Graph-based Scenario-Adaptive Lane-Changing Trajectory Planning for Autonomous Driving

Qing Dong[1], Zhanhong Yan[2], Kimihiko Nakano[2], Xuewu Ji[1], and Yahui Liu[1]

***Abstract*—Trajectory planning is one of the key challenges to the rapid and large-scale deployment of autonomous driving. The lane-changing trajectory planning algorithm for autonomous driving is typically formulated as a optimization process of a cost function, which can be challenging to manually tune for different traffic scenarios. This paper presents a graph-based scenario-adaptive lane-changing trajectory planning approach that overcomes this challenge. Specifically, the cost function recovery method based on maximum entropy inverse reinforcement learning (IRL) is proposed to recover the cost functions of the all demonstrated lane-changing trajectories, and the cost function database is constructed. Then, the scenario matching model based on spatial-temporal graph convolutional network (ST-GCN) is proposed to match the recovered cost functions with the traffic scenarios, making the lane-changing trajectory planning method scenario-adaptive. Our proposed method is evaluated through simulations on the well-known NGSIM dataset and experiments on two typical lane-changing scenarios on the autonomous driving platform. The results show that our method is capable of learning the lane-changing cost function from demonstration and performing scenario-adaptive lane-changing trajectory planning.**

***Index Terms*—Autonomous driving, trajectory planning, scenario-adaptive, inverse reinforcement learning (IRL), spatial-temporal graph convolutional network (ST-GCN).**

I. Introduction

AUTONOMOUS driving has garnered great interest and has achieved remarkable progress in the past decade as a means to reduce traffic accidents, improve traffic efficiency, and provide convenient mobility [1]. To achieve these goals, autonomous driving requires accurate environmental perception and prediction, reasonable planning and precise control [2]. Among them, the planning system of autonomous driving is a crucial component of the safe and practical autonomous driving system, which is typically partitioned into three modules, namely global route planning [3], behavior planning [4], and trajectory planning [5].

The trajectory planning module has significant responsibility for safety and confronts substantial challenges, especially in complex traffic environments. While optimization-based methods have exhibited potential by generating optimal continuous trajectories via cost function optimization, it is unsuitable to plan trajectories under varying interactive traffic scenarios using a single cost function. The manual tuning of the weight parameters of cost functions across all traffic scenarios is both experience-dependent and time-consuming.

To address this issue, the trajectory planning method should be capable of automatically adjusting weights to accommodate a variety of traffic scenarios. Designing such a trajectory planning algorithm poses significant challenges. However, experienced human drivers have demonstrated proficiency in managing a myriad of driving scenarios with ease. Thus, an appealing alternative lies in learning driving policies or cost functions from human driving experts.

In this paper, a graph-based scenario-adaptive lane-changing trajectory planning method for autonomous driving is proposed. The proposed lane-change trajectory planning method is designed to learn the cost function weight vector for specific lane-changing scenarios and then generate lane-changing trajectories through a cost function optimization process. The key contributions of this paper can be summarized as follows:

- The lane-change trajectory features of the cost function are meticulously designed, and a method based on Inverse Reinforcement Learning (IRL) is proposed to recover cost function weight vectors of demonstrated trajectories in the natural driving dataset.
- A scenario matching model based on the Spatial-Temporal Graph Convolutional Network (ST-GCN) is proposed, aiming to model the relationship between the graphically represented traffic scenarios and the cost function weight vectors.

The rest of this paper is organized as follows: Section II introduces the related works. Section III expatiates the problem formulation. Section IV describes our proposed trajectory planning method and implementation details. The proposed method is evaluated in Section V. Finally, we conclude this paper in Section VI.

II. Related Works

A. Trajectory Planning

Numerous trajectory planning techniques have been proposed, categorized broadly into search-based [6], sampling-based [7], interpolating curve fitting-based [8], optimization-based [9], and machine learning-based [10]. Search-based and sampling-based trajectory planning methods are discrete in that they discretize the state space into a grid or sample a finite

Manuscript received: March 18, 2023; Revised May 27, 2023; Accepted July 3, 2023. This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China (NSFC) under grant number 52221005, the Tsinghua-Toyota Joint Research Fund and China Scholarship Council (202206210211). (Corresponding author: Yahui Liu.)

[1]The authors are with the School of Vehicle and Mobility, Tsinghua University Beijing 100084, China (e-mail: dq19@mails.tsinghua.edu.cn; jixuewu@tsinghua.edu.cn; liuyahui@tsinghua.edu.cn). [2]The authors are with the Institute of Industrial Science, The University of Tokyo, Tokyo 153-0041, Japan (e-mail: yanzhjob@iis.u-tokyo.ac.jp; knakano@iis.u-tokyo.ac.jp).

Digital Object Identifier (DOI): 10.1109/LRA.2023.3300250.

number of times in the space. While curve-fitting methods can produce smooth trajectories, maintaining optimality proves challenging. Game-theoretic approaches [11], [12], [13] simultaneously generates predictions for surrounding vehicles and optimal response plans for the autonomous ego vehicle based on the cost of the task. In order to enhance interpretability and ease of adjustment, the Baidu Apollo EM Motion Planner [14] adopts a hierarchical approach to its prediction and planning modules. As for the lane-changing scenarios of autonomous vehicles, EM Motion Planner generates lane-changing planning trajectories by optimizing a pre-defined cost function. Liu et al. [15] propose a dynamic lane-changing trajectory planning method based on cost function optimization. It is note-worthy that these studies do not achieve adaptive adjustment of the cost function tailored to different lane-changing scenarios.

B. Inverse Reinforcement Learning

IRL approaches have been developed for recovering optimization cost functions from demonstrations, which aim to learn a mapping from perceptual features to cost functions rather than from perceptual features to actions [16], [17]. Silver et al. [18] propose maximum margin planning (MMP), an IRL variant for producing properly coupled cost functions of discrete state-action pairs from the demonstration that minimize the difference between the observed and optimal trajectory. In addition to performing IRL on discrete states and action spaces, several studies consider trajectories in continuous state space. Kuderer et al. [19] use feature-based IRL to find the parameters of a cost function that best fits an observed driving style and use it to efficiently plan trajectories for autonomous vehicles under the learned driving style. These studies indicate that learning from demonstrations is superior to manual parameter tuning. However, when applying the cost function to various traffic scenarios, these methods do not always accurately describe and extract traffic scenario features, resulting in an incomplete match between traffic scenarios and the cost function.

C. Graph Neural Networks

Graph-based interactive representation can deal with the complex traffic scenarios, and graph neural network (GNN) attracts great interest in the feature extraction of the traffic scenarios [20]. Mohamed et al. [21] extends the spatial graph convolution network to spatial-temporal graph convolution network (ST-GCN) to incorporate the spatial-temporal information of pedestrian trajectories. Zhou et al. [22] proposed an attention-based spatio-temporal graph neural network (AST-GNN), using spatial graph neural network for interaction modeling, and temporal graph neural network for motion feature extraction. Li et al. [23] addressed the issue of multi-robot path planning by applying GNN, thereby facilitating effective feature communication among robots engaged in the task. Yu et al. [24] accelerated motion planning by reducing collision checks in sample-based motion planning methods, accomplished by training a GNN to perform path exploration and smoothing. These studies demonstrate the marked advantages of GNN in the feature extraction of the traffic scenarios.

III. Problem Formulation

A lane-changing trajectory planning method based on the automatic tuning technology of the trajectory optimization cost function is the focus of this work. Typically, the trajectory optimization cost function is represented as [17]:

$$C_p = \boldsymbol{\theta}_p^{\mathrm{T}} \boldsymbol{f}_\xi \tag{1}$$

where $\boldsymbol{f}_\xi = [f_{\xi,1}, f_{\xi,2}, \cdots, f_{\boldsymbol{\xi},k}]$ is the K-dimensional feature vector used for capturing the expectation related to needs of the planned trajectory $\boldsymbol{\xi}$, e.g., driving efficiency and driving comfort. $\boldsymbol{\theta}_p = [1, \theta_{r,1}, \ldots, \theta_{r,(k-1)}]$ is the weight vector used for balancing the contribution of the features in the feature vector.

To achieve scenario-adaptability, the weight vector $\boldsymbol{\theta}_p$ should be a function of the traffic scenario $\boldsymbol{s}$. The parameters of this function should be acquired offline. Specifically, as shown in Fig. 1, a collection of N lane-changing scenarios, represented as D_{lc}, is provided:

$$D_{lc} = \{(\boldsymbol{s}_1, \boldsymbol{p}_1), (\boldsymbol{s}_2, \boldsymbol{p}_2), \ldots, (\boldsymbol{s}_N, \boldsymbol{p}_N)\} \tag{2}$$

where $\boldsymbol{s}_i$ and $\boldsymbol{p}_i$ respectively contain the trajectories of both the lane-changing ego vehicle (Vehicle 0) and surrounding vehicles (Vehicle 1 to M) in historical and future time domain from the lane-changing start moment, which can be expressed as:

$$\boldsymbol{s}_i = [\boldsymbol{\xi}_h^{(0)}, \boldsymbol{\xi}_h^{(1)}, \cdots, \boldsymbol{\xi}_h^{(M)}] \tag{3}$$

$$\boldsymbol{p}_i = [\boldsymbol{\xi}_p^{(0)}] \tag{4}$$

where $\boldsymbol{\xi}_h^{(i)}$ represents the historical trajectories of vehicle i, comprising T_h historical time steps, while $\boldsymbol{\xi}_p^{(0)}$ denotes the future trajectories, consisting of T_f future time steps for the ego vehicle. These can be expressed as follows:

$$\boldsymbol{\xi}_h^{(i)} = [x_{-T_h}^{(i)}, y_{-T_h}^{(i)}, \cdots, x_0^{(i)}, y_0^{(i)}] \tag{5}$$

$$\boldsymbol{\xi}_p^{(0)} = [x_1^{(0)}, y_1^{(0)}, \cdots, x_{T_f}^{(0)}, y_{T_f}^{(0)}] \tag{6}$$

where $x_t^{(i)}$ and $y_t^{(i)}$ are the longitudinal and lateral coordinates of the vehicle i at the time step t, respectively.

Our task is to learn the matching model with parameter $\boldsymbol{W}$ of scenario $\boldsymbol{s}_i$ and weight vector $\boldsymbol{\theta}_{p,i}$ behind the future scenario $\boldsymbol{p}_i$, which can be described as:

$$\boldsymbol{\theta}_{p,i} = \mathrm{MODEL}_{\boldsymbol{W}}(\boldsymbol{s}_i) \tag{7}$$

In this way, the weight vector $\boldsymbol{\theta}_{p,cur}$ can be computed according to the current traffic scenario $\boldsymbol{s}_{cur}$ in real-time in the online process. Then the cost function in Equation (1) can be constructed, and the optimization problem can be formulated as:

$$^*\boldsymbol{\xi}_p^0 = \underset{\boldsymbol{\xi}_p^0}{\arg\min}\, C_p(\boldsymbol{\theta}_{p,cur}, \boldsymbol{f}_\xi) \tag{8}$$

$$s.t. : (x_t^{(0)}, y_t^{(0)}) \in D_{safe}, t = 1, 2, \ldots, T_f$$

where $^*\boldsymbol{\xi}_p^0$ is the planned trajectory of the autonomous ego vehicle, with the same form as Equation (4). D_{safe} represents the safe space within the boundaries of the road, excluding any obstacles [14].

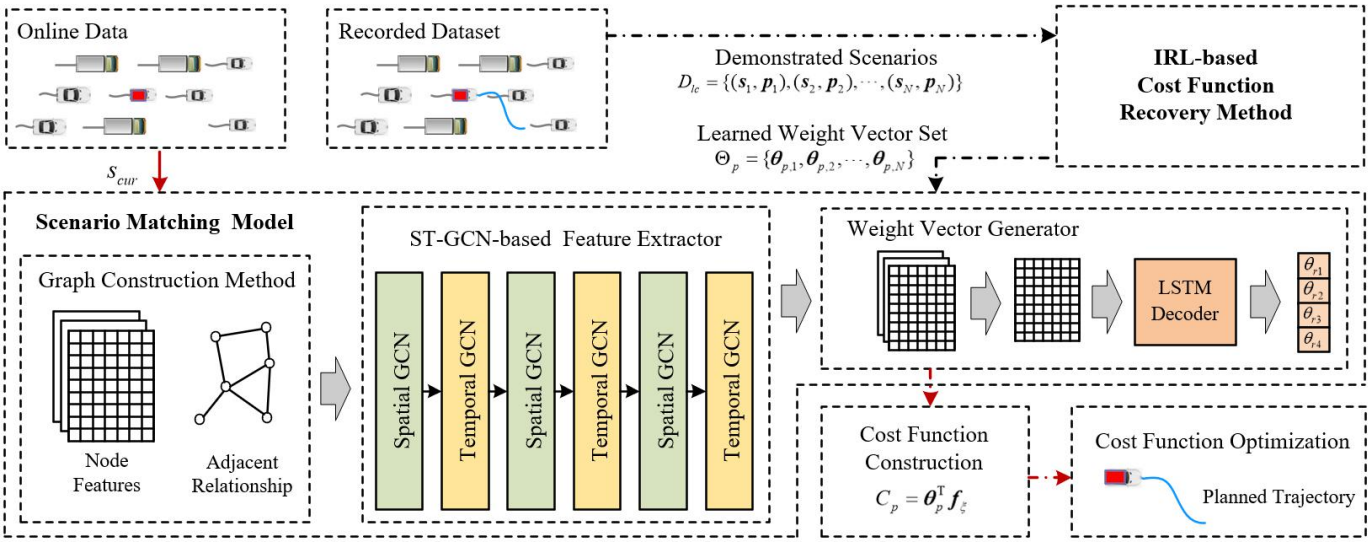

Fig. 1. The framework of the graph-based scenario-adaptive lane-changing trajectory planning method.

IV. Method

This section will discuss the structure and principle of the graph-based scenario-adaptive lane-changing trajectory planning algorithm, as shown in Fig. 1.

A. Cost Function Recovery Method

The cost function recovery method learns the cost function of the lane-changing process in D_{lc} through the maximum entropy IRL framework. It results in a weight vector set comprising of $\boldsymbol{\theta}_{p,i}$, representing the optimal trajectory planning strategy for scenario $\boldsymbol{s}_i$, which can be expressed as:

$$\Theta_p = \{\boldsymbol{\theta}_{p,1}, \boldsymbol{\theta}_{p,2}, \ldots, \boldsymbol{\theta}_{p,N}\} \tag{9}$$

Based on the maximum entropy IRL principle and the Lagrange multiplier method, the trajectory probability distribution model can be represented as [17]:

$$p(\boldsymbol{\xi}|\boldsymbol{\theta}_p) = \frac{\exp(-\boldsymbol{\theta}_p^{\mathrm{T}} \boldsymbol{f}_\xi)}{\int_{\boldsymbol{\xi}} \exp\left(-\boldsymbol{\theta}_p^{\mathrm{T}} \boldsymbol{f}_\xi\right) \mathrm{d}\boldsymbol{\xi}} \tag{10}$$

The goal of maximum entropy IRL is to adjust the $\boldsymbol{\theta}_p$ to maximize the likelihood of the demonstrated trajectory $\hat{\boldsymbol{\xi}}$ under the probability model $p(\boldsymbol{\xi}|\boldsymbol{\theta}_p)$:

$$\boldsymbol{\theta}_p^* = \arg\max_{\boldsymbol{\theta}_p} \boldsymbol{J}(\boldsymbol{\theta}_p) = \arg\max_{\boldsymbol{\theta}_p} \log p(\hat{\boldsymbol{\xi}} \mid \boldsymbol{\theta}_p) \tag{11}$$

The distribution parameter $\boldsymbol{\theta}_p$ must be determined through numerical iteration. Utilizing the principle of inverse optimal control (IOC) [19], the expected trajectory features are approximated using the features of the most probable trajectory, thus circumventing the need to integrate all potential trajectories in a high-dimensional continuous space. The gradient of the likelihood function can then be computed as follows:

$$\nabla_{\boldsymbol{\theta}_p} \boldsymbol{J}(\boldsymbol{\theta}_p) = \boldsymbol{f}\left(\arg\max_{\boldsymbol{\xi}} p\left(\boldsymbol{\xi} \mid \boldsymbol{\theta}_p\right)\right) - \boldsymbol{f}_{\hat{\boldsymbol{\xi}}} \tag{12}$$

Then, the trajectory feature vector $\boldsymbol{f}_\xi$ including the traffic efficiency, comfort and risk are designed.

1) Traffic Efficiency: To capture the driver's desire for a quick arrival, traffic efficiency features are divided into longitudinal and lateral driving efficiency components.

$$f_{ev_x} = \frac{\int_t \|\dot{x}(t) - v_{des}\|^2 \, \mathrm{d}t}{\int_t \|v_{des}\|^2 \, \mathrm{d}t} \tag{13}$$

$$f_{ey} = \frac{\int_t \|y(t) - y_{tar}\|^2 \, \mathrm{d}t}{\int_t \|y_{tar}\|^2 \, \mathrm{d}t} \tag{14}$$

where v_{des} is the speed limit and y_{tar} is the target lateral displacement (the width of the lane), both are non-zero constants.

2) Comfort: Longitudinal acceleration feature f_{a_x} and lateral acceleration feature f_{a_y} are set to measure the comfort:

$$f_{a_x} = \frac{\int_t \|\ddot{x}(t)\|^2 \mathrm{d}t}{\int_t \|a_{x,max}\|^2 \mathrm{d}t} \tag{15}$$

$$f_{a_y} = \frac{\int_t \|\ddot{y}(t)\|^2 \mathrm{d}t}{\int_t \|a_{y,max}\|^2 \mathrm{d}t} \tag{16}$$

where $a_{x,max}$ and $a_{y,max}$ respectively represent the maximum longitudinal and lateral accelerations of the vehicle, which are on-zero constants derived from empirical data.

3) Risk: The risk attribute, derived from both preceding and following vehicles in the current and target lanes, is conceived as an exponential function of the time headway (THW).

$$f_{risk} = f_{risk_p} + f_{risk_f} \tag{17}$$

where

$$f_{risk_p} = \int_t \exp\left[-\left(\frac{x_p(t) - x(t)}{\dot{x}(t)}\right)\right] \mathrm{d}t$$

$$f_{risk_f} = \int_t \exp\left[-\left(\frac{x(t) - x_f(t)}{\dot{x}_f(t)}\right)\right] \mathrm{d}t$$

where $x_p(t)$ and $x_f(t)$ are the longitudinal positions of the preceding and the following vehicles in corresponding lane, respectively. The calculation of risk features requires the future trajectory predictions of surrounding vehicles, which

are obtained from the prediction module during the iterative process of prediction and planning [20].

Based on expected feature matching of lane-changing trajectory, the procedures of the IRL process is as follows [16]. The cost function is initially constructed using a weight vector $\boldsymbol{\theta}_0$ with all elements set to 1. An optimal trajectory is generated by minimizing this cost function, the details of which are elaborated in Subsection C. The gradient as defined in Equation (12) is then computed based on the newly generated trajectory and the demonstrated trajectory. To ensure that all elements of the weight vector $\boldsymbol{\theta}_p$ remain positive throughout the learning process, the direction of the i-th element of the gradient, represented by $\nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)_i$, dictates the update of the i-th element of the weight vector donated by $\theta_{p,i}$. This update process can be represented as:

$$\theta_{p,i} = \begin{cases} \theta_{p,i} + \alpha \cdot \nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)_i & \nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)_i > 0 \\ \theta_{p,i} \cdot \exp(\beta \cdot \nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)_i) & \nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)_i \leq 0 \end{cases} \tag{18}$$

where α and β are the learning rates. This iterative process continues until the 2-norm of $\nabla_{\boldsymbol{\theta}_p}\boldsymbol{J}(\boldsymbol{\theta}_p)$ falls below a predefined threshold. The final optimized weight vector $\boldsymbol{\theta}_p^*$ is obtained and incorporated into the set of weight vectors represented by Equation (9).

B. Scenario Matching Model

The scenario matching model based on ST-GCN is proposed to establish a relationship between the cost function weight vector and the traffic scenario, thus enabling adaptive lane-changing trajectory planning for specific traffic scenarios. As depicted in Fig. 1, our scenario matching model takes the historical trajectory matrix of the traffic scenario described in Equation (3) as input, constructs it into a graph structure, performs feature extraction, and finally outputs the weight parameters of the cost function.

Inspired by [21], the spatial-temporal graph, donated as $G = \{V, E\}$, with both intra-vehicle and inter-frame connections are constructed to represent the traffic scenario. Each lane-changing vehicle in the dataset is considered as the central vehicle in turn, and the graph size varies with its longitudinal velocity. Each vehicle in the traffic scenario is regarded as a node in the graph, and the node set $V = \{V_{it} | i = 1, 2, \cdots, M; t = -T_h, \cdots, -1, 0\}$ contains all vehicle nodes in T_h history steps. The longitudinal position x and lateral position y are selected to be the node features.

The spatial edge set $E_s = \{e_{ij} | (v_i, v_j) \in D_{nbr}; i = 0, 1, \cdots, M\}$ represents the spatial neighboring relationships between vehicles in the graph. All vehicles, except for the autonomous ego vehicle, are connected to their neighboring vehicles whose Euclidean distance is less than a user-defined threshold. The autonomous ego vehicle is connected with the preceding and following vehicles in the current lane and the preceding and following vehicles in the target lane. The temporal edge set $E_t = \{e_{t,t+1} |, t = -T_h, \ldots, -1\}$ represents the natural temporal neighboring relationships between time step t and $t+1$.

The spatial and temporal relationships are represented by adjacency matrix $\boldsymbol{A}_s$ and $\boldsymbol{A}_t$, respectively. Additionally, a learnable spatial adjacency matrix $\boldsymbol{A}_{train}$ is proposed to better represent the spatial relationship. The combined spatial adjacency matrix is expressed as:

$$\boldsymbol{A}_{com} = \boldsymbol{\Lambda}^{-\frac{1}{2}}\boldsymbol{A}_s\boldsymbol{\Lambda}^{-\frac{1}{2}} + \boldsymbol{A}_{train} \tag{19}$$

where $\boldsymbol{\Lambda}^{-\frac{1}{2}}$ is used for normalization to ensure a reasonable aggregation of the current node features and neighboring features. $\boldsymbol{\Lambda}$ is a diagonal matrix, in which the element $\boldsymbol{\Lambda}^{ii}$ at row i and column i reflects the number of neighboring vehicles for the i-th vehicle node in the traffic scenario. It can be expressed as follows [21]:

$$\boldsymbol{\Lambda}^{ii} = \sum_j \boldsymbol{A}_s^{ij} \tag{20}$$

where $\boldsymbol{A}_s^{ij}$ is the element at row i and column j of the spatial adjacency matrix.

The graphic node features, denoted as $\boldsymbol{f}_{input} := \mathbb{R}^{M \times T_h \times C}$, and the combined spatial adjacency matrix, represented as $\boldsymbol{A}_{com} := \mathbb{R}^{M \times M}$, serve as inputs to our spatial-temporal graph convolutional network. Here, C symbolizes the count of node features. Initially, node features are normalized using batch normalization, and then, these normalized two-dimensional coordinates are transformed to a higher-dimensional space via a (1×1) kernel-sized convolution layer, which are represented as follows:

$$\boldsymbol{f}_{conv} = \mathrm{Conv2D}(\mathrm{BatchNorm}(\boldsymbol{f}_{input})) \tag{21}$$

The resulting convolutional feature $\boldsymbol{f}_{conv}$ is fed into the spatial graph convolutional network (S-GCN) to model the spatial interactions of vehicles in the same time steps, which can be implemented as:

$$\boldsymbol{f}_{sgra} = \boldsymbol{A}_{com}\boldsymbol{f}_{conv} \tag{22}$$

The graph feature $\boldsymbol{f}_{gra}$ is then sent into the temporal graph convolutional network (T-GCN) to extract the temporal interaction features for each vehicle along the temporal domain. Since the temporal adjacency matrix is an identity matrix, we utilize a convolutional network with a kernel size of (3×1) for feature extraction:

$$\boldsymbol{f}_{tgra} = \mathrm{Conv1D}(\boldsymbol{f}_{gra}) \tag{23}$$

In order to perform enough message-passing, our method alternates the S-GCN and T-GCN three times to fully perform spatial-temporal interaction, resulting in the spatial-temporal feature represented by $\boldsymbol{f}_{stgra} := \mathbb{R}^{M \times T_h \times C}$.

Finally, the weight generator is proposed to decode the spatial-temporal feature and output the weight vector. Since our task is to plan the trajectory for the autonomous ego vehicle, the ego feature $\boldsymbol{f}_{ego} := \mathbb{R}^{T_h \times C}$ containing time sequence information is sliced from $\boldsymbol{f}_{stgra}$ to obtain information for the ego vehicle. An LSTM decoder is proposed to decode the time sequence context and generate the weights in $\boldsymbol{\theta}_p$, which can be represented as:

$$[\theta_{r,1}, \theta_{r,2}, \theta_{r,3}, \theta_{r,4}] = \mathrm{LSTM}_{dec}(\boldsymbol{f}_{ego}) \tag{24}$$

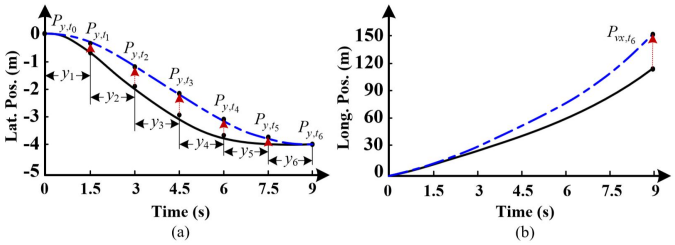

Fig. 2. Lang-changing trajectory representation based on the trajectory support points. (a) Lateral trajectory based on the lateral position support points. (b) Longitudinal trajectory based on the final longitudinal velocity support point. During the IRL optimization process, the support points transition from the initial trajectory (represented in black) to the optimal trajectory (represented in blue).

The loss function for training the network essentially calculates the proximity of the feature weight vector predicted by the network to ground truth in Θ_p, which is defined as:

$$loss = \sum_{i=1}^{4} (\theta_{r,i}^{pred} - \theta_{r,i}^{GT})^2 \tag{25}$$

where $\theta_{r,i}^{pred}$ is the i-th output of the network and $\theta_{r,i}^{GT}$ is the corresponding ground truth recovered from demonstration.

C. Trajectory Generation

Whether weight vectors are updated via the cost function recovery method or learned from the current traffic scenario through the scenario matching model, constructing and optimizing the cost function to generate the trajectory is critical, as illustrated in Equation (8). The inherent continuity of trajectories introduces an infinite-dimensional optimization challenge. To tackle this, we employ quintic splines to represent lateral motion trajectories and quartic polynomials for longitudinal motion, thereby optimizing a finite set of trajectory support points.

Analysis of our lane-changing demonstrated trajectories shows that the majority of vehicles can complete a full lane-changing maneuver within 9 seconds. As shown in Fig. 2, the lane-changing trajectory is defined in this paper with the starting moment of the lane-changing as the starting point on the time axis, and 7 discrete points $[t_0, t_1, ..., t_6]$ are selected at equal intervals of 1.5 seconds. The lateral positions of the start moment t_0 and the ending moment t_6 are assumed to be known in this paper. Therefore, the final chosen lateral set of points $\mathbf{\Omega}_{sp,y}$, consisting of 5 trajectory support points, is used as the interpolation points for the spline interpolation of the lane-changing lateral trajectory:

$$\mathbf{\Omega}_{sp,y} = [P_{y,t_1}, P_{y,t_2}, P_{y,t_3}, P_{y,t_4}, P_{y,t_5}] \tag{26}$$

For the longitudinal motion, the trajectory support point set is solely comprised of the final longitudinal velocity:

$$\mathbf{\Omega}_{sp,x} = [P_{vx,t_6}] \tag{27}$$

The boundary conditions for lateral motion are set with known position at the start and end moments, and zero velocity and acceleration. For longitudinal motion, the boundary conditions are set with known position, velocity, and acceleration at the initial moment, and zero acceleration at the end moment. Based on the principles of quintic spline interpolation [25] and quartic polynomial [26], the lateral and longitudinal lane-changing trajectory can be expressed as functions of the trajectory support points and time:

$$\xi_y = \xi_y(\mathbf{\Omega}_{sp,y}, t) \tag{28}$$

$$\xi_x = \xi_x(\mathbf{\Omega}_{sp,x}, t) \tag{29}$$

The optimal trajectory support point set $\mathbf{\Omega}_{sp,y}^*$ and $\mathbf{\Omega}_{sp,x}^*$ can be obtained through a finite-dimensional optimization problem transformed from (8), where the constraints remain the same:

$$\mathbf{\Omega}_{sp,y}^*, \mathbf{\Omega}_{sp,x}^* = \underset{\mathbf{\Omega}_{sp,x}, \mathbf{\Omega}_{sp,y}}{\arg\min} \; C_p(\boldsymbol{\theta}_{p,cur}, \boldsymbol{f}_\xi) \tag{30}$$

The solution can be obtained using the IPOPT solver [27]. Under an Intel Core i7 CPU, 16GB Memory, and an NVIDIA GeForce RTX1650 Ti Graphics Card environment, the average time of the weight vector generation process and the cost function optimization process are 15ms and 74ms, respectively. Given an initial value, the method may converge to a local optimum. However, through a strategic selection of initial values or by employing the simplify method proposed in [14], which transforms the non-convex solution space into a convex one, the possibility of being ensnared by local optima can be significantly reduced or even eliminated.

After constructing the trajectory expression with trajectory support points, in order to obtain the discrete trajectory, which can be followed by the control system, it's necessary to perform sampling at set intervals.

V. Experiment

Our lane-changing trajectory planning method is trained and tested using the well-known NGSIM dataset. We define the start of the lane-changing process as the moment when the absolute lateral velocity reaches 0.1m/s and extract 1112 lane-changing scenarios to form D_{lc}. We set the historical domain as 3 seconds prior to the start of the lane-changing procedure, and the planning domain as 9 seconds post-onset.

A. Cost Function Learning and Validation

This experiment is designed to evaluate the effectiveness of our proposed cost function recovery method.

We set parameters as follows: v_{des} = 30m/s, y_{tar} = 4m, $a_{x,max}$ = 2m/s^2, $a_{y,max}$ = 1m/s^2. The cost function recovery process of one lane-changing scenario within D_{lc} is depicted in Fig. 3. During the IRL process, the trained trajectory incrementally aligns with the demonstrated trajectory. The ultimate learned trajectory closely resembles the demonstrated trajectory in terms of position, velocity and acceleration. Moreover, as indicated in Fig. 4, both the difference between the expected and empirical feature values, donated as $||\nabla_{\boldsymbol{\theta}_p} \boldsymbol{J}(\boldsymbol{\theta}_p)||_2$ and the gradient of the features demonstrate a consistent convergence towards 0 as the learning process unfolds. This suggests that the proposed algorithm can guarantee the convergence of the learned trajectory towards the demonstrated trajectory, substantiating the effectiveness of the cost function recovery method.

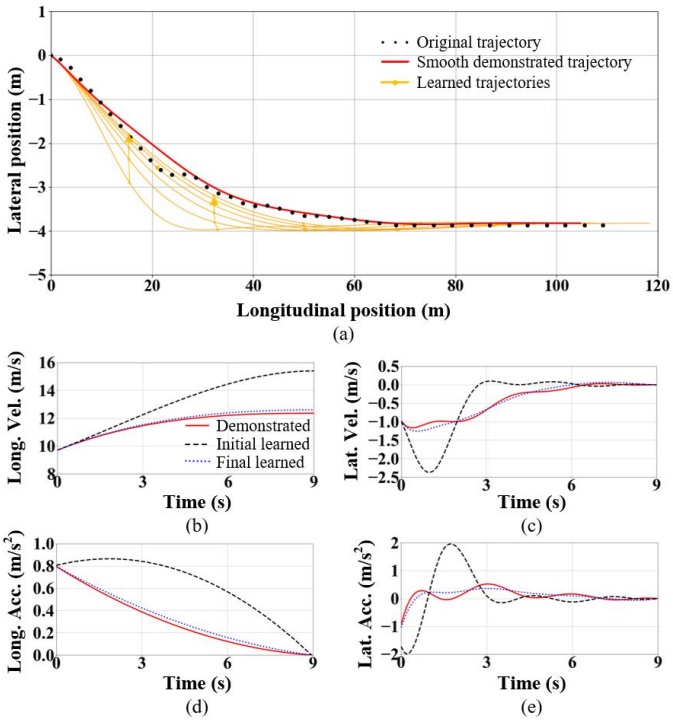

Fig. 3. Maximum entropy IRL-based demonstrated trajectory learning process.

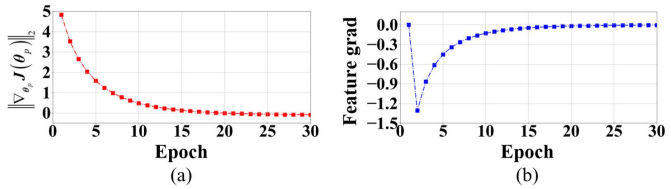

Fig. 4. Quantitative indicators of the learning process. (a) The difference between the empirical feature values and the expected feature values. (b) The gradient of some features.

B. Scenario Matching Model Training and Testing

This experiment is designed to support the claim that our proposed trajectory planning method is capable of automatically adjusting the cost function weights in various lane-changing scenarios.

In this subsection, our proposed scenario matching model is trained and tested utilizing the lane-changing scenario set D_{lc} as input and the obtained weight vector set Θ_p as ground truth. 70% of the lane-changing data are randomly sampled for training, 20% for validating, and the remaining 10% for testing. During training, the batch size is set to 16 and the learning rate is set to 0.001. The ADAM is exploited as the optimization method.

The existing solutions serving as comparisons to our proposed method are as follows.

- Intelligent Driver Model and Minimizing Overall Braking Induced by Lane Changes (IDM + MOBIL) [28]: A fundamental and widely applied model for simulating vehicle lane-changing behaviors, with parameters configured according to the cited literature.
- Expectation Maximization Planner (EM Planner) [14]: A key trajectory planning module developed by Baidu for the Apollo system, which is widely used in mass-produced vehicles.
- General IRL: A method that plans lane-changing trajectories by optimizing a cost function constructed from the average values of the cost function weights of all demonstrated trajectories.
- LSTM-based Model (LM): An LSTM-based model [29] is used as the scenario matching model. Planned trajectories are then generated by optimizing the cost function constructed using the weights generated by this model.
- TrafficPredict-based Model (TM): The structure of TrafficPredict [30], a well-recognized trajectory prediction model, is adopted to extract features of traffic scenarios and generate the cost function weights.

The trajectory feature error vector describing the consistency between the planned trajectory and the demonstrated trajectory is employed to evaluate these method:

$$\boldsymbol{T}_{fe} = [T_{f_{ax}}, T_{f_{ev_x}}, T_{f_{ay}}, T_{f_{ey}}, T_{f_{risk}}] \tag{31}$$

where $T_{f_{ax}}$ is the longitudinal comfort feature error, which is defined as:

$$T_{f_{ax}} = \frac{1}{N}\sum_{i=0}^{N} ||f_{ax} - \tilde{f}_{ax}|| \tag{32}$$

where N represents the total number of lane-changing scenarios in the testing set, and f_{ax} and $\tilde{f}_{ax}$ represent the longitudinal comfort feature of the planned trajectory and the demonstrated trajectory, respectively. Similarly, $T_{f_{ev_x}}, T_{f_{ay}}, T_{f_{ey}}$ and $T_{f_{risk}}$ measure the differences in terms of longitudinal efficiency, lateral comfort, lateral efficiency and risk, respectively.

The feature errors are calculated and shown in Table I. The IDM+MOBIL and EM planners show significant deviations from the demonstrated trajectories across all metrics, surpassing the error magnitude of General IRL. We speculate that this is due to the preset planning strategies exhibiting an overall bias when compared to the average human driving habits, typically leaning towards a more conservative, safety-oriented approach. Both LM and TM are LSTM-based feature weight prediction models, exhibiting similar feature error metrics. Our model generates trajectories that are closer to the demonstrated ones in terms of longitudinal comfort, longitudinal efficiency, lateral efficiency, and risk than the TM planner, with improvements of 4.5%, 6.7%, 1.8%, and 5.8%, respectively. Regarding the lateral comfort feature, our method shows slightly larger errors compared to LM and TM, but the difference is relatively small. This might be because our algorithm is more effective in extracting longitudinal features compared to lateral features. Moreover, when considering the average error across all features, our method still demonstrates superiority. Additionally, the variance of LM and our proposed model on all feature errors are 0.230 and 0.223 respectively. This shows that prediction errors are relatively concentrated around the average error, indicating that our proposed model can provide relatively stable prediction results in most cases.

TABLE I
FEATURE ERRORS BETWEEN PLANNING RESULTS AND DEMONSTRATIONS IN THE TESTING SET

Method	Feature Errors				
	$T_{f_{ax}}$	$T_{f_{ev_x}}$	$T_{f_{ay}}$	$T_{f_{ey}}$	$T_{f_{risk}}$
IDM+MOBIL	0.349	0.423	1.556	0.426	0.109
EM Planner	0.240	0.337	1.050	0.333	0.104
General IRL	0.226	0.325	0.963	0.330	0.087
LM	0.180	0.260	**0.888**	0.272	0.085
TM	0.179	0.255	0.890	0.273	0.086
Proposed	**0.171**	**0.238**	0.894	**0.268**	**0.081**

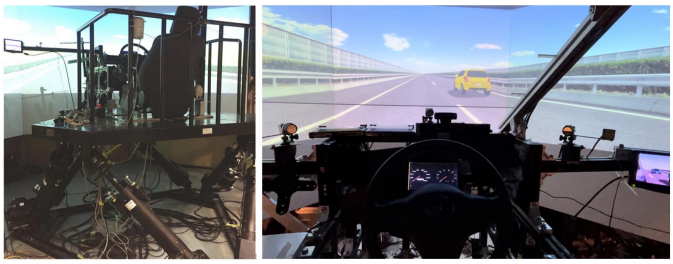

Fig. 5. High-fidelity autonomous driving platform.

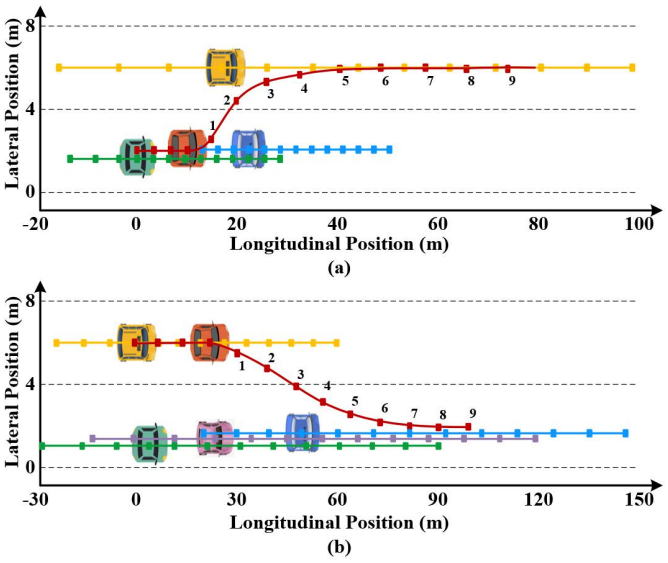

Fig. 6. Specific performance in different lane-changing scenarios. The black dotted lines represent the lane lines. The red vehicle is the autonomous ego vehicle, and the vehicles in other colors represent the surrounding vehicles. The trajectory of each vehicle is plotted for 3 seconds before and 9 seconds after the start of the lane-changing process, with each second divided by solid points. The trajectory of the autonomous ego vehicle in red is the tracking results of the planned trajectory of our scenario-adaptive trajectory planning method, with numbers indicating the planning moment.

C. Experiment and Discussion

This experiment aims to verify the executability of the results from our proposed trajectory planning method.

Two representative lane-changing scenarios are distilled from the testing set and experiments are carried out on a high-fidelity autonomous driving platform (Mitsubishi Precision Co., Ltd., Japan), as depicted in Fig. 5. The traffic scenarios are designed in Mitsubishi Power Carsim-based software, while the driving platform hardware encompasses an electric steering and braking system. Once a lane-changing intent is detected, our scenario matching model inputs 3-second historical trajectories of all vehicles and outputs the learned weight vector. Subsequently, the cost function is structured and optimized to yield the trajectory planning results, which are implemented by the actuators.

Fig. 6 illustrates the recorded scenarios and the tracking results of planned trajectories. In scenario (a), the autonomous red vehicle completes a lane-changing process in 5 seconds, motivated by a preceding vehicle obstructing its path in the current lane. According to human driving habits, the vehicle should accelerate to quickly merge into the target fast lane and swiftly exit its current lane for longitudinal acceleration. The weight vector calculated by our algorithm is [1, 0.738, 0.576, 0.95, 0.996], suggesting a stronger emphasis on driving efficiency and a reduced focus on comfort. This indicates that the planned trajectory, which includes decisive acceleration in both longitudinal and lateral directions, aligns with human driving habits. Scenario (b) presents a more complex situation, requiring 7 seconds to complete a lane-changing process. With three vehicles in the target lane, the ego vehicle carefully maneuvers to yield to the leading blue and purple vehicles and slots in before the trailing green vehicle. The computed

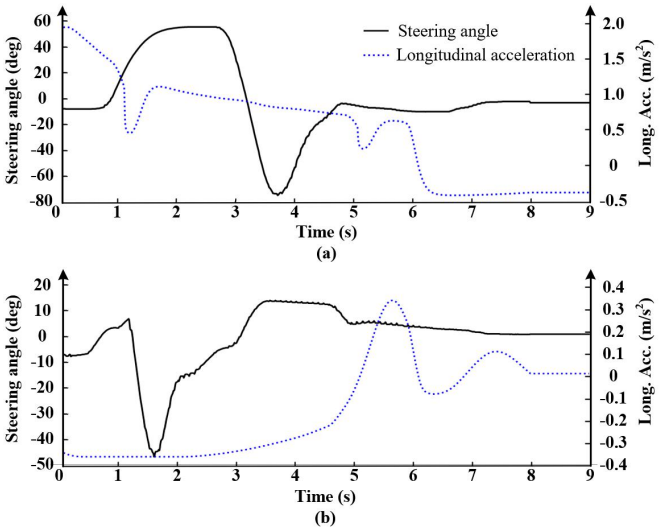

Fig. 7. Detailed execution results of steering angle and longitudinal acceleration of the autonomous ego vehicle in scenario (a) and (b).

weight vector [1, 0.476, 0.765, 0.963, 0.995] indicates a balance between driving efficiency and comfort, reflecting a mild acceleration that aligns with human driving preferences.

Fig. 7 presents the precise execution of steering wheel angle and longitudinal acceleration in the two test scenarios. The consistent, reasonable values suggest that our planned trajectories can be enacted by actuators. Notably, scenario (a) displays a larger average steering wheel angle and more prominent acceleration during lane changes, which aligns with the weight vector's implications in the cost function synthesized by our planning algorithm.

In summary, the above simulation and experimental results

prove that our proposed trajectory planning algorithm can effectively adjust the weight vector of cost function according to the traffic scenario, realizing the scenario-adaptive lane-changing trajectory planning for autonomous driving.

VI. Conclusions

This paper proposes the graph-based scenario-adaptive lane-changing trajectory planning method for autonomous driving based on a combination of the IRL-based cost function recovery method and the ST-GCN-based scenario-matching model. Our method innovatively integrates optimization-based and learning-based trajectory planning methods to enable seamless adaptation to highly dynamic traffic scenarios. The effectiveness and advantages of the proposed method are verified by the simulations on the NGSIM dataset and experiments on two distinct lane-changing scenarios using a high-fidelity autonomous driving platform. Our results demonstrate that the proposed planning framework can adjust the cost function feature weights to better balance various planning objectives. In future work, we will strive to extend our approach to encompass other types of autonomous navigation behaviors.

References

[1] J. Song, G. Tao, Z. Zang, H. Dong, B. Wang, and J. Gong, "Isolating trajectory tracking from motion control: A model predictive control and robust control framework for unmanned ground vehicles," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1699–1706, 2023.

[2] S. Cheng, Z. Wang, B. Yang, L. Li, and K. Nakano, "Quantitative evaluation methodology for chassis-domain dynamics performance of automated vehicles," *IEEE Transactions on Cybernetics*, 2022.

[3] X. Zhang and H. Duan, "An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning," *Applied Soft Computing*, vol. 26, pp. 270–284, 2015.

[4] S. Cheng, Z. Wang, B. Yang, and K. Nakano, "Convolutional neural network-based lane-change strategy via motion image representation for automated and connected vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[5] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[6] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[7] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on control systems technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[8] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles," *IEEE transactions on automation science and engineering*, vol. 7, no. 1, pp. 167–172, 2009.

[9] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K.-M. Dahlén, "On path planning methods for automotive collision avoidance," in *2013 IEEE intelligent vehicles symposium (IV)*. IEEE, 2013, pp. 931–937.

[10] Y. Liu, H. Wang, J. Fan, J. Wu, and T. Wu, "Control-oriented uav highly feasible trajectory planning: A deep learning method," *Aerospace Science and Technology*, vol. 110, p. 106435, 2021.

[11] P. Geiger and C.-N. Straehle, "Learning game-theoretic models of multiagent trajectories using implicit layers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 6, 2021, pp. 4950–4958.

[12] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[13] J. L. V. Espinoza, A. Liniger, W. Schwarting, D. Rus, and L. Van Gool, "Deep interactive motion prediction and planning: Playing games with motion prediction models," in *Learning for Dynamics and Control Conference*. PMLR, 2022, pp. 1006–1019.

[14] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, "Baidu apollo em motion planner," *arXiv preprint arXiv:1807.08048*, 2018.

[15] Y. Liu, B. Zhou, X. Wang, L. Li, S. Cheng, Z. Chen, G. Li, and L. Zhang, "Dynamic lane-changing trajectory planning for autonomous vehicles based on discrete global trajectory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8513–8527, 2021.

[16] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[17] Z. Huang, J. Wu, and C. Lv, "Driving behavior modeling using naturalistic human driving data with inverse reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10 239–10 251, 2021.

[18] D. Silver, J. A. Bagnell, and A. Stentz, "Learning autonomous driving styles and maneuvers from expert demonstration," in *Experimental Robotics*. Springer, 2013, pp. 371–386.

[19] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2641–2646.

[20] Q. Dong, T. Jiang, T. Xu, and Y. Liu, "Graph-based planning-informed trajectory prediction for autonomous driving," in *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*. IEEE, 2022, pp. 1–6.

[21] A. Mohamed, K. Qian, M. Elhoseiny, and C. Claudel, "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 424–14 432.

[22] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction," *Neurocomputing*, vol. 445, pp. 298–308, 2021.

[23] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph neural networks for decentralized multi-robot path planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 785–11 792.

[24] C. Yu and S. Gao, "Reducing collision checking for sampling-based motion planning using graph neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4274–4289, 2021.

[25] K. Erkorkmaz and Y. Altintas, "High speed cnc system design. part i: jerk limited trajectory generation and quintic spline interpolation," *International Journal of machine tools and manufacture*, vol. 41, no. 9, pp. 1323–1345, 2001.

[26] M. Yue, X. Wu, L. Guo, and J. Gao, "Quintic polynomial-based obstacle avoidance trajectory planning and tracking control framework for tractor-trailer system," *International Journal of Control, Automation and Systems*, vol. 17, no. 10, pp. 2634–2646, 2019.

[27] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

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

[29] X. Mo, Y. Xing, and C. Lv, "Interaction-aware trajectory prediction of connected vehicles using cnn-lstm networks," in *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2020, pp. 5057–5062.

[30] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 6120–6127.