DOI: 10.1111/cgf.14699
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Spatio-temporal Keyframe Control of Traffic Simulation using Coarse-to-Fine Optimization

Yi Han[1] He Wang[2] Xiaogang Jin[1]†

[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China
[2]University of Leeds, United Kingdom

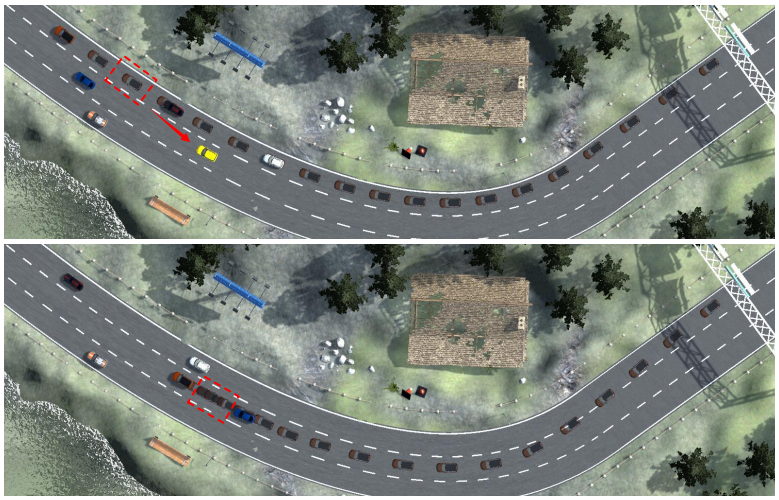

Figure 1: *An example of the controlled simulation with a keyframe. (a) The original trajectories. (b) The keyframe controlled trajectories. The vehicle drives along the rightmost lane originally. A keyframe is assigned in the center lane, which denotes that we want the vehicle to arrive at the position marked by the yellow vehicle at the time framed by the red box. As a result, a new reference path is planned through the position at first, and then the vehicle follows it to meet the spatial-temporal constraints smoothly.*

Abstract

We present a novel traffic trajectory editing method which uses spatio-temporal keyframes to control vehicles during the simulation to generate desired traffic trajectories. By taking self-motivation, path following and collision avoidance into account, the proposed force-based traffic simulation framework updates vehicle's motions in both the Frenet coordinates and the Cartesian coordinates. With the way-points from users, lane-level navigation can be generated by reference path planning. With a given keyframe, the coarse-to-fine optimization is proposed to efficiently generate the plausible trajectory which can satisfy the spatio-temporal constraints. At first, a directed state-time graph constructed along the reference path is used to search for a coarse-grained trajectory by mapping the keyframe as the goal. Then, using the information extracted from the coarse trajectory as initialization, adjoint-based optimization is applied to generate a finer trajectory with smooth motions based on our force-based simulation. We validate our method with extensive experiments.

CCS Concepts

• ***Computing methodologies*** → ***Procedural animation; Interactive simulation;***

1. Introduction

Traffic simulation has received increasing attention due to the development of computer games, film industry, urban planning, au-

† Corresponding author: jin@cad.zju.edu.cn

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

tonomous vehicle driving [LPZ*19], etc. A reliable traffic simulator that can generate high-fidelity virtual traffic data is thus valuable [CBL*20].

While realistic traffic flows can be simulated via a variety of simulators [KEBB12, APA*16, Fel94, DRC*17], editing or imposing specific space-time constraints on vehicles is difficult. Meanwhile, the capability of interactively editing vehicle trajectories is needed when traffic scenes need to be simulated with specific vehicles controlled/showing a pre-defined driving behavior. As a result, currently traffic simulation editing has to be based on labor-intensive manual tuning of simulation parameters and exhaustive trial-and-error runs of simulators. Some attempts have been made to address such a limitation, e.g. by allowing users to manually generate desired trajectories or rare traffic events observed less frequently in previous methods or datasets [HRW*22]. However, they do not address the spatio-temporal nature of the editing constraints, e.g. specifying a certain vehicle to arrive a certain position at a pre-defined moment. Recently, traffic reconstruction methods [VDBO07, SVDBLM10] provide a potential solution via optimization with respect to the space-time constraints. But they deteriorate the trajectory quality, e.g. discontinuous and implausible trajectories, and incur large computational costs which renders them unscalable.

To ensure the pluasibility of edited trajectories, TraED-ITS [HRW*22] integrates a data-driven traffic simulation module inspired by [RXX*19]. Data-driven methods can utilize pre-recorded traffic data to generate realistic behaviors. However, there is a fundamental challenge to combine data-driven methods with optimization-based methods which are widely used in traffic simulation. The core reason is the simulation process of optimization-based methods is intrinsically non-differentiable, making gradient-based learning infeasible. It is worth noting that the social force model, which is widely used in crowd animation, has also shown fantastic potential in traffic simulation recently. A unified force-based framework [CJH*19] and a simplified force-based framework [HCJ21] are successively proposed to simulate mixed traffic scenarios with different types of agents. Because agent's dynamics are explicitly expressed with derivable formulas, optimization based on force models is more practical. However, current force-based traffic simulation methods can only provide scenarios with simple straight lanes, and vehicle's motions are restricted strongly to the lane shapes.

To generate traffic trajectories based on the force-based models with given spatio-temporal constraints, keyframing is a practical and effective technique for controlling coarse results in physically-based simulation. The adjoint method, as one of the popular methods in optimal control for gradient computation, has been introduced in fluid simulation [MTPS04] and general particle dynamics [WMT06], to satisfy given keyframe constraints. However, the results of the adjoint optimization are highly dependent on the initialization. Bad initialization can slow down or even prevent gradient descent from achieving convergence.

None of the prior methods can fulfill all of our requirements perfectly due to the following challenges. Firstly, the current force-based traffic simulation frameworks constrain vehicles to drive along straight lanes and can hardly meet the demands of arbitrary settings or edits given by users. Secondly, there is no proper method for optimizing traffic trajectories with spatio-temporal keyframe control. State-time space search becomes extremely time-consuming if we need smooth traffic behaviours, while gradient-based optimization like the adjoint method may decelerate convergence or generate implausible trajectories due to poor initialization. The optimization process needs to be adapted to efficiently generate plausible traffic behaviours.

To address the above challenges, we provide a novel traffic trajectory editing method that allows users to control vehicle's motions with spatio-temporal keyframes. Vehicles are updated by force-based models using path coordinates to make them more maneuverable rather than being strongly restricted to straight lanes. A coarse-to-fine optimization process is presented to find optimal keyframe controls of the simulation appropriately. The main contributions of this work are as follows:

- A traffic trajectory editing approach, which allows users to specify keyframes to regulate vehicles and generate desired trajectories that can meet the spatio-temporal constraints while the traffic behaviours can remain plausible.
- A novel force-based framework that decouples vehicle's motions from static lanes and can generate microscopic traffic simulation in complex scenarios including diverse interactions.
- A coarse-to-fine optimization process is developed by combining the adjoint method with state-time space search to perform gradient descent effectively and stably to generate smooth trajectories with keyframe controls.

2. Related Work

2.1. Interactive Editing Technology

Traffic simulation software packages like SUMO [KEBB12], SimMobility [APA*16], Vissim [Fel94] and Carla [DRC*17] can generate traffic flows effectively. However, if users want to edit the results or generate some cases with specific behaviours, they have to tune parameters and run the simulation over and over based on the previous result until it meets the expectation.

The interactive editing concept was proposed to solve the similar problem in crowd animation at the very beginning. Cage-based deformation was introduced to edit large-scale crowd animation interactively [KSKL14]. Similar results can be achieved based on mesh deformation [ZZZY20]. Users can also control the simulation in real-time by drawing sketches as reference paths or obstacles [MM17]. In traffic simulation, a human-in-the-loop framework is proposed, which allows users to generate irregular and diverse traffic trajectories [HRW*22] by generating self-defined reference paths or modifying vehicle's attributes. However, this method only allows users to edit in space dimension, and the temporal constraints like specific arriving time are not supported explicitly. Our method allows users to assign spatio-temporal keyframes to regulate vehicle's motions and generate desired trajectories.

2.2. Traffic Simulation

In computer graphics, data-driven methods are used to generate realistic traffic flows based on pre-captured traffic data

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

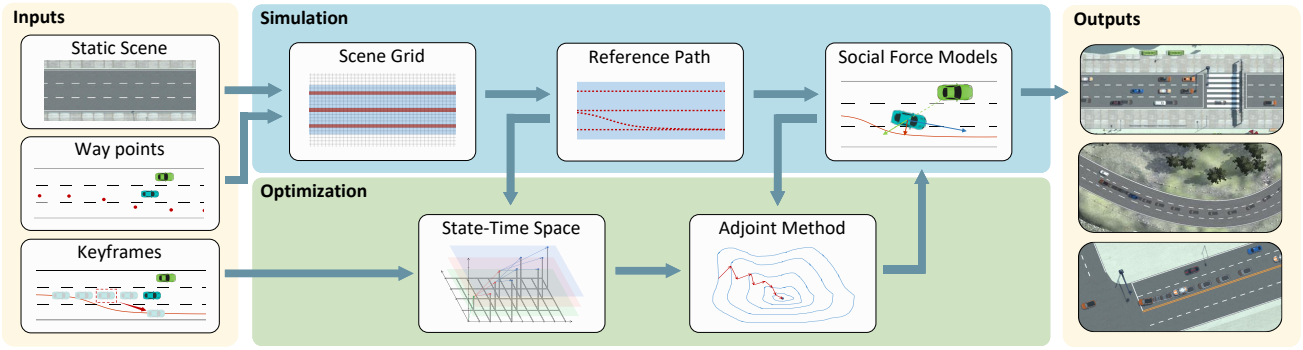

Figure 2: *Overview of our spatio-temporal keyframe control of traffic simulation using a coarse-to-fine optimization process. By discretizing static scenarios, we can generate reference paths for vehicles using the given way-points. We simulate the vehicles using our proposed social force models. With given spatio-temporal keyframes, a coarse-grained state-time search and fine-grained adjoint-based optimization are combined to efficiently generate smooth trajectories that can satisfy the keyframe constraints.*

from video, LiDAR, GPS, and other available sensors. Spatio-temporal data from in-road sensors is provided to reconstruct traffic flows [WSL13, LWL17]. Based on texture synthesis and cage-based deformation, appropriate traffic flows can be populated in any road network with given samples [CDR*17]. A data-driven optimization-based method is proposed to simulate heterogeneous multi-agent systems [RXX*19]. However, it is difficult to combine data-driven methods with numerical optimization if we want to add extra spatio-temporal constraints through the generated trajectories since it is difficult to compute the derivative of data-driven simulation processes.

On the other hand, the social force model, which is widely used in crowd animation [HM95, HFV00, AGR*16, CSC16], has recently shown great potential in traffic simulation due to its high computational effectiveness and flexibility. In order to consider all possible agents in a realistic urban environment, a unified force-based framework is proposed which can describe various interactions among different types of agents [CJH*19]. Based on an object-oriented concept, a simplified model is proposed to parameterize adjustable coefficients for different agents, make parameter tuning more intuitive and improve the scalability of the framework [HCJ21]. With real-world traffic datasets, the coefficients in the force model can also be calibrated automatically with adaptive genetic algorithm [CLH*21]. However, these methods only provide simulation scenarios with straight lanes, and vehicles' behaviours are strongly restricted and difficult to extend to complex scenarios. We develop a more viable force-based framework to simulate vehicles in different environments and generate diverse behaviours.

2.3. Keyframe Control Animation

Keyframe is a classical technique for giving fine-grained controls manually to coarse results in physically-based simulation such as fluid, smoke and cloth. As one of the popular methods of optimal control problems involving gradient computation, the adjoint method is introduced to control fluid simulation [MTPS04], general particle systems [WMT06] as well as elastic motion editing [LHDJ13]. However, the gradient-based methods greatly rely on the initialization, and are improper to be applied to optimize traffic simulation directly. Specifically, traffic reconstruction methods can respect keyframe constraints via searching for paths by mapping the keyframes as the goals in state-time space [VDBO07, SVD-BLM10], but their performance and generated results strongly depend on the space discretization timestep. Therefore, we combine the adjoint method with state-time space search and propose a coarse-to-fine optimization process that can generate plausible traffic simulation results effectively with given keyframes.

3. Method

The overview of our proposed method is demonstrated in Fig. 2. We first introduce the Frenet coordinates and our static scenarios representation (Section 3.1).Then we show the force-based traffic simulation including social forces calculation (Section 3.2.1) and reference path planning (Section 3.2.2). In order to provide keyframe controls with spatio-temporal constraints, we come up with a coarse-to-fine optimization process (Section 3.3). The state-time graph is constructed (Section 3.4.1) to search for coarse-grained trajectories as initialization (Section 3.4.2, 3.5.1), and the adjoint method is further applied to obtain smooth trajectories and plausible behaviours (Section 3.5.2).

3.1. Frenet Coordinates and Scenario Representation

We perform our framework in both the Cartesian coordinates and the Frenet coordinates. Frenet frame is a more intuitive way to describe the vehicle state in the lane. A Frenet coordinate $[s,d]$ consists of the longitudinal displacement along the lane and the lateral deviation relative to the lane center. We use $\boldsymbol{p} = [x,y]$, $\hat{\boldsymbol{p}} = [s,d]$, $\boldsymbol{v} = [v^x, v^y]$ and $\hat{\boldsymbol{v}} = [v^s, v^d]$ to represent positions and velocities in the Cartesian coordinates and the Frenet coordinates, respectively. Similarly, we use notations $\boldsymbol{f} = [f^x, f^y]$ and $\hat{\boldsymbol{f}} = [f^s, f^d]$ to represent forces or other attributes in these two coordinate systems.

Specifically in our work, *Lane* is the static areas where vehicles can drive, usually defined segment-by-segment with specific lane width in the configuration files. *Path* is a sequence of points linking

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

a starting point to a destination, and we interpolate them by cubic spline. Namely, the center line of each *Lane* with a specific driving direction is a *Path*. We use $\mathcal{L}$ and $\mathcal{P}$ to represent a *Lane* and a *Path*, respectively.

Given a certain scene configuration, the lanes in the scene are always determined. So we initialize the reference path set as:

$$\begin{aligned} \mathcal{P}^* &= \mathcal{P}^*_{topo} \cup \mathcal{P}^*_{user}, \\ \mathcal{P}^*_{topo} &= DFS\left(\mathcal{L}^*\right), \\ \mathcal{P}^*_{user} &= \varnothing, \end{aligned} \tag{1}$$

where $\mathcal{P}^*$ is the reference path set containing all the available reference paths in the scene, and $\mathcal{L}^*$ is the lane set containing all the lanes defined in the configuration file. $\mathcal{P}^*_{topo}$ and $\mathcal{P}^*_{user}$ represent the reference paths determined by lanes' topology and user settings, respectively. Each path in $\mathcal{P}^*_{topo}$ is a unique link from the origin of a lane without in-comings to the end of a lane without out-goings. *DFS* represents the depth-first search function.

In accordance with [HRW*22], we generate a grid map for the scene in 2D space with a given resolution. The lanes are embedded into the grid nodes by capsule-like shape approximation for every segment, and we label the grid cells of undrivable areas, drivable areas and lane centers with different values. The 2D grid map will be used to plan the reference paths with way-points from users and store vehicles' information to accelerate real-time neighbor search during the simulation.

3.2. Force-based Traffic Simulation

3.2.1. Force Calculation

We model three factors which heavily influence a vehicle's state: self-motivation, surrounding neighbors and the environment. The state of a vehicle at time t is denoted as $[\hat{\boldsymbol{v}}_t, \hat{\boldsymbol{p}}_t, \boldsymbol{v}_t, \boldsymbol{p}_t, \theta_t, \hat{\boldsymbol{v}}_{o,t}, \mathcal{P}_k]$. $\hat{\boldsymbol{v}}_t, \boldsymbol{v}_t \in \mathbb{R}^2$ represents its velocity in Frenet coordinates and Cartesian coordinates, respectively. $\hat{\boldsymbol{p}}_t, \boldsymbol{p}_t \in \mathbb{R}^2$ represent its position in Frenet coordinates and Cartesian coordinates, respectively. $\theta_t \in \mathbb{R}$ represents its orientation by Euler angles. $\hat{\boldsymbol{v}}_{o,t} \in \mathbb{R}^2$ represents the free-flow desired velocity. $\mathcal{P}_k \in \mathcal{P}^*$ represents the reference path k it follows. The vehicle's attributes in the Frenet coordinates and the Cartesian coordinates can be interconverted by the cubic spline function S_k of its reference path. The dynamics of a vehicle are formulated as:

$$\begin{aligned} \hat{\boldsymbol{f}}_t &= \hat{\boldsymbol{f}}_{o,t} + \hat{\boldsymbol{f}}_{k,t} + S_k\left(\sum_{j \in \mathcal{N}_t} \boldsymbol{f}_{j,t}\right), \\ \hat{\boldsymbol{v}}_{t+1} &= \hat{\boldsymbol{v}}_t + \frac{\hat{\boldsymbol{f}}_t}{m} \cdot \Delta t, \\ \hat{\boldsymbol{p}}_{t+1} &= \hat{\boldsymbol{p}}_t + \hat{\boldsymbol{v}}_{t+1} \cdot \Delta t, \\ [\boldsymbol{v}_{t+1}, \boldsymbol{p}_{t+1}, \theta_{t+1}] &= S_k\left(\hat{\boldsymbol{p}}_{t+1}\right), \end{aligned} \tag{2}$$

where $\hat{\boldsymbol{f}}_t$ represents the total force on the vehicle in Frenet coordinates, which is a combination of the self-motivated force $\hat{\boldsymbol{f}}_{o,t}$, the path keeping force $\hat{\boldsymbol{f}}_{k,t}$ from its reference path $\mathcal{P}_k$ and the collision avoidance forces $\boldsymbol{f}_{j,t}$ exerted by the neighbor vehicle j in its neighbors set $\mathcal{N}_t$. Specifically, the collision avoidance forces are computed in Cartesian coordinates at first. m represents the vehicle's mass. The velocity and the position in Frenet coordinates are updated by the forces. The velocity and the position in Cartesian coordinates as well as the orientation are obtained using the cubic spline function S_k. Δt is the timestep used in the simulation.

Self-motivated force: We assume that each vehicle has a desired velocity to travel at when it is not constrained by the presence of neighbor vehicles. Different from the previous methods [CJH*19, HCJ21, CLH*21], we formulate the self-motivated force as:

$$\hat{\boldsymbol{f}}_{o,t} = \omega_o m \left(\frac{2}{1 + e^{\hat{\boldsymbol{v}}_{o,t} - \hat{\boldsymbol{v}}_t}} - 1\right) \hat{\boldsymbol{a}}, \tag{3}$$

where ω_o is a corresponding weight, and $\hat{\boldsymbol{a}}$ is the maximum acceleration of the individual in Frenet coordinates. Such formulation can make vehicles gradually change and keep their velocity to the desired $\hat{\boldsymbol{v}}_{o,t}$, while the derivative of the expression exists at each point in its domain.

Path keeping force: Typically drivers tend to drive along the lane center due to safety and traffic rules in real world traffic. As mentioned, we consider the center line of a lane with a driving direction as a path, so the lane keeping behaviors can also be regarded as path keeping. We define the path keeping force as an attractive force from the path:

$$\hat{\boldsymbol{f}}_k = \begin{cases} \omega_k |d_t| \boldsymbol{u}_k, & |d_t| \geq \frac{1}{2}(w_l - w_v) \\ \boldsymbol{0}, & otherwise \end{cases}, \tag{4}$$

where ω_k is a corresponding weight, $d_t \in \hat{\boldsymbol{p}}_t$ is the current lateral displacement of the vehicle related to the path, and $\boldsymbol{u}_k$ is the unit vector pointing from the vehicle to the path. w_l is the lane width, and w_v is the vehicle's width. As shown, path keeping force is active only if a vehicle's deviation from its reference path exceeds the threshold in order to prevent the vehicle oscillating around the path.

Collision avoidance force: A vehicle should avoid colliding with others who are too close to it. We formulate such behaviour as a point-to-point repulsive force that effects between the vehicle and its surrounding neighbors. The collision avoidance force between a vehicle and its neighbor j is defined as:

$$\begin{aligned} \boldsymbol{f}_j &= \omega_c \frac{a \cdot b}{\left(1 + c||\boldsymbol{p}_{j,t} - \boldsymbol{p}_t||\right)^2} \boldsymbol{u}_c, \\ a &= \begin{cases} cos\phi, & \phi \leq \frac{\pi}{4} \\ 0, & otherwise \end{cases}, \\ b &= s_0 + ||\boldsymbol{v}_t|| T_0 + \frac{||\boldsymbol{v}_t|| \cdot ||\boldsymbol{v}_{j,t} - \boldsymbol{v}_t||}{2||\hat{\boldsymbol{a}}||}, \\ c &= \frac{1}{s_0}, \end{aligned} \tag{5}$$

where ω_c is a corresponding weight, a is a visual factor, b and c are parameterized coefficients, and $\boldsymbol{u}_c$ is the unit vector pointing from j to the vehicle. We formulate the coefficients inspired by the intelligent driver model (IDM) [THH00]. ϕ is the angle between the vehicle's moving direction and the direction of the vehicle pointing to j. $\boldsymbol{v}_{j,t}$ and $\boldsymbol{p}_{j,t}$ is the velocity and the position of j in Cartesian coordinates. s_0 and T_0 are the safety space headway between two vehicles and the reacting time for the vehicle to brake, which are both constant for a certain individual. The neighbor set $\mathcal{N}_t$ can be

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

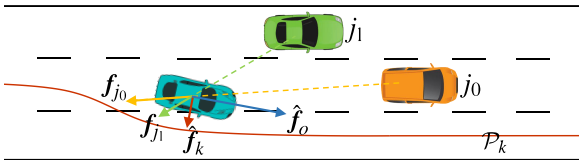

Figure 3: *An example of the forces on a vehicle. The vehicle follows the reference path $\mathcal{P}_k$ and has two neighbors, where j_0 is in the same lane and j_1 is in the adjacent lane.*

updated at each frame effectively by the 2D grid map, where the search range is 100 × 100 grid nodes in our implementation.

Based on the force models, our traffic simulation can generate smooth traffic behaviours such as acceleration/deceleration, lane keeping and lane changing. Since we update the vehicles along their reference paths, our framework can simulate the traffic in complex scenarios. We give an example of the forces on a driving vehicle in Fig. 3.

3.2.2. Reference Path Planning

In addition to the paths extracted from lane centers, new reference paths can be created by clicking a sequence of key points in the scene. The given key points are mapped to the specific nodes on the constructed 2D grid map, and then the whole path is planned segment-by-segment, where the first node of each segment is considered as the start and the second one is the terminal. We use the A* algorithm to plan the path and define the heuristic function as:

$$h(\boldsymbol{n}) = ||\boldsymbol{n} - \boldsymbol{n}_{goal}|| + \mu_a \cdot e^{(\mu_b \cdot sign)}, \tag{6}$$

where $||\boldsymbol{n} - \boldsymbol{n}_{goal}||$ is the Euler distance between the current node and the terminal, and μ_a and μ_b are adjustable coefficients. $sign \in [0,1,2]$ is the sign filled in different types of nodes we metioned in Section 3.1, where 0 represents unreachable area, 1 represents drivable area and 2 represents lane center. The second term on the right-hand side of the equation can make the planning tend to search along with lane centers.

We post-process the planning results by down-sampling, Gaussian smoothing and interpolating with cubic spline. Finally the user-defined path $\mathcal{P}$ becomes available for vehicles to follow, denoted as $\mathcal{P}^*_{user} = \mathcal{P}^*_{user} \cup [\mathcal{P}]$. Specifically, once a vehicle's reference path is changed, its state values represented in Frenet coordinates need to be updated according to the new cubic spline function.

3.3. Spatio-temporal Keyframe Control

In order to provide further keyframe controls with both spatial and temporal constraints, the adjoint method is applied since it is proven to be effective in optimal control. We can find a set of optimal controlling desired speeds and corresponding controlling forces, and make vehicle's behaviours satisfy the keyframe constraints based on our force-based simulation.

However, gradient-based optimization greatly depends on the initial values of the parameters to be optimized. Bad initialization may decelerate or even prevent gradient descent from achieving convergence. Therefore, we perform another optimization in a discrete state-time space to find a coarse-grained trajectory at first, and then treat it as initialization to further optimize and improve the behaviours. The pseudo-code of our coarse-to-fine optimization is described in Algorithm 1, and we will introduce the details in the following sections.

Algorithm 1 Coarse-to-fine optimization

Input:
State-time graph for a vehicle along its reference path;
K setting keyframes $Q = [\tilde{\boldsymbol{q}}_0, \tilde{\boldsymbol{q}}_1, ..., \tilde{\boldsymbol{q}}_K]$;
The maximum iteration number N for the adjoint method;
Learning rate α, exponential decay rates β_0, β_1;
Output:
Trajectory $\mathcal{T}$ under keyframes control;

Initialize controlling desired speeds $V \Leftarrow \varnothing$, controlling forces $F \Leftarrow \varnothing$;
for $i = 0$ to $K-1$ **do**
  Start state-time node $[s_{start}, v^s_{start}, t_{start}] \Leftarrow \tilde{\boldsymbol{q}}_i$;
  Goal state-time node $[s_{goal}, v^s_{goal}, t_{goal}] \Leftarrow \tilde{\boldsymbol{q}}_{i+1}$;
  Find a coarse trajectory in state-time graph with A* algorithm, $\mathcal{T}_c \Leftarrow \mathcal{A}^*([s_{start}, v^s_{start}, t_{start}], [s_{goal}, v^s_{goal}, t_{goal}])$;
  Append controlling desired speeds, $V \Leftarrow V \cup [v^s_{o,0}, v^s_{o,1}, ...]$ extracted from current $\mathcal{T}_c$;
end for
Pad controlling desired speeds V;
for $i = 0$ to $N-1$ **do**
  Compute corresponding forces with controlling desired speeds, $F \Leftarrow V$;
  Simulate the current whole trajectory $\mathcal{T}_o$ with F using our force-based traffic simulation algorithm;
  if loss decreases **then**
    Update $\mathcal{T} \Leftarrow \mathcal{T}_o$;
  end if
  Compute gradient of desired speeds using the adjoint method, $\nabla V \Leftarrow \mathcal{A}djoint(\mathcal{T}_o, V)$;
  Gradient descent, $V \Leftarrow Adam(\nabla V, \alpha, \beta_0, \beta_1)$;
end for
return Fine trajectory $\mathcal{T}$;

3.4. Coarse Search in State-Time Space

3.4.1. State-Time Graph Construction

We define the state space of a vehicle as a subset of its entire state representation shown in Section 3.2.1 for simplicity in the following. The state space along a vehicle's reference path in the longitudinal direction is denoted as $[s, v^s]$, where $s \in \hat{\boldsymbol{p}}$ and $v^s \in \hat{\boldsymbol{v}}$ are the longitudinal displacement and the longitudinal speed, respectively.

The state space can be discretized into grid nodes by giving another timestep $\Delta\tilde{t}$, which is much larger than the one we use in traffic simulation. Assume that the acceleration of a vehicle can only be chosen from a discrete set $[-a^s, 0, a^s]$ where $a^s \in \hat{\boldsymbol{a}}$ is the maximum longitudinal acceleration. According to the state dynamics shown in Eq. 2, the intervals along the v^s-axis and the s-axis are

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

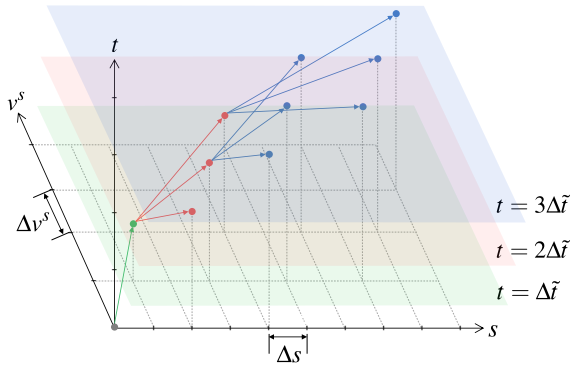

Figure 4: *A part of discretized state-time space and directed state-time graph. The reachable nodes are marked by the dots with different colors laying on the corresponding s-v^s planes, representing time at $\Delta\tilde{t}$ (green), $2\Delta\tilde{t}$ (red) and $3\Delta\tilde{t}$ (blue). The transitions from parent nodes denoted by arrows are also colorized.*

$\Delta v^s = a^s\Delta\tilde{t}$ and $\Delta s = \frac{1}{2}a^s(\Delta\tilde{t})^2$. Therefore, starting from a given state node $[s, v^s]$, there should be three reachable state nodes after $\Delta\tilde{t}$: $[s+(2\frac{v^s}{\Delta v^s}+1)\Delta s, v^s+\Delta v^s]$ for acceleration, $[s+2\frac{v^s}{\Delta v^s}\Delta s, v^s]$ for speed maintenance and $[s+(2\frac{v^s}{\Delta v^s}-1)\Delta s, v^s-\Delta v^s]$ for deceleration. As a result, the behaviours along the reference path can be represented as a directed graph with finite combinations of reachable state nodes.

The state-time space of a vehicle is the state space augmented by the time dimension. Similarly, we can also construct a directed graph in the state-time space that contains transitions from given state-time nodes $\tilde{\boldsymbol{q}} = [s, v^s, t]$ to their corresponding three reachable state-time nodes. Therefore, the given keyframes can be mapped to specific state-time nodes, and we need to find a trajectory from the start node $[s_{start}, v^s_{start}, t_{start}]$ to the goal $[s_{goal}, v^s_{goal}, t_{goal}]$ through the spanning graph. A part of discretized state-time space and state-time graph are demonstrated in Fig. 4.

The state-time space and the corresponding directed state-time graph for any vehicle are identical with the same discretization process. In practice, we only need to explicitly construct one directed state-time graph and reuse it for different reference paths.

3.4.2. Coarse Trajectory Search

Finding a trajectory through the state-time graph with given a start and goal can be seen as finding an optimal path in the special 3D solution space. We also use the A* algorithm to search for it. The heuristic function for a certain node is defined as:

$$\begin{aligned}
\tilde{h}(\tilde{\boldsymbol{q}}) &= \omega_d\tilde{h}_d(\tilde{\boldsymbol{q}}) + \omega_a\tilde{h}_a(\tilde{\boldsymbol{q}}),\\
\tilde{h}_d(\tilde{\boldsymbol{q}}) &= \sqrt{|s-s_{goal}|^2 + |v^s - v^s_{goal}|^2 + |t-t_{goal}|^2},\\
\tilde{h}_a(\tilde{\boldsymbol{q}}) &= \frac{|v^s - v^s_{parent}|}{\Delta\tilde{t}},
\end{aligned} \tag{7}$$

where $\tilde{h}_d$ is the distance between the state-time node and the goal, and $\tilde{h}_a$ is the variation of speed compared with its parent state, making the vehicle accelerate or decelerate as infrequently as possible. ω_d and ω_a are corresponding weights.

Moreover, other moving vehicles can be pre-converted to static obstacles in the state-time space [SVDBLM10]. During the A* search, the nodes occupied by other vehicles are marked as unreachable from their parent nodes. The nodes that exceed allowed speed, path length or maximum time duration are also prohibited. However, such constraints may raise failure that we can not find a trajectory that matches the keyframes. To ensure that the A* algorithm always produces an output, we will return the trajectory that can reach the node closest to the goal and has the minimal heuristic value when the search can not reach the true goal.

Obviously, this generated trajectory is implausible because of the large timestep used to construct the state-time graph as well as the discrete acceleration options. Though using a small timestep or adding more alternative acceleration values can generate better results, they will also make space discretization and state-time search become extremely time-consuming because of the exponential rise of the number of state-time nodes. Therefore, we will use the adjoint method to refine the behaviours along the coarse trajectory depending on our force-based simulation.

3.5. Trajectory Refinement with Adjoint Method

3.5.1. Initialization Using Coarse Trajectory

We use the notation $\mathcal{T}$ to represent a trajectory in the following. After searching a coarse trajectory $\mathcal{T}_c$ in state-time space, we use the speeds extracted from $\mathcal{T}_c$ as the initialized controlling desired speeds $V = [v^s_{o,0}, v^s_{o,1}, ...]$, and further optimize them based on our force models to make the simulation satisfy the given keyframes.

As we mentioned in the last section, different timesteps in traffic simulation and state-time space discretization are used. It will cause different numbers of frames when a certain trajectory is registered by simulation and state-time space. We pad the sequence of controlling desired speeds V extracted from coarse trajectory by linear interpolation to align the number of frames with the trajectory generated by simulation. Then we can use the padded V to calculate the corresponding controlling forces F and trajectory $\mathcal{T}_o$. $\mathcal{T}_c$ and $\mathcal{T}_o$ can be seen as the different representations for a certain trajectory in state-time space and traffic simulation, respectively. Obviously, we have a relationship that $\#\mathcal{T}_o \cdot \Delta t = \#\mathcal{T}_c \cdot \Delta\tilde{t}$, where $\#\mathcal{T}_o$ and $\#\mathcal{T}_c$ are the numbers of frames, and both sides are equal to the total driving time duration of this trajectory.

3.5.2. Adjoint-based Optimization

To measure the difference between the trajectory $\mathcal{T}_o$ with T frames generated by the controlling desired speeds V and a set of keyframes Q specified by users, we define the objective function which we want to minimize as:

$$\begin{aligned}
&\Phi(\mathcal{T}_o, V) = \frac{1}{2}\sum_{t=0}^{T}\left(\omega_t||\mathcal{T}_{o,t} - Q_t||^2 + \omega_v||v^s_{o,t}||\right),\\
&s.t. \quad \mathcal{T}_{o,t+1} = G\left(\mathcal{T}_{o,t}, v^s_{o,t}\right), \quad t \in [0, 1, ...T-1],
\end{aligned} \tag{8}$$

where ω_t is a weight to emphasize the influence of the state at certain keyframes, $\mathcal{T}_{o,t} = [s_t, v^s_t]$ is the state at time t along the trajectory, and Q_t is the keyframe at time t if there exists. In fact, $\mathcal{T}_{o,t}$ should be the state-time node $[s, v^s, t]$ as our definition, but the time

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

Parameter	Value	Unit	Description
Δt	0.01	s	the timestep used in the force-based traffic simulation
$\hat{\boldsymbol{a}}$	[5.0, 1.0]	m/s^2	the maximum acceleration of the vehicles in Frenet coordinates
s_0	4.0 ± 1.0	m	the jam space headway between two vehicles for safety
T_0	1.0 ± 0.5	s	the reacting time for the vehicles to brake
w_v, w_l	1.8, 3.5	m	the width of lanes and the vehicles, respectively
ω_o, ω_k, ω_c	1.0, 0.5, 3.0	–	the weights for the force calculations in Eq. 3, Eq. 4 and Eq. 5, respectively
μ_a, μ_b	20.0, -1.5	–	the coefficients for the heuristic-based path planning in Eq. 6
$\Delta\tilde{t}$	0.5	s	the timestep used in state-time space discretization
ω_d, ω_a	1.0, 2.0	–	the weights for the heuristic-based state-time search in Eq. 7
ω_t, ω_v	1.0, 0.1	–	the weights for the objective function in the adjoint method in Eq. 8
α, β_0, β_1	0.01, 0.9, 0.999	–	the parameters of Adam optimizer used for gradient descent in Algorithm 1

Table 1: *The values of the important parameters used in our experiments.*

dimension is left out here because we strictly align every timestamp when calculating the objective function. A regularization term weighted by ω_v is also added to prevent overfitting.

As shown in Eq. 8, the optimization should satisfy a series of time stepping constraints advanced via function G, an abbreviation of the state dynamics functions we have already shown in Eq. 2. According to the adjoint method, we introduce a set of Lagrange multipliers and transform the optimization into

$$\begin{aligned} \nabla V &= \frac{d\Phi}{dV} = \sum_{t=0}^{T} \lambda_t \cdot \frac{\partial G}{\partial v_{o,t}^s} + \frac{\partial \Phi}{\partial V}, \\ \lambda_t &= \begin{cases} \frac{\partial \Phi}{\partial \mathcal{T}_{o,t}}, & t = T \\ \lambda_{t+1} \cdot \frac{\partial G}{\partial \mathcal{T}_{o,t}} + \frac{\partial \Phi}{\partial \mathcal{T}_{o,t}}, & t < T \end{cases}, \end{aligned} \tag{9}$$

where λ_t is the Lagrange multiplier for time t, which is also called an adjoint state in the adjoint method. These Lagrange multipliers are calculated by iterating backward in time at first, and then we can obtain the gradient of the controlling desired speeds V by substitution. Finally, the controlling desired speeds are updated by gradient descent algorithm to get a new set of controlling forces which tends to decrease the difference between the simulation trajectory and the given keyframes. We use Adam optimizer for gradient descent in our implementation.

For clarity, we further demonstrate how to solve the terms $\partial G/\partial v_{o,t}^s$ and $\partial G/\partial \mathcal{T}_{o,t}$ in the above equations. For the speed component $v^s \in [s, v^s]$ along $\mathcal{T}_o$, according to Eq. 2, the transition function G can be written as:

$$v_{t+1}^s = G(\mathcal{T}_{o,t}, v_{o,t}^s) = v_t^s + \frac{f_t^s}{m} \cdot \Delta t, \tag{10}$$

where f_t^s is the longitudinal component of the total force at time t. So we can obtain that

$$\begin{aligned} \frac{\partial G}{\partial v_{o,t}^s} &= \frac{\partial f_t^s}{\partial v_{o,t}^s} \frac{\Delta t}{m}, \\ \frac{\partial G}{\partial \mathcal{T}_{o,t}} &= \left[1 + \frac{\partial f_t^s}{\partial v_t^s} \frac{\Delta t}{m}, \frac{\partial f_t^s}{\partial s_t} \frac{\Delta t}{m} \right]. \end{aligned} \tag{11}$$

According to the force models demonstrated in Section 3.2.1, we can easily calculate these expressions. In our implementation, we only compute the derivative of self-motivated force. Since the collision avoidance force calculation contains a non-differentiable piecewise function, and the path keeping force has no contribution to the vehicle's longitudinal motions. In a similar way, the two terms for the position component $s \in [s, v^s]$ are

$$\begin{aligned} \frac{\partial G}{\partial v_{o,t}^s} &= \frac{\partial f_t^s}{\partial v_{o,t}^s} \frac{(\Delta t)^2}{m}, \\ \frac{\partial G}{\partial \mathcal{T}_{o,t}} &= \left[\Delta t \left(1 + \frac{\partial f_t^s}{\partial v_t^s} \frac{\Delta t}{m} \right), 1 + \frac{\partial f_t^s}{\partial s_t} \frac{(\Delta t)^2}{m} \right]. \end{aligned} \tag{12}$$

After repeating optimization for certain iterations, the final refined trajectory $\mathcal{T}$ can meet the given spatio-temporal keyframe constraints and also gets smoother.

4. Experimental Results

4.1. Experimental Setup

The following experiments were implemented on a computer with a 3.60GHz Intel(R) Xeon(R) W-2123 CPU with 8-core processors and 32GB memory. Our source code was implemented in C++, compiled as a x64 dynamic link library and imported into Unity3D for visualization. The values of some important parameters we used in our experiments are shown in Table 1, which were pre-defined in configure files and loaded by the program.

There are three scenarios used in our cases which were manually created by SUMO NetEdit and exported as XML files. The first scenario contains a curvy road with three lanes in the same direction. The second scenario contains a straight road with three lanes in the same direction as well as a crosswalk. The third scenario contains an intersection with a four-lane dual carriageway. The discretization resolution of the 2D grid map is 0.5m × 0.5m for each.

4.2. Keyframe Controlled Trajectories

We designed some cases to show the results of keyframe controlled simulation with our method. In the following figures, we use a yellow vehicle to represent the position constraint and a red box to

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

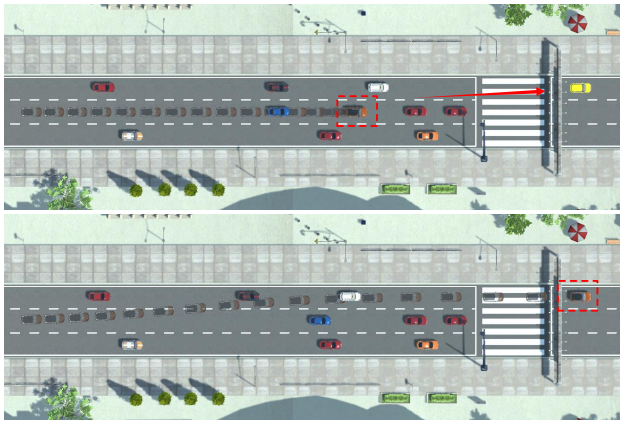

Figure 5: *The original trajectories (top) and the keyframe controlled trajectories (bottom) of running the red light in the crosswalk scenario.*

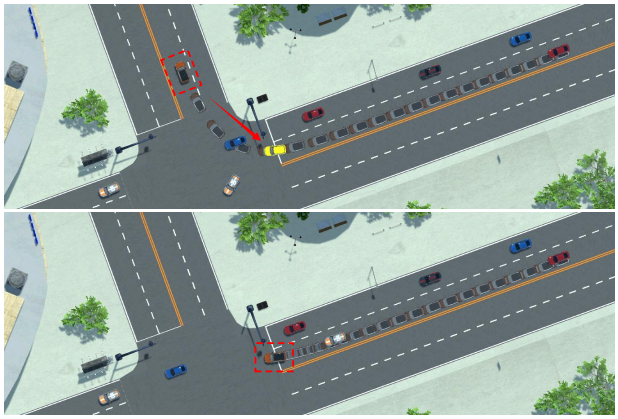

Figure 6: *The original trajectories (top) and the keyframe controlled trajectories (bottom) of yielding to the oncoming vehicles in the intersection scenario.*

frame the time constraint of the keyframe. That is to say, we want the vehicle to reach the position marked by the yellow vehicle at the moment framed by the red box. The edited vehicles are shown with its historical states, while the other vehicles are only shown with their final states.

The first case was generated in the scenario with a curvy road, which we have already shown in Fig. 1. The specific vehicle drove along the rightmost lane originally. We set a keyframe in the center lane and assigned the vehicle to reach the position at the framed time. As a result, a new reference path through the position was planned. Then, the vehicle changed the lane and decelerated to arrive at it on time.

The second case was generated in the crosswalk scenario. The specific vehicle braked in the center lane and waited for the red light originally. We set a keyframe to assign the vehicle to overtake and get through the crosswalk instead of waiting. As a result, a new reference path was planned to lead the vehicle to the lane which had not been blocked yet. The vehicle accelerated to run the red light (see Fig. 5).

Discretization Timestep $\Delta\tilde{t}$	State-Time Search Time
0.5	0.173
0.25	36.924
0.1	–

Table 2: *The state-time search time (s) over different discretization timesteps (s). Due to memory limitation, it is hardly to obtain the state-time search time when discretization timestep is 0.1s.*

Simulation Timestep Δt	Adjoint-based Optimization Time
0.5	0.002
0.1	0.004
0.05	0.011
0.01	0.185
0.005	0.699

Table 3: *The adjoint-based optimization time (s) over different traffic simulation timesteps (s) for a certain trajectory.*

The third case was generated in the interaction scenario. The specific vehicle turned right in the left lane without yielding to the oncoming vehicles driving forward in the right lane originally. We set a keyframe to assign it to wait before turning. In this case, we only needed to specify the position and the corresponding arrival time since the reference path for the vehicle was not changed. As a result, the vehicle decelerated and yielded at the intersection to let the vehicles in the right lane go first (see Fig. 6).

4.3. Performance and Comparison

To evaluate the performance of our coarse-to-fine optimization process, we performed a series of experiments with different timesteps or initialization for the adjoint method. The following experiments are based on a keyframe which constrains a vehicle to travel 100 meters in 10 seconds along its reference path, setting the number of the maximum iterations of the adjoint method to 100.

We searched for the coarse trajectory in the state-time space with different discretization timesteps $\Delta\tilde{t}$=0.5s, 0.25s and 0.1s. The corresponding search times are shown in Table 2. $\Delta\tilde{t}$=0.5s was used to reconstruct the traffic flows In [SVDBLM10]. When $\Delta\tilde{t}$ gets smaller, the generated trajectories with state-time search may become smoother, but search time and required memory will also increase rapidly. According to Table 2, the search time is 36.924s when $\Delta\tilde{t}$=0.25s. Due to the memory limitation, the search time can hardly be obtained when $\Delta\tilde{t}$=0.1s. Therefore, as mentioned earlier, it is impractical to obtain plausible keyframe controlled results by using a smaller discretization timestep in state-time space.

We optimized the trajectory using the adjoint method with different simulation timesteps Δt. The corresponding optimization times are shown in Table 3. The optimization time mainly depends on the number of frames of the trajectory, so it becomes more time-consuming when Δt gets smaller. In our implementation, we chose

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

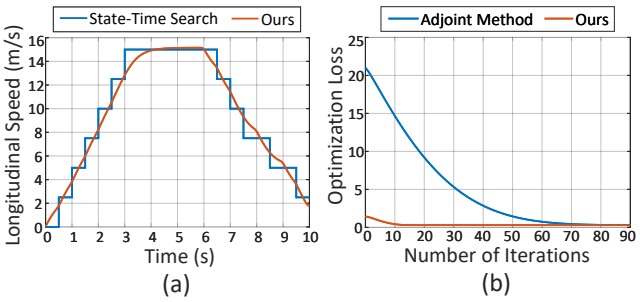

Figure 7: *(a) The longitudinal speeds of the vehicle generated by state-time search and our method with the same keyframe constraints. (b) The optimization loss over different numbers of iterations of the adjoint method initialized with average speed and our method with the same keyframe constraints.*

$\Delta\tilde{t}$=0.5s and Δt=0.01s in order to strike a balance between the total optimization time and the plausibility of generated trajectories. Thus the total optimization time with the given keyframe using our coarse-to-fine optimization is about 0.358s for 100 iterations.

We compare the results generated by state-time search only and our method. The longitudinal speeds along the generated trajectories are shown in Fig. 7 (a). The speed of the state-time search result changes suddenly since the vehicle can only choose the determined acceleration values in a discrete set at each timestep, which will cause implausibilities. In comparison, the result generated by our coarse-to-fine optimization is much smoother.

We further compare the loss during the optimization of the adjoint method and our method, and the loss values over different numbers of iterations are shown in Fig. 7 (b). For the adjoint method, we initialize the controlling desired speeds as the average speed that the vehicle needs to travel from the start to the key-frame position within the time. The loss achieves convergence at approximately the $70th$ iteration. In comparison, with our coarse-to-fine process, we initialize the controlling desired speeds better by utilizing the information of the coarse trajectory search in the state-time space. The loss at the beginning is thus much smaller than the optimization initialized with average speed and can achieve fast convergence at approximately the $15th$ iteration. Therefore, we can also give an early stop if the loss no longer decreases to further reduce the optimization time in our method.

Finally, the key aspects of our method in comparison with previous works are summarized in Table 4. Our method includes all the positive aspects of the previous works such as generating smooth traffic motions, allowing to simulate in complex scenarios and edit vehicles during the simulation. Furthermore, since our traffic simulation is based on the force models, we can provide spatio-temporal key frames to regulate vehicles and generate desired trajectories in a more intuitive way.

5. Discussion

5.1. Conclusion and Limitation

We have presented a novel traffic trajectory editing method that allows users to specify spatio-temporal keyframes to control vehicles' behaviours. We propose a force-based traffic simulation framework containing self-motivated force, path keeping force and collision avoidance force. It mainly updates vehicles based on the Frenet coordinates along vehicles' reference paths which can be defined manually. To provide keyframe controls, we propose a coarse-to-fine optimization process. First, we discretize the state-time space along the path, construct a state-time graph and plan a coarse trajectory from the start to the keyframe node. Then, we utilize the coarse trajectory to initialize the adjoint method and efficiently find a set of optimal controlling desired speeds to generate a finer trajectory based on our force-based simulation.

Method	(a)	(b)	(c)	(d)	(e)
[SVDBLM10]	State-Time Search	✗	✓	✗	✓
[CLH*21]	Force-based	✓	✗	✗	✗
[HRW*22]	Data-driven	✓	✓	✓	✗
Ours	Force-based	✓	✓	✓	✓

Table 4: *Comparison with previous methods. Several criteria are presented: (a) Traffic simulation model, (b) Smooth traffic motions, (c) Allowing to simulate in complex scenarios, (d) Allowing to interactively edit vehicles during the simulation, (e) Allowing to constrain vehicles with spatio-temporal keyframes.*

Though the proposed method is promising, it still has some limitations. Firstly, the keyframe constraints may become inoperative if the environment is congested. As we stated in Section 3.4.2, the state-time search regards the neighbors of an individual as static obstacles, which means its neighbors will not be optimized at the same time if we use keyframes to constrain it. If the vehicle is blocked when it tries to meet the keyframes during the state-time search, the result will be replaced by a trajectory that can reach the node closest to the actual goal. Though the problem can be solved by iteratively editing the surrounding vehicles who block the individual until it can meet the keyframes, we believe that a optimization process taking all the possible vehicles into consideration at once will be a worthwhile endeavor. Secondly, the simulation results may become irregular due to users' arbitrary edits. For example, a vehicle may decelerate for safety and comfort when it follows a path with sharp curves rather than maintaining a high speed in the real traffic. This problem can also be solved by interactively setting keyframes or editing vehicle's desired speed by users to make the behaviours more perceptually realistic.

5.2. Failure Cases and Refinements

To demonstrate our framework's capability, even for some failure cases caused by the aforementioned limitations, our method is still capable of refining them by manually providing more keyframes. For more information, please see our supplementary video.

The first scenario is that the vehicle misses the keyframe due to traffic congestion (see Fig. 8 (a)). We offer two solutions to the problem. The first solution is to give the vehicle another keyframe that causes it to overtake the leader (see Fig. 8 (b)). The second solution is to assign keyframes to its leaders, forcing them to yield (see Fig. 8 (c)). As a result, both refinements eventually succeed in getting the vehicle to meet the keyframe.

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

The second scenario is that the vehicle behaves abnormally when performing a high-speed U-turn (see Fig. 9 (a)). This is due to the fact that in real-world traffic, drivers tend to slow down for safety and comfort when following a more curved path. We also provide two methods for improving the behavior's perceptual realism. The first solution is to assign a keyframe that causes the vehicle to simply decelerate when passing through the curve (see Fig. 9 (b)). The second solution is to assign two keyframes, one to wait for oncoming vehicles and one to complete the U-turn, to further prevent aggressive driving and maintain polite behavior (see Fig. 9 (c)).

Acknowledgment

Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant No. 62036010) and the Key Research and Development Program of Zhejiang Province (Grant No. 2020C03096).

References

[AGR*16] ALAHI A., GOEL K., RAMANATHAN V., ROBICQUET A., FEI-FEI L., SAVARESE S.: Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 961–971. 3

[APA*16] ADNAN M., PEREIRA F. C., AZEVEDO C. M. L., BASAK K., LOVRIC M., RAVEAU S., ZHU Y., FERREIRA J., ZEGRAS C., BEN-AKIVA M.: Simmobility: A multi-scale integrated agent-based simulation platform. In *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record* (2016). 2

[CBL*20] CHAO Q., BI H., LI W., MAO T., WANG Z., LIN M. C., DENG Z.: A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 287–308. 2

[CDR*17] CHAO Q., DENG Z., REN J., YE Q., JIN X.: Realistic data-driven traffic flow animation using texture synthesis. *IEEE Transactions on Visualization and Computer Graphics 24*, 2 (2017), 1167–1178. 3

[CJH*19] CHAO Q., JIN X., HUANG H.-W., FOONG S., YU L.-F., YEUNG S.-K.: Force-based heterogeneous traffic simulation for autonomous vehicle testing. In *2019 International Conference on Robotics and Automation (ICRA)* (2019), IEEE, pp. 8298–8304. 2, 3, 4

[CLH*21] CHAO Q., LIU P., HAN Y., LIN Y., LI C., MIAO Q., JIN X.: A calibrated force-based model for mixed traffic simulation. *IEEE Transactions on Visualization and Computer Graphics* (2021). 3, 4, 9

[CSC16] COSGUN A., SISBOT E. A., CHRISTENSEN H. I.: Anticipatory robot path planning in human environments. In *2016 25th IEEE international symposium on robot and human interactive communication (RO-MAN)* (2016), IEEE, pp. 562–569. 3

[DRC*17] DOSOVITSKIY A., ROS G., CODEVILLA F., LOPEZ A., KOLTUN V.: Carla: An open urban driving simulator. In *Conference on robot learning* (2017), PMLR, pp. 1–16. 2

[Fel94] FELLENDORF M.: Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority. In *64th Institute of Transportation Engineers Annual Meeting* (1994), vol. 32, Springer, pp. 1–9. 2

[HCJ21] HAN Y., CHAO Q., JIN X.: A simplified force model for mixed traffic simulation. *Computer Animation and Virtual Worlds 32*, 1 (2021), e1974. 2, 3, 4

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407*, 6803 (2000), 487–490. 3

[HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical Review E 51*, 5 (1995), 4282. 3

[HRW*22] HAN Y., REN J., WANG S., SUN W., YANG R., JIN X.: Traedits: Diversity and irregularity-aware traffic trajectory editing. *IEEE Robotics and Automation Letters* (2022). 2, 4, 9

[KEBB12] KRAJZEWICZ D., ERDMANN J., BEHRISCH M., BIEKER L.: Recent development and applications of sumo-simulation of urban mobility. *International Journal on Advances in Systems and Measurements 5*, 3&4 (2012). 2

[KSKL14] KIM J., SEOL Y., KWON T., LEE J.: Interactive manipulation of large-scale crowd animation. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–10. 2

[LHDJ13] LI S., HUANG J., DESBRUN M., JIN X.: Interactive elastic motion editing through space–time position constraints. *Computer Animation and Virtual Worlds 24*, 3-4 (2013), 409–417. 3

[LPZ*19] LI W., PAN C., ZHANG R., REN J., MA Y., FANG J., YAN F., GENG Q., HUANG X., GONG H., ET AL.: Aads: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics* (2019). 2

[LWL17] LI W., WOLINSKI D., LIN M. C.: City-scale traffic animation using statistical learning and metamodel-based optimization. *ACM Transactions on Graphics (TOG) 36*, 6 (2017), 1–12. 3

[MM17] MONTANA L. R., MADDOCK S.: Sketching for real-time control of crowd simulations. In *Proceedings of the Conference on Computer Graphics & Visual Computing* (2017), pp. 81–88. 2

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG) 23*, 3 (2004), 449–456. 2, 3

[RXX*19] REN J., XIANG W., XIAO Y., YANG R., MANOCHA D., JIN X.: Heter-sim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization. *IEEE Transactions on Visualization and Computer Graphics 27*, 3 (2019), 1953–1966. 2, 3

[SVDBLM10] SEWALL J., VAN DEN BERG J., LIN M., MANOCHA D.: Virtualized traffic: Reconstructing traffic flows from discrete spatiotemporal data. *IEEE Transactions on Visualization and Computer Graphics 17*, 1 (2010), 26–37. 2, 3, 6, 8, 9

[THH00] TREIBER M., HENNECKE A., HELBING D.: Congested traffic states in empirical observations and microscopic simulations. *Physical review E 62*, 2 (2000), 1805. 4

[VDBO07] VAN DEN BERG J., OVERMARS M.: Kinodynamic motion planning on roadmaps in dynamic environments. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2007), IEEE, pp. 4253–4258. 2, 3

[WMT06] WOJTAN C., MUCHA P. J., TURK G.: Keyframe control of complex particle systems using the adjoint method. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2006), pp. 15–23. 2, 3

[WSL13] WILKIE D., SEWALL J., LIN M.: Flow reconstruction for data-driven traffic animation. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–10. 3

[ZZZY20] ZHANG Y., ZHANG X., ZHANG T., YIN B.: Crowd motion editing based on mesh deformation. *International Journal of Digital Multimedia Broadcasting 2020* (2020). 2

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

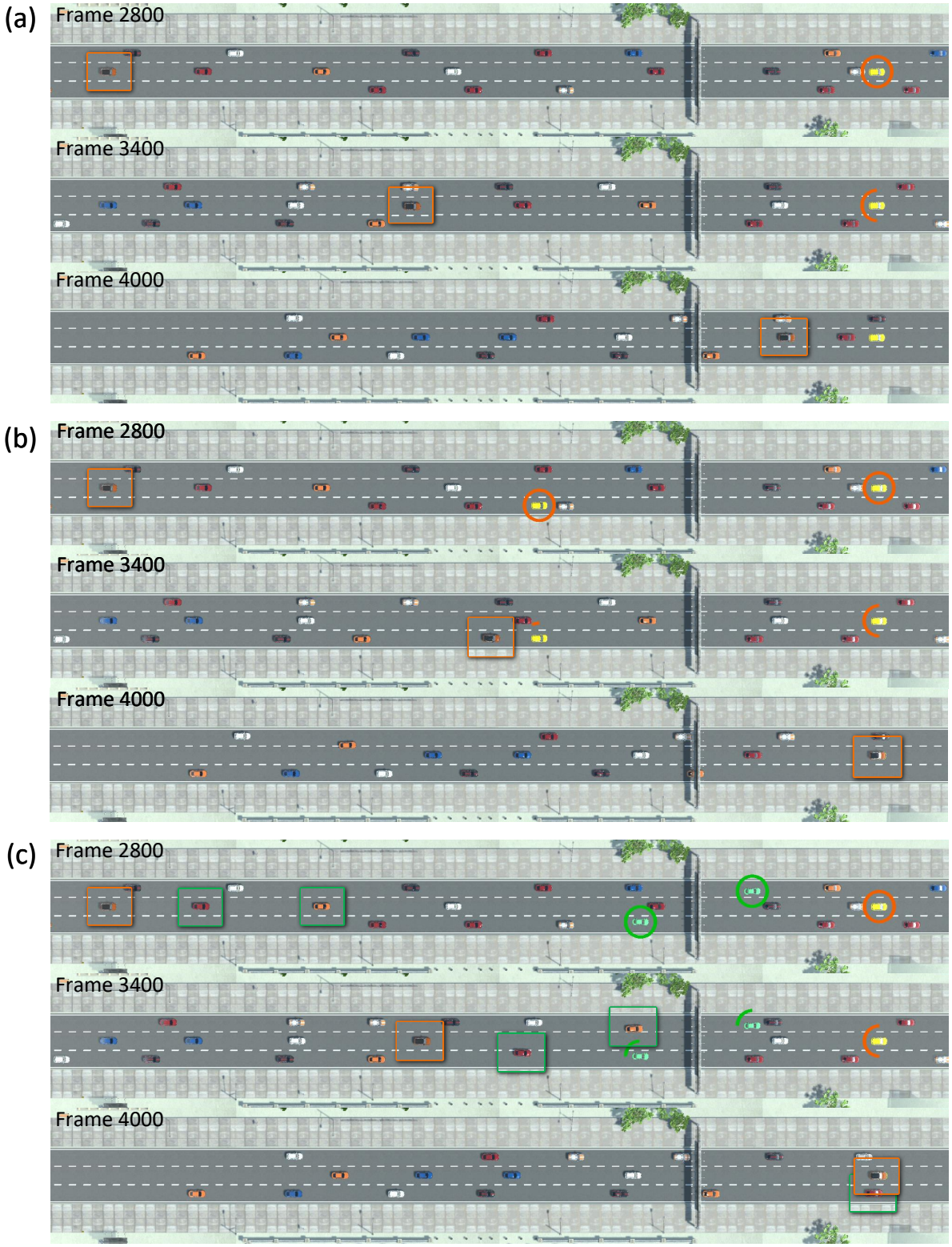

Figure 8: *The first failure case. (a) The vehicle misses the keyframe due to traffic congestion. (b) The first solution is to give the vehicle another keyframe that causes it to overtake the leader. (c) The second solution is to assign keyframes to its leaders, forcing them to yield.*

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

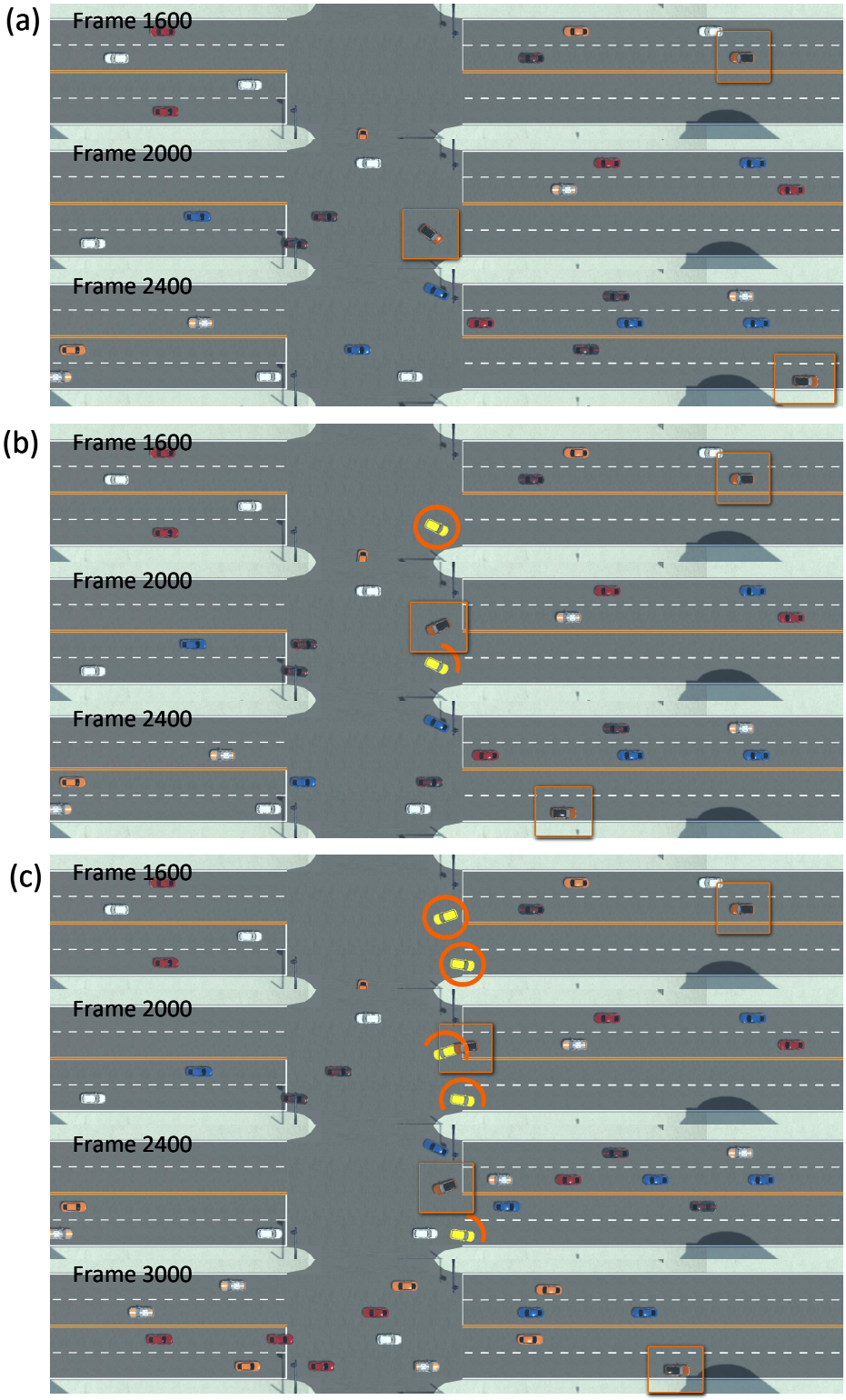

Figure 9: *The second failure case. (a) The vehicle behaves abnormally when performing a high-speed U-turn. (b) The first solution is to assign a keyframe that causes the vehicle to simply decelerate when passing through the curve. (c) The second solution is to assign two keyframes, one to wait for oncoming vehicles and one to complete the U-turn, to further prevent aggressive driving and maintain polite behavior.*

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.