GIN: Graph-based Interaction-aware Constraint Policy Optimization for Autonomous Driving

Se-Wook Yoo[1], Chan Kim[1], Jin-Woo Choi[1], Seong-Woo Kim[1], and Seung-Woo Seo[1]

***Abstract*— Applying reinforcement learning to autonomous driving entails particular challenges, primarily due to dynamically changing traffic flows. To address such challenges, it is necessary to quickly determine response strategies to the changing intentions of surrounding vehicles. This paper proposes a new policy optimization method for safe driving using graph-based interaction-aware constraints. In this framework, the motion prediction and control modules are trained simultaneously while sharing a latent representation that contains a social context. To reflect social interactions, we illustrate the movements of agents in graph form and filter the features with the graph convolution networks. This helps preserve the spatiotemporal locality of adjacent nodes. Furthermore, we create feedback loops to combine these two modules effectively. As a result, this approach encourages the learned controller to be safe from dynamic risks and renders the motion prediction robust to abnormal movements. In the experiment, we set up a navigation scenario comprising various situations with CARLA, an urban driving simulator. The experiments show state-of-the-art performance in navigation strategy and motion prediction compared to the baselines. The code is available online[1].**

I. INTRODUCTION

In safety-critical systems such as autonomous driving, the autonomous vehicle must avoid dynamic obstacles without collision while following the lane. Recently, deep reinforcement learning (DRL) [1] have shown general driving strategies by maximizing the reward for following the lane while providing a negative penalty for a collision. Nevertheless, this does not guarantee that the safety constraints are observed. To overcome this issue, a method based on safe reinforcement learning (RL) [2] is proposed to approximate the expected long-term costs as constraints for policy optimization. Although this method tends to maintain minimum constraints for the collision risks over static objects, it fails for dynamic objects. This is because it is difficult to infer the randomness of the cost distribution by providing sparse cost signals only when collisions occur. In urban driving, it is necessary to prevent potential risks by generating cost signals while considering the predicted motions between dynamic agents.

However, it is particularly difficult to predict motions in congested situations because the intentions of dynamic objects are not directly observable. To solve this problem, previous works [3]–[5] used expensive high-definition (HD) maps as prior knowledge to predict various motion patterns over different types of roads, such as multi-lane roads, multi-way intersections, or roundabouts. Nevertheless, These methods are not generally applied to changes in specific road context information such as traffic signs and road marks. Moreover, the accuracy of abnormal patterns decreases in case some agents do not conform with traffic rules. To address this issue, [6] attempts to predict trajectories according to perceived maneuvers, such as lane changing, braking, or turning. Unfortunately, incorrect classification of maneuver types also leads to inaccurate predictions. To show robust prediction, [7], [8] reflect the interactions of social movements by learning latent variables that encode sequential information over objects. Although [7] shows high accuracy through social pooling, it incurs a high computational cost for multiple objects. Unlike this, [8] expresses dynamic agents as graph form, and the spatial importance is considered in a graph convolution network (GCN) [9]. This shows faster and more accurate results for multiple objects. However, prediction through GCN shows non-universal results for new graph structures that have not been seen in pre-collected data. Although we use GCN structure to understand the interaction for motion prediction, arbitrary graphs are dealt with using the data being collected by a learning RL agent.

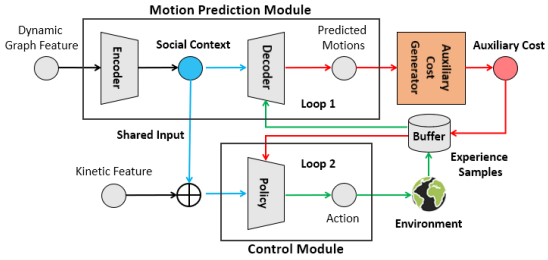


Fig. 1. Proposed scheme. The blue line indicates the social context vector shared by the modules for motion prediction and control, while the combination of colored lines illustrates feedback loops between two modules. According to the red line, auxiliary costs are additional constraint signals for control. Moreover, the green line illustrates that the saved samples serve as ground truth for motion prediction.

In this paper, we present a new framework that understands the interaction of surrounding vehicles and makes appropriate decisions according to the detected risk. Fig. 1 shows the overview of our scheme, which combines the modules for motion prediction and control to learn tactical driving strategies. Specifically, the two modules are simultaneously trained through the RL framework while sharing a social context vector that explains the interactions between agents. To learn the context vector, we spatiotemporally refine the

[1]Seoul National University, Seoul, Republic of Korea {tpdnr1360, chan_kim, wlsdn9530, snwoo, sseo}@snu.ac.kr

This research was supported by the Challengeable Future Defense Technology Research and Development Program through the Agency for Defense Development(ADD), funded by the Defense Acquisition Program Administration(DAPA) in 2022(No.915034201) and in part by the Automation and Systems Research Institute, Seoul National University.

[1]https://github.com/Usaywook/gin.git

dynamic graph features using an encoder. Using this context vector, we create two feedback loops to combine the two modules effectively. The first one generates auxiliary costs by geometrically interpreting the predicted motions, which are used as constraints for policy optimization. The other one utilizes samples collected by the policy during training as ground truth for the prediction.

The main contributions of the proposed method are as follows:

- We propose a new safe RL framework with a policy constraining interaction-aware risks by geometrically interpreting the predicted trajectories.
- We represent the movements of vehicles as a dynamic graph and train it to learn social interaction directly without prior knowledge.
- We integrate the path prediction and control modules within a single RL framework by allowing the two modules to share spatiotemporally compressed social context for robust prediction and response to dynamic risks.

The remainder of this paper is organized as follows. Related works are reviewed in Section II. The background is explained in Section III. The proposed methods are presented in Section IV. Our experimental results are presented in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

In this section, we discuss two related approaches, namely, safe RL and interaction-aware motion planning.

A. Safe Reinforcement Learning

In RL literature, the concept of safety is unavoidable in the safety-critical domain. Generally, the constraint Markov decision process (CMDP) [10] is a natural approach for finding a solution among set of allowable policies that guarantee safety. [2] approximates the average cumulative costs under the CMDP using a policy gradient (PG) algorithm based on DRL. To adjust the trade-off caused by the different scales between the reward and cost, [11] proposed an alternative penalty in which the designed cost function was multiplied by the learned adaptive weight. However, these methods do not account for the potential risk of the constraint threshold being exceeded in a heavy-tailed cost distribution. To address this problem, [12] replaces average cumulative costs with conditional value-at-risk (CVaR) [13]. This minimizes the heavy-tailed risk by predicting the full distribution. [14] derives the PG formula using the CVaR measure to show behaviors according to the risk level. Nevertheless, it has only been applied to a few scenarios static obstacles exist for a short period.

In contrast, we focus on navigation problems with complex interactions between dynamic agents over long periods. Furthermore, we adopt a worst-case soft actor-critic (WCSAC) [15] method that interprets the CVaR criterion in terms of alternative penalties. This reduces the approximated error over the cost distribution by balancing the adaptive weights for entropy and safety. However, this method has several limitations. For example, if a conservative risk level is used, it becomes overly pessimistic. Moreover, if the cost signal is sparse, the variance of the distribution decreases. In this case, the CVaR criterion is close to the mean value. This still has limitations, especially when rare events pose a great risk. To overcome the problems, we create dense cost signals by combining the interaction-aware motion prediction process with the WCSAC policy optimization method.

B. Interaction-aware Motion Planning

Recently, several studies [16]–[18] proposed integrating RL and planning to learn driving skills. These works predict the trajectory of surrounding vehicles and then plan the trajectory of the ego-vehicle to show reactive behaviors but do not reflect the interaction during planning. Some methods predict model approximated by a partially observable Markov decision process (POMDP) [19] to obtain a tactical driving strategy. For example, [20] predicts a temporally correlated region using a search tree created through a Monte Carlo tree search (MCTS). Meanwhile, [21] receives temporal sequences of occupancy grid maps (OGMs) for implicit intention reasoning and [22] predicts the time horizon for motion planning. Besides, there have been approaches to encourage cooperative and competitive behaviors in a multi-agent system. [23] approximates the sum of the influence of other agents and their intent as a linear function. [24] uses a GCN-based Q-network that shares the observations of all agents. However, it is costly to use sophisticated inputs such as OGMs, HD maps, or shared observations. On the contrary, interaction-aware prediction methods do not require expensive resources. Although some methods using social pooling [7] have shown high accuracy, these are practically intractable for predicting multiple objects. Meanwhile, in [25], road contexts, such as lane and dynamic agents, were expressed as graphs for intuitive interpretation. In this method, after filtering graph features through GCN, the learned features capture high-dimensional global graph relationships that maintain semantic and spatial locality. Similarly, in [8], the surrounding agents are expressed in a graph form. This makes the prediction for multiple objects accurate and rapid. Hence, we adopt the structure of this method to reflect interactions that maintain spatiotemporal locality.

Unlike previous studies, in which interaction was reflected only in motion prediction, we consider the interaction to create constraints for control. To combine motion prediction and policy models efficiently, we learn spatiotemporal compressed representation, which both models share as input. In addition, we create a meaningful connection between the prediction model and policy through the auxiliary costs generated by the geometric analysis of the predicted motions. Eventually, we find an optimal policy constrained by the predicted interaction-aware long-term costs for safety.

III. PRELIMINARIES

A. Constrained Markov Decision Processes (CMDP)

The CMDP is an extension of the MDP, is utilized for safe RL. The function $\mathcal{C} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^+$ of costs c is added to a tuple of MDP $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is a set of states s, $\mathcal{A}$ is a set of actions a, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a function of rewards r, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a state transition probability distribution, $\gamma \in (0, 1)$ is a discount factor, and $\pi : \mathcal{S} \times \mathcal{A} \to$ [0,1] is a policy distribution. The objective of safe RL is to find an optimal policy in a set of allowable policies that satisfy the constraints. This maximizes the expected return $\mathbb{E}_\pi[\sum_t^\infty \gamma^t r(s_t, a_t)]$, denoted as $Q_\pi^r(s, a)$, such that the expected long-term cost remains below given threshold d, which is formulated in [2] as follows:

$$\max_\pi Q_\pi^r(s, a) \; s.t. \; \mathbb{E}_\pi \left[\sum_t^\infty \gamma^t c(s_t, a_t)\right] \leq d. \tag{1}$$

B. Worst-Case SAC (WCSAC)

After approximating the expected long-term cost as the mean $Q_\pi^c(s, a)$ of the Gaussian distribution, constraining the average to be less than the threshold d does not consider the potential risk induced by the heavy-tail of the cost distribution $p_\pi(c|s, a)$. To address this issue, the worst-case measure CVaR_α is used in WCSAC. CVaR_α is the average in the area where the CDF of $p_\pi(c|s, a)$ is greater than a risk level $\alpha \in (0, 1)$. By taking into account the variance $V_\pi^c(s, a)$ of the Gaussian distribution, WCSAC replaces the existing measure $Q_\pi^c(s, a)$ with a closed-form estimation for CVaR_α, which is derived in [26]. As the estimated value for CVaR_α, the risk-sensitive criterion Γ_π^α for α is calculated in [15] as follows:

$$\Gamma_\pi^\alpha \doteq \text{CVaR}_\alpha = Q_\pi^c(s, a) + \alpha^{-1}\phi(\Phi(\alpha))\sqrt{V_\pi^c(s, a)}, \tag{2}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are the PDF and CDF of the standard normal distribution, respectively. The objective of WCSAC is to encourage safe exploration by auto-tuning adaptive entropy β and safety weight κ under the problem of maximum entropy RL [27] with safety constraints. For all times t and given risk level α, the objective is minimized in [15] as follows:

$$\mathbb{E}_\pi[\beta \log \pi(a_t|s_t) - X_{\alpha,\kappa}^\pi(s_t, a_t)] \; s.t. \; \Gamma_\pi^\alpha(s_t, a_t) \leq d, \tag{3}$$

where $X_{\alpha,\kappa}^\pi(s, a) = Q_\pi^r(s, a) - \kappa\Gamma_\pi^\alpha(s, a)$. This implies entropy regularized Q-value is maximized, while risk-term is minimized.

C. Graph Convolution Network (GCN)

As a variant of a convolutional neural network (CNN) [28], GCN reflects flexible adjacency to handle abstract concepts such as social interaction. This method extracts valuable information from an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The N node features $\mathcal{V} \in \mathbb{R}^{N \times D_v}$ with depth D_v is filtered by aggregating information from adjacent nodes along the edges $\mathcal{E}$ using the following propagation rule [9]:

$$\psi(A, H^l; W^l) = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^l W^l), \tag{4}$$

where an adjacency matrix with added self-connections is denoted by $\tilde{A} = A + I_N \in \mathbb{R}^{N \times N}$, the degree matrix of $\tilde{A}$ is $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, W^l is a layer-specific trainable weight, $\sigma(\cdot)$ denotes an activation function, and $H^l \in \mathbb{R}^{N \times D_h}$ is the embedding vector in the l^{th} layer except for $H^0 = X$.

IV. METHOD

We introduce a framework that predicts interaction-aware motion and controls an ego-vehicle to respond to dynamically changing driving styles of the surrounding vehicles. Fig. 2 shows the detailed structure of each part. All hyperparameters are tuned through coarse grid search. This section describes how these three modules are used: input preprocessing, interaction-aware motion prediction, and interaction-aware control.

A. Input Preprocessing

We use two input features to perform navigation to the destination. One is a path tracking feature F_k. This is required to track the given global path. The second is the social graph feature F_d, which accounts for the dynamic environment.

1) Path Tracking Feature: The path tracking feature F_k is created by processing module ϕ_k using a measurement vector m_t and global path τ_G. m_t comprises position q_t, heading angle θ_t, and speed v_t of ego-vehicle at the current time t. τ_G that follows road network graph $\mathcal{G}_n$ is generated in advance from start p_s to the goal p_g location using the A* algorithm [29]. As shown in Fig. 2, $F_k \in \mathbb{R}^5$ consists of $e_c^0, e_h^0, e_c^d, e_h^d$, and v_t. Specifically, e_c^0 and e_h^0 are the cross tracking error and the heading angle error at t, respectively. e_c^d and e_h^d are the minimum distance and angular difference, respectively, which is calculated at the point moved by the lookahead distance d along τ_G, assuming that the ego-vehicle at q_t moves while maintaining θ_t and v_t.

2) Social Graph Feature: We represent the movement of dynamic agents on the road in graph form F_d by using processing module ϕ_d. F_d consists of a node matrix $\mathcal{V} \in \mathbb{R}^{N \times T_h \times D_v}$ and a set of adjacency matrices $\mathbf{A} = \{A_k \in \mathbb{R}^{N \times N} | \; k = 1, \ldots, L\}$, where N is the number of surrounding vehicles, T_h is the historical length, D_v is the depth for each node, and L is the maximum number of hops. The i^{th} row and t^{th} column vector $v_{it} \in \mathbb{R}^{D_v}$ of $\mathcal{V}$ comprises the position and heading angle in the egocentric coordinate system, relative velocity, relative acceleration, relative angular velocity, and mask. A mask indicates whether an object exists within the recognition range. To create the adjacency matrix $A \in \mathbb{R}^{N \times N}$ shown in Fig. 3, we use two types of edges: inter-frame edges $E_F = \{(v_{it}, v_{i(t+1)}) |$ for $i = 1, \ldots, N$, and $t = 1, \ldots, T_h - 1\}$ and spatial edges $E_S = \{e_{ij} = (v_{it}, v_{jt}) |$ if $d_{ij} \leq D_{close}$ for $i, j = 1, \ldots, N$, and $t = T_h\}$, where d_{ij} is the Euclidean distance between the last vertices along inter-frame edges E_F of i^{th} and j^{th} vehicles, and D_{close} is the distance threshold for considering neighbors. Subsequently, the i, j element of A is obtained using the indicator function

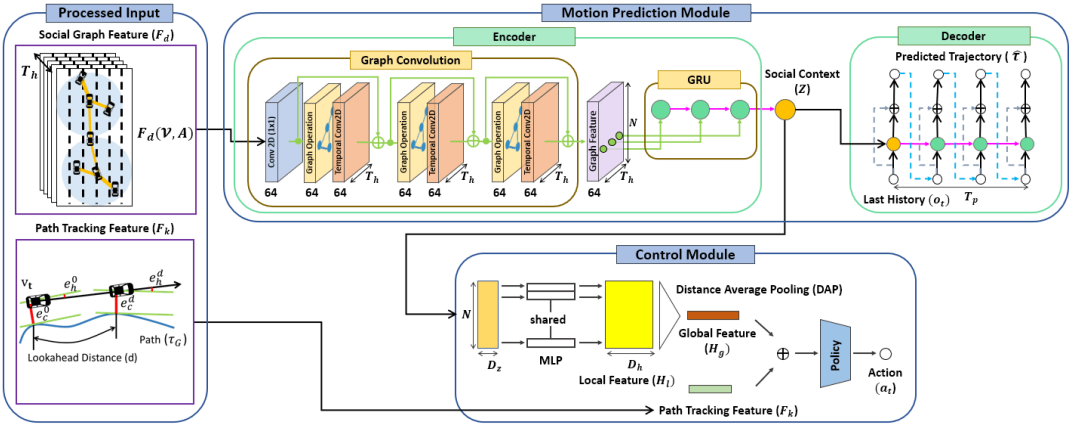


Fig. 2. Overall architecture of proposed framework. Motion prediction is performed by variational autoencoding models while creating intermediate representation that explains the social context. The control module receives input features for tracking a path and understanding social context. Then, it outputs an action that considers social interaction.

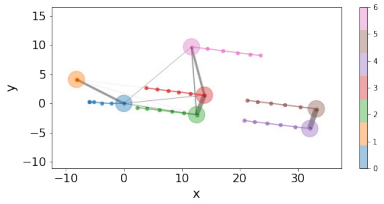


Fig. 3. Visualization of the social graph feature. The value of each color indicates the index of surrounding vehicles. The 0^{th} index belongs to the ego-vehicle located at origin. E_F with the chromatic color depicts the progress of movement. E_S expressed using gray colors indicates proximity, where the width represents the size of the weight.

$\mathbb{1}_{\{\exists e_{ij} \in E_S\}}$. To learn the high-order relation, the i, j element of A_k is obtained as follows:

$$[A_k]_{i,j} = a_{kij} e^{-\frac{d_{ij}}{K}}, \text{ for } i, j = 1, \ldots, N, \tag{5}$$

where a_{kij} is an element of A to the k^{th} power, and the weights of the edges are modeled as a Boltzmann function with the temperature constant K. To consider appropriate interaction, we use $N = 20$, $T_h = 24$, $L = 3$, $D_{close} = 15$ and $K = 5$.

B. Interaction-aware Motion Prediction

Trajectories are predicted by considering the interactions between ego-vehicle and other vehicles. We utilize F_d as the input because it concisely represents dynamic agents instead of costly information such as OGM or HD maps. The structure of the graph dramatically changes as the agents enter or leave the recognition range. To enable the prediction module to reason the arbitrary graph inductively, the module is trained with sub-graph samples $\breve{F}_d$ collected by the RL agent. The module for motion prediction consists of an encoder ν and decoder ρ, parameterized by η and ζ, respectively.

1) Encoder: Sampled from the replay buffer $\mathcal{D}$, $\breve{F}_d$ is used as input to train encoder ν_η and decoder ρ_ζ. As shown in the encoder part of Fig. 2, a convolutional layer with a 1 by 1 kernel is applied to increase the size of the node features. Subsequently, the features pass through three layers of the graph convolutional block to extract high-order relations that encourage sophisticated driving strategies. Each graph convolutional block compacts the latent representation while maintaining spatiotemporal connectivity. In this procedure, the sum of the graph operation $\sum_{k=1}^{L} \psi(A_k, H^l; W_k^l)$ in Eq. 4 extracts spatially correlated features along the spatial edges E_S. Next, a convolution operation is applied along the inter-frame edges E_F to understand the temporal dependencies between the dynamic agents, which helps the learned intentions to remain consistent. Finally, a residual connection is utilized to prevent the gradient from vanishing. After multilayer blocks, to create a social context $Z \in \mathbb{R}^{N \times D_z} = (z_1, \ldots, z_N)$, the sequential features are summarized by the GRU model.

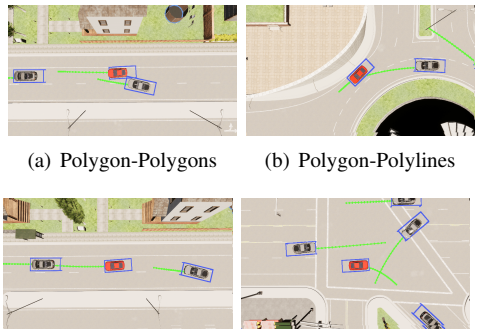
2) Decoder: Based on the implied context Z, the decoder ρ_ζ generates the future trajectory $\hat{\tau}_t$ of the surrounding vehicles with prediction time horizon T_p, then allows the control module to know long-term risks using $\hat{\tau}_t$. This is because it can provide signals for detected risks by geometrically interpreting the predicted trajectories. The initial hidden state of the decoder GRU is the social context, and the initial input is the current location o_t of each vehicle. Each GRU cell predicts the velocity profiles of vehicle through a skip connection. The prediction module is trained using regression loss with p-norm, which is as follows:

$$\mathcal{L}_P = \frac{1}{T_p} \sum_{t=1}^{T_p} ||\rho_\zeta(\nu_\eta(\breve{F}_d), o_t) - \tau_t||_p, \tag{6}$$

where τ_t is the ground truth future trajectory.

C. Interaction-aware Control

Our control module outputs actions to track a path while avoiding a collision in dynamic environment via the RL framework. For this reason, our method uses the path tracking feature F_k and social context Z, represents the interaction of objects. To compress the social context in the view of the control of an ego-vehicle, we design a pooling module. Moreover, auxiliary costs are generated using the predicted motions. Finally, we explain how to optimize the modules for motion prediction and control simultaneously through the off-policy PG method under the CMDP.



(a) Polygon-Polygons (b) Polygon-Polylines

(c) Polyline-Polygons (d) Polyline-Polylines

Fig. 4. Four cases of social risk. The ego-vehicle is red, and the others are gray. Boundaries with safety margins are expressed by blue bounding boxes. The green polylines represent the predicted future trajectories.

1) Distance Average Pooling: Because the order of the nodes changes when creating a graph within the recognition range, we conserve the permutational invariance of the social context Z. For each element of Z, the local features $H_l = (h_1, \ldots, h_N) \in \mathbb{R}^{N \times D_h}$ are obtained by a multi-layer perceptron (MLP) that shares weights. Then, distance average pooling (DAP) is applied to create a global feature $H_g \in \mathbb{R}^{D_h}$. Since the closer the agents are to the ego-vehicle, the more the intentions should be reflected, DAP is designed as follows:

$$H_g = \frac{1}{N} \sum_{i=1}^{N} h_i e^{\frac{-\sqrt{x_i^2+y_i^2}}{K}}, \tag{7}$$

where K regulates the influence according to the distance, (x_i, y_i) is the relative position of the i^{th} vehicle in the egocentric coordinate system, and $h_i \in \mathbb{R}^{D_h}$ is the i^{th} row vector of H_l. After DAP, we concatenate the global feature H_g and hand-designed path tracking feature F_k, and pass $[H_g, F_k]$ as input to the policy π, parameterized by θ.

2) Auxiliary Cost: We create auxiliary costs by using the predicted trajectories to make the sparse cost signals dense, which helps the learned policy π_θ to avoid long-term risks. We divide the risk detection into four cases, as shown in Fig 4. To interpret the social risk geometrically, we used polygons and polylines. The polygon is a local bounding box with safety margin (ϵ_l, ϵ_w) added to the length and width of each vehicle. The polyline represents the trajectory of each vehicle as predicted by the decoder ρ_ζ. For the case Fig. 4(a), we check the overlap between the polygon of the ego-vehicle and others using the separating axis theorem (SAT) [30]. In the cases shown in Fig. 4(b-d), we utilize a method [31] to check the intersection between two-line segments based on a counter-clockwise (CCW) algorithm. To create line segments, we connect the starting and ending points of the polygons. For the polygons, we use each side as a line segment. The auxiliary cost is defined as follows:

$$\hat{c}_\rho = \mathbb{1}_{\{\exists G \cap \mathbf{G}\}} \vee \mathbb{1}_{\{\exists G \cap \mathbf{L}\}} \vee \mathbb{1}_{\{\exists L \cap \mathbf{G}\}} \vee \mathbb{1}_{\{\exists L \cap \mathbf{L}\}}, \tag{8}$$

where G and L are the polygon and polyline, respectively, of the ego-vehicle. The bold fonts such as $\mathbf{G}$ and $\mathbf{L}$ are those of the other vehicles. $\cap$ represents a set of intersections between ego-ones and others, and $\vee$ is an OR operator.

(a) Town 03 (b) Navigation

Fig. 5. Visualization of navigation environment: **(a)** is the map used in the virtual environment and **(b)** explains an episode. All possible spawn points are represented by navy dots. The start and end points are red and blue, respectively. The navigation route is indicated in green.

Algorithm 1 GIN

1: **Initialize**: Parameters of networks $\pi_\theta, Q_\psi, C_\mu, \nu_\eta, \rho_\zeta$
2: **Initialize**: Adaptable weights β, κ
3: **Initialize**: Replay buffer $\mathcal{D} \leftarrow \emptyset$
4: Copy target networks $< \bar{\theta}, \bar{\psi}, \bar{\mu} >$ with $< \theta, \psi, \mu >$
5: Load road network graph $\mathcal{G}_n$ from database
6: **for** each iteration **do**
7: Randomly select p_s and p_g
8: $\tau_g \leftarrow \mathrm{A}^*(p_s, p_g, \mathcal{G}_n)$
9: **for** each environmental step **do**
10: $F_d \leftarrow \phi_d(o_{t-T_h+1:t})$ and $F_k \leftarrow \phi_k(\tau_g, m_t)$
11: $\breve{F}_d \leftarrow \phi_d(o_{t-T_h-T_p+1:t})$ centered at $t = T_h$
12: $Z \leftarrow \nu_\eta(F_d)$
13: $H_l \leftarrow MLP(Z)$, then $H_g \leftarrow DAP(H_l)$
14: $s_t = \{H_g, F_k\}$
15: $a_t \sim \pi_\theta(a_t|s_t)$, $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$
16: $\hat{\tau}_t \leftarrow \rho_\zeta(Z, o_t)$
17: $c_t^+ = c(s_t, a_t) + \hat{c}(\hat{\tau}_t)$
18: $\mathcal{D} \leftarrow \mathcal{D} \cap \{s_t, a_t, r(s_t, a_t), c_t^+, s_{t+1}, \breve{F}_d\}$
19: **for** each gradient step **do**
20: Randomly sample experience from $\mathcal{D}$
21: Update η, ζ with $\nabla_{\eta,\zeta}\mathcal{L}_P$ from Eq. 6 using $\breve{F}_d$
22: $\theta, \psi, \mu, \beta, \kappa \leftarrow$ WCSAC$(s_t, a_t, r_t, c_t^+, s_{t+1})$
23: Update $< \bar{\theta}, \bar{\psi}, \bar{\mu} >$ with $< \theta, \psi, \mu >$

3) Optimization: Here, the **G**raph-based **IN**tention-aware constraint policy optimization, namely **GIN**, is explained through Alg. 1. Before training loop, we obtain τ_g that follows $\mathcal{G}_n$ using the A* algorithm (lines 7, 8). At each step, the agent interacts with the environment, generates samples during training, and stores them in $\mathcal{D}$ (lines 9-18). ϕ_d and ϕ_k operate separately to create F_d and F_k (line 10). F_d describes dynamic objects through history of observations and F_k is to explain the kinematics of the system. Additionally, we accumulate $\breve{F}_d$ centered at T_h of length $T_h + T_p$ in the time domain to use the data for predicting $\hat{\tau}_t$ (line 11). Given processed state, the agent executes an action, and proceeds to the next state (lines 12-15). To avoid long-term risks considering this interaction, we use the auxiliary cost $\hat{c}_\rho$ in Eq. 8. It is obtained using $\hat{\tau}_t$ obtained from decoder ρ_ζ (lines 16-18). For the gradient step, all model parameters are updated with batches sampled from $\mathcal{D}$ (lines 20-23). The encoder and decoder are updated using the prediction loss in Eq. 6. The parameters $< \theta, \psi, \mu >$ of the actor, critic,

TABLE I

COMPARISON AVERAGE RESULTS WITH STANDARD DEVIATION IN NAVIGATION SCENARIO

METHOD	RETURN	SUCCESS RATE	DISTANCE	COST	COLLISION RATE	SURVIVAL STEP	SUCCESS STEP
SAC	945 ± 598	0.48 ± 0.07	122 ± 73	11.5 ± 16.6	0.44 ± 0.06	254 ± 148	366 ± 75
WCSAC	984 ± 557	0.54 ± 0.11	130 ± 71	7.2 ±13.1	0.35 ± 0.12	**325** ± 179	432 ± 86
GIN (Ours)	**1284** ± **488** (30.5%↑)	**0.76** ± **0.03** (40.7%↑)	**161** ± **58** (23.8%↑)	**4.0** ± **10.4** (44.4%↓)	**0.19** ± **0.03** (45.7%↓)	303 ± **119** (6.8%↓)	**337** ± **70** (7.9%↓)

and approximated cost, and the adaptable weights $< \beta, \kappa >$ are learned by the WCSAC algorithm using accumulated samples through Eq. 3. In our case, we use $d = 5$ because we partially allow unsafe interaction predicted by $\hat{c}_\rho$. Finally, the parameters of the target networks $< \bar{\theta}, \bar{\psi}, \bar{\mu} >$ are updated with moving averages for learning stability.

V. EXPERIMENTS

A. Simulation Settings

To evaluate our method, we set up a navigation environment that contained diverse scenes utilizing the CARLA simulator [32], as shown in Fig. 5. In our experiment, since the default traffic flow is conservative, we adjust the scenario settings as follows: a fixed route length is randomly selected, traffic is densely generated by using all possible spawn points, the speed difference percentage for each vehicle is 20%, traffic lights and signs are ignored to create abnormal patterns. Finally, to track the path obtained by the A^* algorithm, the reward function r is designed as follows:

$$r = v_s - |v_d| - 0.2e_c - 10\Delta e_c - 0.5|\delta_s v_s| - 0.5|e_h v_s| \quad (9)$$

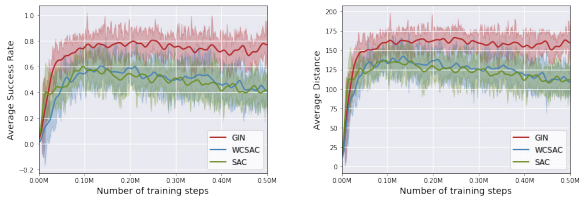
where v_s and v_d are the longitudinal and latitudinal velocities over the reference path, e_c and e_h are the cross-tracking and heading angle errors, respectively, and δ_s is the steering wheel angle. The coefficient of each term is tuned through a coarse grid search. To reflect the collision with the control, the environmental cost function is calculated using the normalized collision intensity.

B. Metrics

We set seven measurements to compare the extent to which the driving strategies are tactical. Return, distance, cost, and survival step are measured with the total cumulative rewards, traveled distance, cumulative costs, and steps, respectively, during an episode. Success rate is the number of episodes in which the destination is reached over all episodes. Collision rate is the number of episodes in which collisions occur over all episodes. Finally, the success step refers to the total number of steps taken over all successful episodes. Furthermore, to compare how accurately motions are predicted, we use the average displacement error (ADE) and final displacement error (FDE). All the above metrics are measured using the averages of five seeds over 100 episodes.

C. Tactical Driving Strategy

We evaluate performance in terms of efficiency and safety. The SAC [33] and WCSAC [15] algorithms based on the off-policy maximum entropy RL framework are adopted as baselines. For a fair comparison, baselines also use separate features for path tracking and dynamic objects as inputs. The difference is that they employ observations only in a single time frame. Subsequently, max pooling is applied



(a) Success Rate (b) Distance

Fig. 6. Learning curve for navigation environment. The darker-colored lines and shaded areas denote the average return and standard deviations.

to obtain the global features. Table. I shows the averages with standard deviations for all metrics for each method. WCSAC exhibits slightly better performance than SAC. This is because using the worst-case measurement reduces the probability of collision, as shown in the cost and collision rate. Nevertheless, a high percentage of crash situations still exists. This is because the long-term dynamic risks do not tend to be reflected due to sparse cost signals. Furthermore, although WCSAC increases the survival steps, the augmented values in the success step indicate a loss in terms of efficiency by driving conservatively.

In contrast, our method shows dramatic improvements in all respects, compared to baselines. Although our method has a lower value than WCSAC in terms of survival steps, it has an efficient driving strategy in terms of a considerable decrease in reaching steps to the goal. Moreover, according to Fig. 6, unlike baselines that show similar curves over success rate and collision rate, our method shows a significant improvement in sample efficiency and stability. This is because it considers social interactions by capturing the local connectivity between agents through the encoder. It also reduces the collision rate by utilizing the dense long-term cost signals predicted by decoders as additional constraints. Although SAC has a faster initial learning speed owing to more aggressive driving than WCSAC, conflicts become more frequent as learning progresses. Consequently, the performance gradually converges to a lower value. In contrast, WCSAC maintains a slightly better final performance.

D. Robust Behavior Prediction

To evaluate the performance of motion prediction, we adopt vanilla-gated recurrent unit (V-GRU), multi-modal transformer (mmTrans) [4] and graph-based interaction-aware trajectory prediction (GRIP) [8] as baselines. V-GRU corresponds to the GRIP without the graph convolution module. For fair comparison, the map aggregator part is excluded in mmTrans because other methods do not use HD map. For GRIP, the control part is excluded from GIN. For baselines, 200 successful episodes are used for training. Each episode is performed using the autopilot agent obtained from CARLA. To validate the robustness of various scenes,

TABLE II

COMPARISON TRAJECTORY PREDICTION RESULTS

METHOD	ADE	FDE	TIME (ms)
V-GRU	0.57 ± 0.10	1.45 ± 0.26	**26.92 ± 1.71**
mmTrans	0.58 ± 0.12	1.26 ± 0.26	94.09 ± 20.06
GRIP	0.55 ± 0.07	1.32 ± 0.20	63.87 ± 4.98
GIN (Ours)	**0.48 ± 0.06** (12.7%↓)	**1.15 ± 0.19** (8.7%↓)	62.74 ± 4.79

TABLE III

CHANGES IN PERFORMANCE WHILE ADJUSTING MODEL

INDEX	COST	WEDGE	DAP	SUCCESS RATE
B1				0.41 ± 0.03
B2		✓		0.48 ± 0.08
B3		✓	✓	0.71 ± 0.05
B4	✓			0.50 ± 0.04
B5	✓	✓		0.55 ± 0.09
B6	✓		✓	0.73 ± 0.02
B7	✓	✓	✓	0.74 ± 0.05

20 episodes that included both successes and failures are collected as test data.

According to Table. II, the accuracy is compared through ADE and FDE, and the efficiency is compared through the computation time for samples of 500 batch size. mmTrans and GRIP have similar values to V-GRU in ADE but significantly reduce errors in FDE. This is because most vehicles interact longitudinally along lane. Although mmTrans has better average score in FDE than GRIP, it has high standard deviation and longer computation time. In contrast, our method shows a greater improvement in both accuracy metrics. Moreover, our method performs better than mmTrans while maintaining a similar operation time to GRIP. Notably, this performance is achieved even without the prepared training data. This shows the advantage of utilizing data that include various scenes obtained in the trial-and-error process of RL. It is difficult for the original GCN to generalize the dynamic graph, whereas our method overcomes this difficulty by combining graph feature learning with RL.

E. Ablation Study

To validate our encoder modeling method for social context extraction, we compare the performances while replacing our encoder with four different architectures: a hidden Markov model (HMM), V-GRU, mmTrans, and GRIP. Specifically, HMM is pretrained with 500 samples of accident-free scenes. To apply the HMM, the observation variable is modeled as a Gaussian variable, and the social context is approximated using a categorical variable with 3-levels. Incidentally, V-GRU and mmTrans are the same as the experiment in Table. II of Section V.D. For a fair comparison, Eq. 6, which trains the decoder, is excluded. Moreover, all policies are equivalently optimized using Eq. 3. As presented in Table. IV, mmTrans and GRIP show a dramatic improvement over others. This is because the spatiotemporally abstracted features effectively express the dynamic interactions. Although GRIP has slightly higher success step than mmTrans, it has high performance in other metrics and stable convergence with low variance. Additionally, since GRIP has low model capacity, it can converge fast with little data, making it more suitable as the backbone of the RL framework than mmTrans.

TABLE IV

COMPARISON OF ENCODER MODELING METHOD

METHOD	SUCCESS RATE	COLLISION RATE	SUCCESS STEP
HMM	0.23 ± 0.07	0.25 ± 0.11	508.78 ± 98.75
V-GRU	0.19 ± 0.05	0.26 ± 0.06	451.80 ± 93.04
mmTrans	0.68 ± 0.13	0.24 ± 0.11	**330.63 ± 40.12**
GRIP	**0.74 ± 0.02**	**0.20 ± 0.05**	364.30 ± 77.29

TABLE V

CHANGES IN PERFORMANCE WHILE ADJUSTING AUXILIARY COST

TYPE	PREDICTION LENGTH	SUCCESS RATE	COLLISION RATE	SUCCESS STEP
Polygon	-	0.65	0.31	**282.92**
Polygon + Polyline	12	0.74	0.21	327.87
	24	**0.79**	**0.16**	316.85
	36	0.71	0.18	345.84
Polygon + Temporal Distance	24	0.58	0.18	434.95

Table. III reveals how much each module contributes to the performance improvement. By comparing B1-B3 and B4-B7, the impact of modifying each module is evaluated depending on whether costs are used for constraint policy optimization. The differences from B1 to B2, from B4 to B5, and from B6 to B7 represent the degree of improvement by using weighted edge (Wedge) in Eq. 5. Comparing B2 to B3 and B5 to B6, the distance average pooling (DAP) in Eq. 7 shows a significant improvement because it helps ego-centered control by paying attention to spatially nearby objects.

Table. V shows the effect by type of auxiliary cost signals used in Eq. 8. The *Polygon* represents the costs are generated only in case of Fig. 4(a). For the *Polyline*, the costs correspond to Fig. 4(b-d), and the *Temporal Distance* uses the costs with the minimum clearance between the ego-vehicle and other cars across time. Using *Polygon* alone except for *Polyline* cannot detect long-term risks, resulting in frequent collisions. However, using *Polygon* and *Polyline* achieves high improvement without being overly conservative by choosing an appropriate prediction time horizon. Meanwhile, using *Temporal Distance* instead of *Polyline* makes driving overly conservative because it incurs unnecessary costs although there is no actual intersection between ego and other trajectories.

F. Visualization

To illustrate the social context vectors, we cluster the distribution using t-distributed stochastic neighbor embedding (t-SNE). The trajectory distribution for each cluster is divided according to the direction and shape, as shown in 7. Specifically, the 6th cluster (yellow) is a distribution that moves in the opposite direction to the ego-vehicle, the 4th distribution (brown) moves in the vertical direction, and the 0th distribution (blue) refers to the intention to stand still and move forward. Others (black) represent the same direction, but have slightly different shapes. This is because most of the patterns encountered are in the same direction, and the latent vectors are in similar locations.

VI. CONCLUSION

We efficiently integrate motion prediction and control modules by summarizing the dynamic scene with the shared

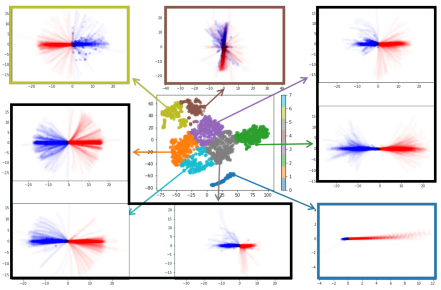


Fig. 7. Visualization of social context with trajectory distributions. Each axis represents a value for x and y in the Cartesian coordinate system. The color for distinguishing the social context indicates the label for each trajectory distribution. In the trajectory distribution, the blue portion denotes the previous history, whereas the red portion represents the predicted future.

vector reflects the spatiotemporal interaction through the graph-based network. Furthermore, we enable policies to avoid long-term risks by using cost signals obtained from prediction results as constraints. Incorporating the two modules through a safe RL framework makes predictions robust in risky situations. This significantly improves the performance of navigation problems with dense traffic. In addition, visualization of the learned latent variables shows how our method captures interactions. Although there remains a limitation that steering is unstable, it can be mitigated through post-processing or replacement with a stable control module. In the future, we will extend our work to reflect road context information obtained from visual sensors in graph form without using HD maps. Another exciting direction would be to investigate the multimodal intentions of the surrounding heterogeneous agents.

REFERENCES

[1] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[2] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *International conference on machine learning*. PMLR, 2017, pp. 22–31.

[3] Y. Yoon, T. Kim, H. Lee, and J. Park, "Road-aware trajectory prediction for autonomous driving on highways," *Sensors*, vol. 20, no. 17, p. 4703, 2020.

[4] Y. Liu, J. Zhang, L. Fang, Q. Jiang, and B. Zhou, "Multimodal motion prediction with stacked transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7577–7586.

[5] H. Song, D. Luan, W. Ding, M. Y. Wang, and Q. Chen, "Learning to predict vehicle trajectories with model-based planning," in *Conference on Robot Learning*. PMLR, 2022, pp. 1035–1045.

[6] M. Schreier, V. Willert, and J. Adamy, "Bayesian, maneuver-based, long-term trajectory prediction and criticality assessment for driver assistance systems," in *17th international ieee conference on intelligent transportation systems (ITSC)*. IEEE, 2014, pp. 334–341.

[7] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[8] X. Li, X. Ying, and M. C. Chuah, "Grip: Graph-based interaction-aware trajectory prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3960–3966.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] E. Altman, *Constrained Markov decision processes: stochastic modeling*. Routledge, 1999.

[11] C. Tessler, D. J. Mankowitz, and S. Mannor, "Reward constrained policy optimization," *arXiv preprint arXiv:1805.11074*, 2018.

[12] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[13] R. T. Rockafellar, S. Uryasev *et al.*, "Optimization of conditional value-at-risk," *Journal of risk*, vol. 2, pp. 21–42, 2000.

[14] Y. C. Tang, J. Zhang, and R. Salakhutdinov, "Worst cases policy gradients," *arXiv preprint arXiv:1911.03618*, 2019.

[15] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, "Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence. AAAI Press, online*, 2021.

[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. Nilsson, L. Laine, N. Van Duijkeren, and B. Jacobson, "Automated highway lane changes of long vehicle combinations: A specific comparison between driver model based control and non-linear model predictive control," in *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)*. IEEE, 2015, pp. 1–8.

[18] F. Damerow and J. Eggert, "Risk-aversive behavior planning under multiple situations with uncertainty," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 656–663.

[19] M. J. Kochenderfer, *Decision making under uncertainty: theory and application*. MIT press, 2015.

[20] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE transactions on intelligent vehicles*, vol. 5, no. 2, pp. 294–305, 2019.

[21] C. Kim, H.-S. Yoon, S.-W. Seo, and S.-W. Kim, "Stfp: Simultaneous traffic scene forecasting and planning for autonomous driving," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6016–6022.

[22] Z. Wang, Y. Zhuang, Q. Gu, D. Chen, H. Zhang, and W. Liu, "Reinforcement learning based negotiation-aware motion planning of autonomous vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4532–4537.

[23] S. Qi and S.-C. Zhu, "Intent-aware multi-agent reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7533–7540.

[24] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202*, 2018.

[25] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.

[26] V. Khokhlov, "Conditional value-at-risk for elliptical distributions," *Evropskỳ časopis ekonomiky a managementu*, vol. 2, no. 6, pp. 70–79, 2016.

[27] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1352–1361.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[29] D. Keirsey, E. Koch, J. McKisson, A. Meystel, and J. Mitchell, "Algorithm of navigation for a mobile robot," in *Proceedings. 1984 IEEE International Conference on Robotics and Automation*, vol. 1. IEEE, 1984, pp. 574–583.

[30] S. Gottschalk, "Separating axis theorem. technical report tr96-024," *Department of Computer Science. University of North Carolina, Chapel Hill*, pp. 20–46, 1996.

[31] B. Chazelle and H. Edelsbrunner, "An optimal algorithm for intersecting line segments in the plane," *Journal of the ACM (JACM)*, vol. 39, no. 1, pp. 1–54, 1992.

[32] 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.

[33] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.