TraEDITS: Diversity and Irregularity-Aware Traffic Trajectory Editing

Yi Han[1], Jiaping Ren[2], Shuning Wang[1], Wenxin Sun[1], Ruigang Yang[2] and Xiaogang Jin[1]

***Abstract*— We present TraEDITS, a novel traffic trajectory editing framework for autonomous vehicle testing, which can generate new traffic behaviors by controlling each vehicle interactively to increase the diversity or irregularity of traffic testing data. Given a traffic flow with its original trajectories, user's edits, environmental constraints, and physical constraints as input, our framework is able to generate an edited traffic flow through a global path planning module and a data-driven microscopic traffic flow simulation module. With the way-points from the user, our global planning module generates lane-level navigation by heuristic-based path planning in discrete space. By taking internal properties of the vehicle, velocity continuity, reference path, and collision avoidance into consideration, our simulation module generates vehicles' motions based on energy optimization driven by real traffic data. Given edits of the desired speed, the lateral deviation, or the reference path, our approach can generate a new trajectory of the selected vehicle, and adjust the surrounding vehicles' trajectories accordingly. As a result, our framework is able to simulate irregular or even rare events existing in real traffic. Meanwhile, our framework can enhance the diversity and irregularity of traffic behaviors and interactions by creating challenging scenarios like swerve, nudge, and U-turn. We validate the usability and plausibility of our framework through extensive experiments and user studies.**

I. Introduction

Though autonomous driving shows great potential in improving the quality of people's lives, many challenges remain to be solved for applying the technique safely. A key challenge is the safety validation of automated vehicles. In recent years, instead of road testing, the driving simulator has become an important tool to generate various traffic conditions in a safe, low-cost, and controllable way [1], [2].

Current traffic simulation software packages, such as SUMO [3], SimMobility [4], Vissim [5] and Carla [6], serve as effective tools for users to generate traffic flows. However, if users want to generate specific cases or refine existing simulation results to increase the diversity of testing data or generate irregular scenarios, they have to start the simulation over and over by empirically adjusting parameters based on the previous results, which is frustrating trial-and-error workflow. We note that similar editing problems exist in crowd animation [7], where artists can edit crowd animation interactively using cage-based deformation [8], mesh-based deformation [9] or using graphical sketches [10]. However, these editing methods are designed for crowds instead of traffic trajectories. In computer graphics, data-driven methods have been increasingly used to simulate traffic behaviors because they are effective in reproducing real-world traffics at the street level. With the help of spatial-temporal data acquired by existing in-road sensors, continuous traffic flows can be reconstructed to reproduce dynamic traffic events [11], [12], [13]. Realistic traffic flows can also be generated by texture synthesis methods [14] and data-driven optimization methods [15]. Though these methods can generate plausible traffic simulations, vehicles have to strictly follow the lane shapes and cannot capture irregular maneuvers like U-turn, nudge or sudden swerve that may occur in the real world.

None of the above mentioned methods is able to meet our demand due to the following main challenges. First, when we edit an existing trajectory, we must take traffic rules into consideration (e.g. safe lane changing, acceleration/deceleration, and road regulations), which cannot be addressed by interactive editing techniques for crowds [7], [8], [9] [10], where agents can move along any trajectories as long as collision avoidance is ensured. Moreover, the editing method must be controllable so that users can generate specific results while only simulating once by controlling and editing vehicles during the simulation process. Second, with user's edits, vehicles may be guided to show irregular motions which are difficult to be captured by previous data-driven simulation methods [11], [12], [13], [14] [15], since they update vehicle's motions strongly depending on static lanes. In addition, such behaviors also need to be regulated according to the physical and kinematic constraints of the vehicles, otherwise the simulation may become unrealistic.

To address the above challenges, we present TraEDITS, a traffic trajectory editing framework to increase the diversity or irregularity of traffic testing data. Inspired by [15], we employ an optimization-based traffic simulation with an additional path planning module, and update vehicles' motions based on path coordinates in order to generate more diverse behaviors and interactions for autonomous vehicle testing. By introducing the interactive editing concept, our framework allows users to control each vehicle and edit final trajectories during the ongoing simulation process. The main contributions of this work are as follows:

- We develop a traffic optimization method by taking existing traffic data, user's edits, environmental constraints and physical constraints into consideration, to facilitate traffic trajectory editing for autonomous vehicle testing.
- We develop a traffic trajectory editing framework to generate ideal trajectories by interactively controlling individuals in the simulation environment, rather than repeatedly simulating by empirically adjusting parameters.
- The framework decouples vehicles' motions from static lanes to path coordinates, so we can generate scenarios with diverse behaviors and interactions which are observed less-frequently in previous methods or datasets and could be used for testing and data augmentation.

[1]State Key Lab of CAD&CG, Zhejiang University.
[2]Inceptio Technology.

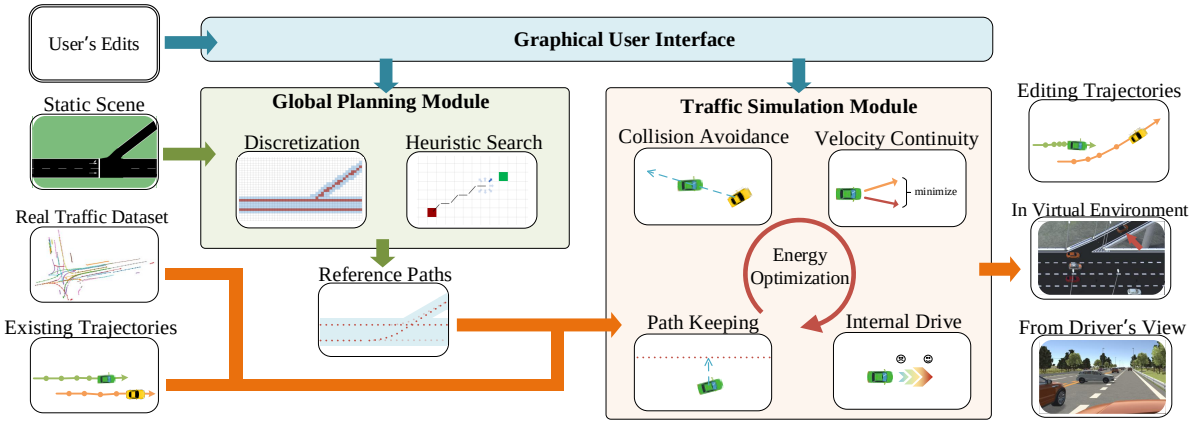

Fig. 1: Overview of our TraEDITS framework with data-driven traffic simulation module and global planning module.

II. TraEDITS Framework

Fig. 1 shows the overview of our TraEDITS framework. We first pre-process inputs for TraEDITS (Section II-A). In the traffic simulation module, we update vehicles' motions based on energy optimization with real traffic data (Section II-B). In the global planning module, we discretize scenarios and plan new reference paths based on heuristic search (Section II-C). We further compute physical constraints given by vehicle kinematics and path geometry (Section II-D) to improve vehicles' behavior. A graphical interface is also built for interactive editing (Section II-E).

A. Coordinate Frame and Scenario Representation

TraEDITS performs in both Cartesian coordinates and Frenet coordinates. We use (s, d) to describe vehicle's position on a path in Frenet coordinates [16], where s denotes the longitudinal displacement along the path, and d denotes the lateral displacement relative to the path. We use $\boldsymbol{p}^C = [x, y]$, $\boldsymbol{v}^C = [v^x, v^y]$, $\boldsymbol{p}^F = [s, d]$, and $\boldsymbol{v}^F = [v^s, v^d]$ to represent position and velocity in Cartesian coordinates and Frenet coordinates, respectively.

We define *Lane* and *Path* differently. *Lane* indicates the specific area that vehicles can drive, which is static and usually given by scene files segment-by-segment. Lane $\mathscr{L}_i = [\mathbf{Q}_i^{\mathscr{L}}, w_i, \mathbf{T}_i]$, where $\mathbf{Q}_i^{\mathscr{L}}$ represents endpoints-pairs of each segment in Cartesian coordinates, w_i represents lane width, and $\mathbf{T}_i$ represents topology between current lane and others, including incoming, outgoing, and adjacent lanes. *Path* is the way from one position to another, usually a set of sequential points linking the start to the terminal and can be generated dynamically. Path $\mathscr{P}_i = [\mathbf{Q}_i^{\mathscr{P}}, \mathbf{S}_i, \mathbf{C}_i]$, where $\mathbf{Q}_i^{\mathscr{P}}$ represents the set of points, $\mathbf{S}_i$ represents interpolation of the points using cubic spline, and $\mathbf{C}_i$ represents a set of curves on the path, including start and end points, center, and radius of each curve. It is noteworthy that $\mathbf{Q}_i^{\mathscr{P}}$ can be easily transformed between Cartesian and Frenet coordinates by $\mathbf{S}_i$.

For a scenario with K lanes, the set of lanes is denoted as $\mathscr{L}^* = [\mathscr{L}_0, \mathscr{L}_1, ...\mathscr{L}_K]$. We transform lanes topology given by $\mathscr{L}^*$ into a directed graph and initialize the set of paths as $\mathscr{P}^* = \mathscr{P}^*_{topo} \cup \mathscr{P}^*_{user}$. $\mathscr{P}^*_{topo} = DFS(\mathscr{L}^*)$ and $\mathscr{P}^*_{user} = \varnothing$ represent the paths generated by lanes topology and user settings, respectively, and DFS represents depth-first search function. For any $\mathscr{P}_i \in \mathscr{P}^*_{topo}$, we obtain the set of points $\mathbf{Q}_i^{\mathscr{P}}$ by gathering the endpoints of lanes in an available depth-first search result, and use them to compute the cubic spline function $\mathbf{S}_i$. Finally, each path $\mathscr{P}_i \in \mathscr{P}^*_{topo}$ is a unique link from the origin of a lane without in-comings to the end of a lane without out-goings.

B. Data-driven Traffic Simulation Module

For the vehicles that have not been edited or affected by other edited vehicles yet, we try not to change their original motions so that we can update them with their original trajectories if there exist. For the vehicles that have already been edited or affected by other edited vehicles, we update them with our simulation algorithm because they have to adjust their motions according to their own changes or edited neighbors.

Our data-driven traffic simulation algorithm based on energy minimization is presented as follows. The candidates generated from the dataset are denoted as $D = \cup_v d_v$, where $d_v = [\boldsymbol{v}^F]$ and $\boldsymbol{v}^F$ is velocity transformed into Frenet coordinates. For a scenario that contains N vehicles, the state of any vehicle i $(i = 1, ..., N)$ at time t is denoted as $s_{i,t} = [\boldsymbol{p}^F_{i,t}, \boldsymbol{p}^C_{i,t}, \boldsymbol{v}^F_{i,t}, o_{i,t}, e_{i,t}, \mathscr{P}_{k,t}, \delta_{i,t}]$, where $\boldsymbol{p}^F_{i,t}$, $\boldsymbol{p}^C_{i,t} \in \mathbb{R}^2$ represent i's current position in Frenet coordinates and Cartesian coordinates, respectively, $\boldsymbol{v}^F_{i,t} \in \mathbb{R}^2$ represents i's current velocity in Frenet coordinates, $o_{i,t} \in \mathbb{R}$ represents i's current orientation as Euler angle, $e_{i,t} \in \mathbb{R}$ represents a free-flow speed that i desires to travel at, $\mathscr{P}_{k,t} \in \mathscr{P}^*$ represents current reference path k that i follows, and $\delta_{i,t} \in \mathbb{R}$ represents current lateral deviation relative to the reference path center that i desires to keep. The state dynamics of vehicle i are formulated as:

$$
\begin{aligned}
\boldsymbol{v}^F_{i,t+1} &= \underset{\boldsymbol{v}^F \in d_v \in D}{\arg\min}\, \mathcal{E}(s_{i,t}), \\
\boldsymbol{p}^F_{i,t+1} &= \boldsymbol{p}^F_{i,t} + \boldsymbol{v}^F_{i,t+1} \cdot \Delta t, \\
[\boldsymbol{p}^C_{i,t+1}, o_{i,t+1}] &= \mathbf{S}_k(\boldsymbol{p}^F_{i,t+1}), \\
[e_{i,t+1}, \mathscr{P}_{k,t+1}, \delta_{i,t+1}] &= \mathcal{G}_{i,t+1},
\end{aligned}
\tag{1}
$$

where $\boldsymbol{v}^F \in d_v \in D$ is an optimal candidate which minimizes the energy function $\mathcal{E}$ and updates i's velocity, Δt is a timestep, and $\boldsymbol{p}^F_{i,t+1}$ is the position in Frenet coordinates at time $t+1$. After obtaining the velocity and the position in Frenet coordinates, we further update the position in Cartesian coordinates and the orientation of i using cubic spline function $\mathbf{S}_k$ of i's current reference path $\mathscr{P}_{k,t}$. The notation $\mathcal{G}_{i,t+1}$ represents the user's edits, and the physical constraints given by kinematics or path geometry for i. The energy function $\mathcal{E}$ is defined as:

$$\mathcal{E} = w_d\mathcal{E}_d + w_c\mathcal{E}_c + w_k\mathcal{E}_k + w_a\mathcal{E}_a, \tag{2}$$

where $\mathcal{E}_d$ is the internal drive term to make vehicles tend to drive at their desired speeds, $\mathcal{E}_c$ is the velocity continuity term to smooth velocity variation, $\mathcal{E}_k$ is the path keeping term to make vehicles tend to follow their reference path, $\mathcal{E}_a$ is the collision avoidance term to ensure vehicles avoid collisions with their neighbors, and w_d, w_c, w_k, w_a are the corresponding weights.

Internal Drive Term: We assume that there is a desired speed at which the driver feels comfortable traveling on roads in sparse traffic flow. The internal drive term is used to make a vehicle reach and keep its desired speed, which is defined as follows:

$$\mathcal{E}_d = e^{\left(\left\|\boldsymbol{v}^F\right\| - e_{i,t}\right)}, \tag{3}$$

where $e_{i,t}$ is the desired speed of vehicle i at time t.

Velocity Continuity Term: Vehicles cannot change their velocity suddenly within a short timestep Δt due to physical limitations. Our velocity continuity term is in accordance with [15], and it can smooth velocity variation of a vehicle both in magnitude and direction.

Path Keeping Term: Vehicles normally tend to keep driving along lane centers in the real world. We prefer the vehicles to drive along paths rather than lanes in our work, for there may exist manually generated paths that are away from lane centers. The path keeping term is defined as:

$$\mathcal{E}_k = e^{\left(\left|v^d \cdot \Delta t - \delta_{i,t}\right|\right)}, \tag{4}$$

where $v^d \in \boldsymbol{v}^F$ is the lateral component of velocity in Frenet coordinates, and $\delta_{i,t}$ is the lateral deviation that i desires to keep away from its reference path at time t.

Collision Avoidance Term: A vehicle should avoid collisions with others that are too close to it. Vehicles can behave more smoothly and realistically when the collision avoidance term is penalized in the energy function, so we treat it as a part of optimization instead of a hard constraint. Here we consider the collisions that may occur within the next timestep:

$$\mathcal{E}_a = \frac{1}{\|\mathscr{N}_t\|} \sum\nolimits_{j \in \mathscr{N}_t} w_{i,j} \cdot e^{\left(l - \left\|\boldsymbol{p}^C_{j,t} - \boldsymbol{p}^C_{i,t}\right\|\right)}, \tag{5}$$

where j is any neighbor vehicle belonging to the neighbors set $\mathscr{N}_t$ at time t, l is a cut-off distance for the maximum look-ahead, $w_{i,j}$ is the weight between each j and i, set to the cosine of angle between i's moving direction and direction of i pointing to j, or zero if it is negative.

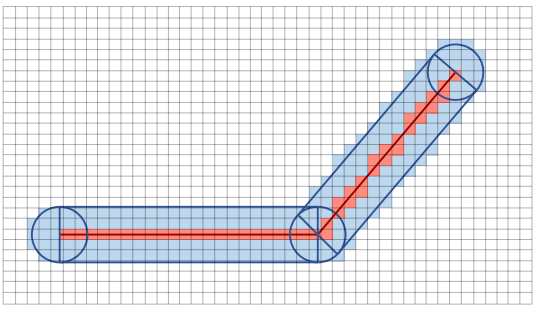

Fig. 2: Lane approximation using capsule based on grid map. The white areas are undriveable, the blue areas are drivable and the red areas are lane centers.

In comparison with [15], we update vehicles' states in Frenet coordinates to decouple their motions from the strong restriction of static lanes, so that more diverse behaviours and interactions are possible. The adjustable values like desired speed, lateral deviation and reference path are further integrated into the energy function, which provides basic interfaces to control and edit vehicles interactively.

C. Global Planning Module

The global planning module provides manipulations of generating self-defined reference paths dynamically. For a given scenario, the space is discretized into the grid map which stores information about lanes and vehicles. Then users can set sequential key points which are used to plan a new path based on heuristic search.

Scenario Discretization: The grid map is constructed on a 2D plane with a given resolution. All the nodes are initialized as undrivable areas. Then according to given static scenario files, we fill the nodes by a specific sign representing drivable areas according to a lane shape approximation. We use a capsule with an oriented bounding box and two circles to represent each lane segment (Fig. 2). Lane centers are given by endpoints-pair of each segment and highlighted with another sign. Finally, median filter is applied to fill in the holes or gaps caused by undersampling. After simulation starts, the grid map is used to store vehicles' positions and updates dynamically to accelerate real-time neighbors search.

Path Planning With Post-Processing: Users can create a new path $\mathscr{P}_i = [\mathbf{Q}^{\mathscr{P}}_i, \mathbf{S}_i, \mathbf{C}_i]$ by setting a sequence of key points. These key points are mapped to specific nodes on the grid map first. Then the whole path is generated segment-by-segment, where the first node of each segment acts as the start point and the second one acts as the goal. We apply the A* search algorithm [17] to plan the path with a modified heuristic function:

$$g(\boldsymbol{n}) = \left\|\boldsymbol{n} - \boldsymbol{n}_{goal}\right\| + \alpha \cdot e^{(\beta \cdot sign)}, \tag{6}$$

where $\left\|\boldsymbol{n} - \boldsymbol{n}_{goal}\right\|$ is the Euler distance between the current node and the goal, and the second term is to make planning process tend to search along lane centers. $sign \in [0, 1, 2]$ is the node' sign mentioned above, where 0 represents undrivable areas, 1 represents drivable areas, and 2 represents lane centers. α, β are adjustable coefficients.

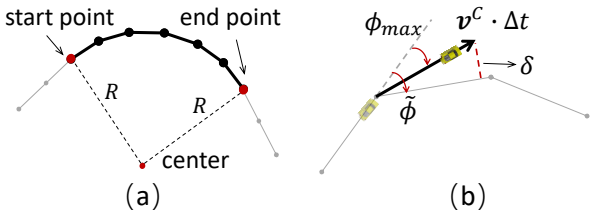

Fig. 3: (a) A segment of identified curve with start and end points, center and radius R, (b) the lateral deviation δ given by the steering angle limitation.

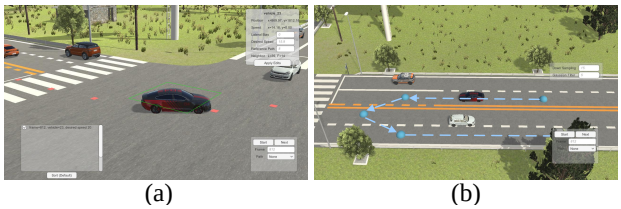

Fig. 4: Graphical user interface of TraEDITS. (a) Main simulation menu, edits stack and vehicle attributes window. (b) Path planning mode with clicking key points.

We further down-sample the result and smooth the points $\mathbf{Q}_i^{\mathscr{P}}$ with a Gaussian filter, and fit them with the cubic spline $\mathbf{S}_i$. Finally the manually generated path $\mathscr{P}_i$ becomes available for vehicles to follow, as $\mathscr{P}^*_{user} = \mathscr{P}^*_{user} \cup [\mathscr{P}_i]$.

D. Extra Physical Constraints

Frenet coordinates representation decouples vehicles' motions from strictly following static lanes, so that vehicles can drive along any path based on the traffic simulation module. However, such deformation may cause distortions since vehicles usually take different actions when following paths with varying shapes in the real world. We further compute the constraints given by kinematics and path geometry to improve vehicles' behaviours according to identified curves on the path.

Curve Identification: As shown in Fig. 3(a), identifying curves for path $\mathscr{P}_i$ is to find out the start and end points, center, and radius of each curve segment to create $\mathbf{C}_i$ based on the path points $\mathbf{Q}_i^{\mathscr{P}}$ and the cubic spline function $\mathbf{S}_i$. We perform equidistant sampling for the whole path and calculate the curvature for each point based on the cubic spline function at first. Then we label any point whose curvature becomes non-zero as the start point, and find its pair-wise end point whose curvature returns to zero. Finally, we calculate the center and the radius as in [18] for each curve whose length is greater than a threshold.

Constraints Computation: A four-wheel vehicle changes its direction by turning the steering wheel. The steering angle ϕ is defined as the angle between the steering wheel direction and the front of the vehicle. In general, the vehicle's steering angle has an upper bound that can be denoted as $|\phi| < \phi_{max}$. When a vehicle drives along a path with an extremely sharp curve, it will certainly lead to a lateral deviation from the path because it is unable to apply the necessary steering angle. A temporary desired lateral deviation is given to handle such

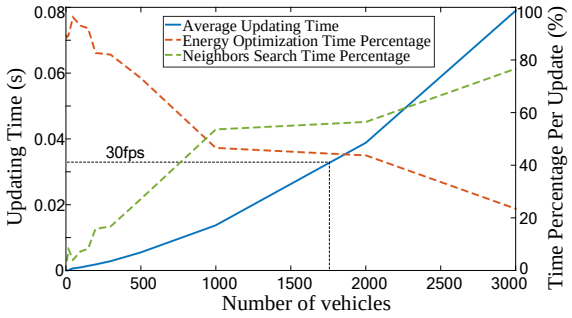

Fig. 5: Performance of the traffic simulation module. We show average update time (s) with parallel implementation, time percentage of energy optimization and neighbors search per frame over different numbers of vehicles.

#Key Points	Path Length	#Heuristic Search Steps	Percentage of Lane Center	Planning Time
2	2571.43	222,066	99.47%	25.61
3	2568.99	142,026	99.03%	8.44
5	2579.02	31,186	97.81%	1.08
	1370.97	5,867	89.33%	0.21
	721.24	1,595	78.50%	0.11
10	2563.17	7,657	90.01%	0.20

TABLE I: Performance of global planning module. We show number of heuristic search steps, percentage of nodes labeled as "lane center" in the final path and planning time (s) by varying number of key points and path length (foot).

situation and computed as:

$$\tilde{\delta}_{i,t} = \left\| \boldsymbol{v}^C_{i,t} \Delta t \right\| \cdot \sin\left(\max(|\tilde{\phi}| - \phi_{max}, 0) \right), \tag{7}$$

where $\tilde{\delta}_{i,t}$ is the temporal lateral deviation for i passing the current curve, $\tilde{\phi}$ is the necessary steering angle to pass the curve along the path without deviations, and ϕ_{max} is the maximum allowed steering angle for the vehicle. If $\left|\tilde{\phi}\right| > \phi_{max}$, the vehicle will deviate from the path rather than driving along it. We give a demonstration in Fig. 3(b).

Moreover, drivers usually decelerate to a slower speed through a turn for both comfort and safety. We compute such safe speed for each curve, and set it as a temporal desired speed to the vehicle passing a certain curve. By following [19], such safe speed is computed as:

$$\tilde{e}_{i,t} = \sqrt{(\mu + \tan\theta)\, gR}, \tag{8}$$

where μ and θ are the friction coefficient and bank angle of the lane, respectively, g is the gravitational acceleration, and R is the radius of the curve.

The physical constraints above are appended to the set $\mathcal{G}_{i,t} = \mathcal{G}_{i,t} \cup [\tilde{\delta}_{i,t}, \tilde{e}_{i,t}]$ with a higher priority than other user's edits and applied to traffic simulation by Eq. 1.

E. Graphical Editing Interface

We design a graphical user interface (Fig. 4) to collect user's edits and feed them to the global planning module and the traffic simulation module. When simulating, users

Parameter	Value	Unit	Description
w_d	1.0	-	the weight for the internal drive term in Eq. 2
w_c	0.5	-	the weight for the velocity continuity term in Eq. 2
w_k	10.0	-	the weight for the path keeping term in Eq. 2
w_a	5.0	-	the weight for the collision avoidance term in Eq. 2
l	50.0	$foot$	the cut-off distance for the maximum lookahead in Eq. 5
α β	20.0 -1.5	-	the coefficients for heuristic-based path planning in Eq. 6
ϕ_{max}	30.0	$degree$	the upper bound for vehicle's steering angle in Eq. 7
μ	1.0	-	the friction coefficient of the lanes in Eq. 8
θ	0.0	$degree$	the bank angle of the lanes in Eq. 8
g	9.8	m/s^2	the gravitational acceleration in Eq. 8

TABLE II: The values of the important parameters.

can move the viewport in 3D space to select a certain vehicle i and show its attributes. Some of its attributes can be edited like desired speed e_i, lateral deviation δ_i, and reference path $\mathscr{P}_k$. Each edit for i is stacked in the set with an expected timestamp t, denoted as $\mathcal{G}_{i,t} = \mathcal{G}_{i,t} \cup [e_{i,t}, \mathscr{P}_{k,t}, \delta_{i,t}]$, and applied by Eq. 1 in the traffic simulation module when the time is right. Under the path planning mode, users can click key points on the lanes which will be sent to the global planning module to create new reference paths. Parameters of path smoothing can also be adjusted here.

III. Experimental Results

A. 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. TraEDITS was implemented in C++, compiled as a x64 dynamic link library and imported into Unity3D where the graphical interface was built. We applied parallel computation to improve performance because vehicle updates in the traffic simulation module are highly parallelizable. The values of some important parameters we used in our experiments are shown in Table II, which were pre-defined in a JSON configure file and loaded by the program.

The real traffic dataset and original trajectories used in our experiments were from *Next Generation Simulation* (NGSIM) [20]. We filtered the original data before using them since they exhibit certain noise artifacts.

There are two scenarios used in our experiments which were generated by *SUMO NetEdit* and originally defined in XML files. The first scenario is an off-ramp on a highway. The off-ramp has a single lane and the main road has four lanes in the same direction. Its size is 3205.67×340.29 (foot) with 12.84% "drivable" areas and 1.23% "lane center" areas. The second scenario is an intersection with a four-lane dual carriageway. Its size is 1877.44×1389.98 (foot) with 5.16% "drivable" areas and 0.51% "lane center" areas. The discretization resolution is 1.0×1.0 (foot) for each.

B. Performance

To evaluate the performance of the traffic simulation module, we performed a series of experiments with different numbers of vehicles. As shown in Fig. 5, the computation time scales almost linearly with the number of simulated vehicles, and it can stay at 30 frames per second (fps) when the number of vehicles is around 1,800 in the test scenarios. We give a detailed percentage of average computing time (ms) of energy optimization and neighbors search. The time percentage of energy optimization decreases and the time percentage of neighbors search increases when the number of vehicles becomes large. In fact, the time of energy optimization almost remains unchanged since it basically depends on the size of dataset, while there are more neighbors to search when placing more vehicles in the constant area.

To evaluate the performance of the global planning module, we randomly plan different paths by choosing different key points in the scenarios. Firstly, the path length is kept and the planning time over different numbers of key points are given. Then, under constant 5 key points, the planning time is computed with varying path lengths. The number of heuristic search steps and the percentage of the nodes labeled as "lane center" in the final planning path are also counted. As shown in Table I, the total planning time decreases if we give more key points or make the path shorter. For a very long path with only 2 specified key points (starting point and goal), the search steps of path planning will significantly increase since much of them are failed trials. Specifically, the percentages of lane center of the planning paths are all larger than 70% in our experiments, which validates that our modified heuristic function shown in Eq. 6 is effective. This value decreases when the total path length gets shorter or more key points tend to make the path away from lane centers.

C. Editing Cases

We designed some cases to show the results of trajectories editing based on existing data by TraEDITS.

The first two cases were generated in the off-ramp scenario. In the first case, we generated a new path crossing four lanes from the furthest lane to off-ramp in a short distance. Then we assigned the path to the specific vehicle driving along the furthest lane, which made it act like deciding to leave the highway suddenly when nearly missing the off-ramp (see Fig. 6). We also generated another swerve-case in the crowded environment, where the specific vehicle

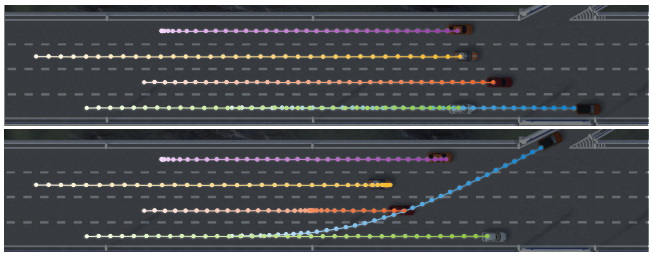

Fig. 6: The original trajectories (top) and the edited trajectories (bottom) for a sudden swerve on the highway.

Fig. 7: The original trajectories (top) and the edited trajectories (bottom) for a swerve in a crowded environment.

originally derived along the lane, is guided to cross through the congested traffic flow and leave the highway with a sharper path (see Fig. 7).

The last cases were generated in the intersection scenario. In the third case, we let the vehicle which got stuck by leaders in both available lanes deviate from lane center slightly and accelerate, leading it to nudge and overtake successfully (see Fig. 8). In the fourth case, we generated a new path that can make vehicles take a U-turn to the opposite-direction lane, and assigned the path to the specific vehicle. The vehicle decelerated to pass the curve smoothly and then returned to the original state, while the other vehicles slowed down to wait for it (see Fig. 9).

These examples show that our editing framework can increase the diversity and irregularity of traffic behaviors which are rarely seen in real captured traffic data. The generated cases can also be applied in virtual environment for autonomous driving testing and data augmentation. Fig. 11 shows snapshots of the U-turn result from driver's view in Unity3D.

D. User Study

We conducted two user studies. The first one is to evaluate the usability of TraEDITS, and the second one is to evaluate the plausibility of generated results. Both user studies were taken by 18 participants (13 males and 5 females).

Usability Evaluation: In the first user study, participants needed to complete the three following tasks in TraEDITS:

- *Overtake*: Let the specified vehicle overtake the vehicle in front based on the original trajectories.
- *U-turn*: Let the specified vehicle make a U-turn based on the original trajectories.

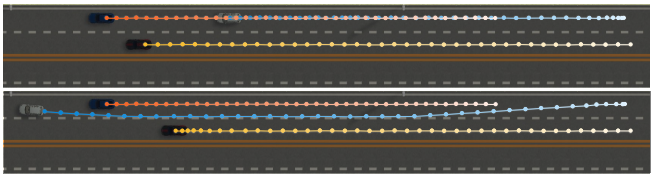

Fig. 8: The original trajectories (top) and the edited trajectories (bottom) for nudging and overtaking.

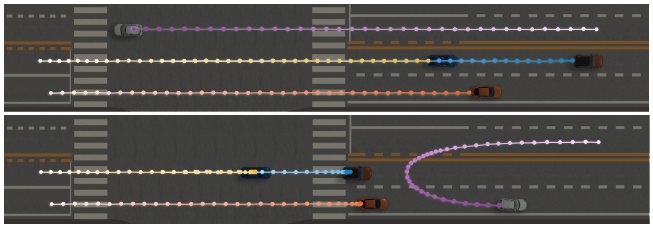

Fig. 9: The original trajectories (top) and the edited trajectories (bottom) for making a U-turn at the intersection.

- *Schedule*: Make the vehicles coming from four directions (15 vehicles per direction) pass safely at the intersection without traffic lights during the simulation.

After accomplishing all the tasks, participants were asked to finish a questionnaire with a six-dimensional NASA Task Load Index [21], [22] (mental demand, physical demand, temporal demand, performance and frustration). Participants had to give a score from 0 to 5 for each term. The questionnaire also has three statements, and we use 5-point Likert scale to let participants give a score to each statement from 1 to 5 for "totally disagree" to "totally agree".

Fig. 10(a) shows the average scores of the six-dimensional NASA-TLX for the three tasks. Except for the third task, mental demand, physical demand, temporal demand, effort and frustration are obviously lower than 2 while performance of all the tasks are higher than 4. The reason why participants spent more energy on the third task is that the number of vehicles that were required to be edited was much more than those in the other two tasks. Generally speaking, TraEDITS allows users to control vehicles and edit their trajectories intuitively with a little effort even for the first time.

Fig. 10(b) shows participants' attitudes towards the three appending statements. We performed one-sample t-tests for the three instances by assuming the mean score of each term is larger than 3 (remain neutral). Overall, participants significantly agree with that "*TraEDITS is beginner-friendly with intuitive operations*" ($mean = 4.667$, $t(17) = 14.577$, $q < 0.001$), "*It is easy to use TraEDITS to complete the tasks*" ($mean = 4.222$, $t(17) = 7.083$, $q < 0.001$) and "*The editing results generated by TraEDITS are satisfactory*" ($mean = 4.222$, $t(17) = 6.414$, $q < 0.001$).

Plausibility Evaluation: In the second user study, we compare the U-turn results generated by TraEDITS with the ones by Heter-sim [15], and the U-turn results generated by TraEDITS with different editing times. We designed three comparison pairs by showing two side-by-side pre-recorded videos for each. Participants were asked to indicate their

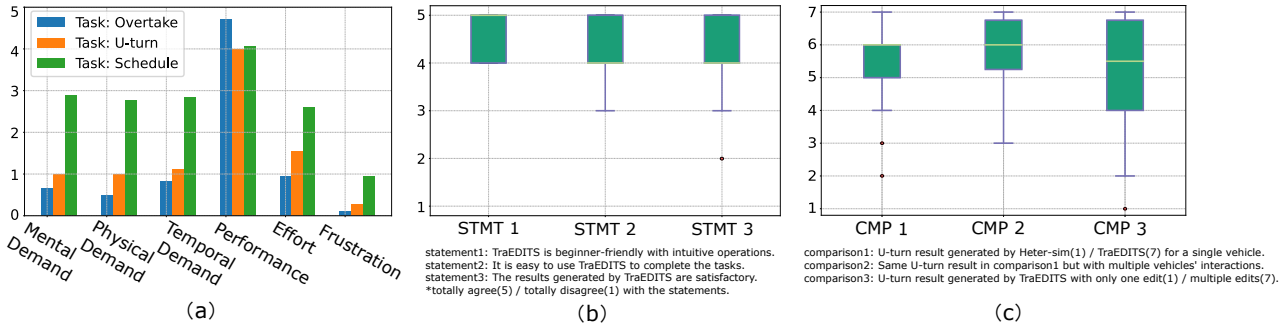

Fig. 10: (a) Average scores of the six-dimensional NASA-TLX for participants to accomplish the three tasks. (b) The scores of participants' attitudes towards the three statements. (c) The detailed scores for the three comparison pairs.

Fig. 11: Snapshots of the U-turn result from driver's view.

preference for a video using a 7-point Likert scale, with 1 indicating a strong preference for the left video, 7 for the right and 4 for no preference.

Fig. 10(c) shows the scores for comparisons. We performed one-sample t-tests for the three instances by assuming the mean score of each term is larger than 4 (no preference). The key frames of the U-turn results used for following comparisons are also shown in Fig. 12.

For the first comparison, we showed U-turn results generated by Heter-sim (left) and TraEDITS (right) for a single vehicle in the scene. The score shows that our result is significantly better ($mean = 5.500$, $t(17) = 4.467$, $p < 0.001$) because the vehicle in our result can slow down to a safe speed and deviate slightly from path centers while the other one kept a high speed all the way.

For the second comparison, we showed the same U-turn results in comparison 1 but included other vehicles' interactions in the scene. The score also shows our result looks significantly more better even with interactions ($mean = 5.500$, $t(17) = 3.768$, $p = 0.0015 < 0.05$).

For the third comparison, we showed U-turn results generated by TraEDITS with a single edit version (left) and a multiple edits version (right). In the single edit version, the edited vehicle made a U-turn while ignoring the other vehicles which made all others wait for it (see Fig. 12 middle). In the multiple edits version, we changed the edited vehicle's behaviour to wait for others to pass by first and then finish its U-turn (see Fig. 12 bottom). The score shows that participants significantly prefer to the result with iterative edits ($mean = 5.056$, $t(17) = 2.587$, $p = 0.019 < 0.05$).

Specifically, most of participants thought it is not only dangerous to ignore vehicles in the target lane when performing lane-change, especially for U-turn, but socially it can also be perceived as highly aggressive driving. So the result in which the vehicle waits until all the other oncoming vehicles have passed gains more recognition. Such editing can be seen as unexpected driving situation refinement caused by driving behaviors which is more implicit than simple traffic violations.

IV. Conclusion

We have presented TraEDITS, a new traffic trajectory editing framework for autonomous vehicle testing, to increase the diversity or irregularity of traffic testing data. Our framework integrates data-driven traffic simulation and global path planning seamlessly, and it can deal with custom constraints such as the original traffic trajectories, user's edits, environmental constraints, and physical constraints. We update vehicles based on optimization with real-world data in the traffic simulation module, and plan reference paths with discretized grid maps and a modified A* algorithm in the global planning module. Extra constraints from steering angle and road geometry are also computed. We provide a graphical interface to let users edit and generate ideal trajectories intuitively.

Though the proposed framework is promising, it is still preliminary and can be improved in several ways. Firstly, vehicles' motions for passing curves can be extracted from real-world data instead of empirically modeling them with steering angle and road geometry. Secondly, reference paths are better planned in the continuous environment since the results based on grid maps strongly depend on the discretization resolution. Thirdly, a more efficient manipulation method like bulk operation is necessary when editing large-scale traffic flows.

References

[1] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[2] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M. C. Lin, and Z. Deng, "A survey on visual traffic simulation: Models, evaluations, and applications in autonomous driving," in *Computer Graphics Forum*, vol. 39, no. 1. Wiley Online Library, 2020, pp. 287–308.

[3] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *International journal on advances in systems and measurements*, vol. 5, no. 3&4, 2012.

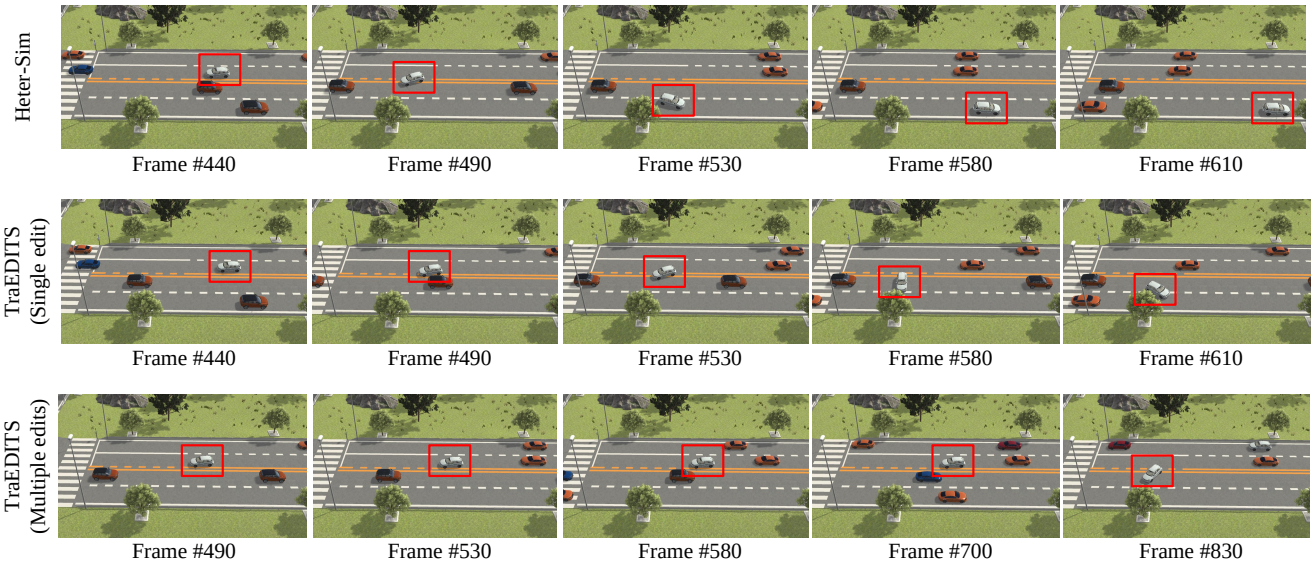

Fig. 12: The U-turn results generated by Heter-sim (top) and TraEDITS. There are two versions generated by TraEDITS. The first is generated with a single edit for only making a U-turn (middle), and the second is generated with multiple edits for making a U-turn after decelerating to wait for others first (bottom).

[4] M. Adnan, F. C. Pereira, C. M. L. Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva, "Simmobility: A multi-scale integrated agent-based simulation platform," in *95th Annual Meeting of the Transportation Research Board Forthcoming in Transportation Research Record*, 2016.

[5] "Vissim," http://vision-traffic.ptvgroup.com, 2020.

[6] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[7] T. Kwon, K. H. Lee, J. Lee, and S. Takahashi, "Group motion editing," *ACM Transactions on Graphics (TOG)*, vol. 27, no. 3, pp. 1–8, 2008.

[8] J. Kim, Y. Seol, T. Kwon, and J. Lee, "Interactive manipulation of large-scale crowd animation," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–10, 2014.

[9] Y. Zhang, X. Zhang, T. Zhang, and B. Yin, "Crowd motion editing based on mesh deformation," *International Journal of Digital Multimedia Broadcasting*, vol. 2020, 2020.

[10] L. R. Montana and S. Maddock, "Sketching for real-time control of crowd simulations," in *Proceedings of the Conference on Computer Graphics & Visual Computing*, 2017, pp. 81–88.

[11] J. Sewall, J. Van Den Berg, M. Lin, and D. Manocha, "Virtualized traffic: Reconstructing traffic flows from discrete spatiotemporal data," *IEEE transactions on visualization and computer graphics*, vol. 17, no. 1, pp. 26–37, 2010.

[12] D. Wilkie, J. Sewall, and M. Lin, "Flow reconstruction for data-driven traffic animation," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[13] W. Li, D. Wolinski, and M. C. Lin, "City-scale traffic animation using statistical learning and metamodel-based optimization," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 6, pp. 1–12, 2017.

[14] Q. Chao, Z. Deng, J. Ren, Q. Ye, and X. Jin, "Realistic data-driven traffic flow animation using texture synthesis," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 2, pp. 1167–1178, 2017.

[15] J. Ren, W. Xiang, Y. Xiao, R. Yang, D. Manocha, and X. Jin, "Heter-sim: Heterogeneous multi-agent systems simulation by interactive data-driven optimization," *IEEE transactions on visualization and computer graphics*, 2019.

[16] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[17] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[18] Z. Li, M. V. Chitturi, A. R. Bill, and D. A. Noyce, "Automated identification and extraction of horizontal curve information from geographic information system roadway maps," *Transportation research record*, vol. 2291, no. 1, pp. 80–92, 2012.

[19] C. Gámez Serna and Y. Ruichek, "Dynamic speed adaptation for path tracking based on curvature information and speed limits," *Sensors*, vol. 17, no. 6, p. 1383, 2017.

[20] "Next generation simulation," https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm, 2013.

[21] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Advances in psychology*. Elsevier, 1988, vol. 52, pp. 139–183.

[22] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9. Sage publications Sage CA: Los Angeles, CA, 2006, pp. 904–908.