanced for the estimation of residual effects of treatments," *Australian J. Chemistry*, vol. 2, no. 2, pp. 149–168, 1949.

Huikun Bi received the BSc degree in information and computing science from Hebei University, China. She is working toward the PhD degree in the Institute of Computing Techinology, Chinese Academy of Sciences and University of Chinese Academy of Sciences. Her main research interests include traffic simulation and trajectory prediction in deep learning.

Tianlu Mao received the PhD degree from the Institute of Computing Techinology, Chinese Academy of Sciences, in 2009, where she is also working as associate professor. Her main scientific interests include the areas of crowd simulation and cloth animation.



Zhaoqi Wang is a researcher and a director of PhD students with the Institute of Computing Technology, Chinese Academy of Sciences. His research interests include virtual reality and intelligent human computer interaction. He is a senior member of the China Computer Federation.

Zhigang Deng received the BS degree in mathematics from Xiamen University, China, the MS degree in computer science from Peking University, China, and the PhD degree in computer science from the Department of Computer Science, University of Southern California, in 2006. He is a professor of computer science with the University of Houston. His research interests include computer graphics, computer animation, and HCI. He is a senior member of the IEEE.