IEEE Robotics and Automation Letters, Volume 8, Issue 6, June 2023, pp. 3685-3692
DOI:10.1109/LRA.2023.3270739

Map-Adaptive Multimodal Trajectory Prediction Using Hierarchical Graph Neural Networks

Xiaoyu Mo, *Student Member, IEEE,* Yang Xing, *Member, IEEE,* Haochen Liu, *Student Member, IEEE,* and Chen Lv, *Senior Member, IEEE*

***Abstract*—Predicting the multimodal future motions of neighboring agents is essential for an autonomous vehicle to navigate complex scenarios. It is challenging as the motion of an agent is affected by the complex interaction among itself, other agents, and the local roads. Unlike most existing works, which predict a fixed number of possible future motions of an agent, we propose a map-adaptive predictor that can predict a variable number of future trajectories of an agent according to the number of lanes with candidate centerlines (CCLs). The predictor predicts not only future motions guided by single CCLs but also a scene-reasoning prediction and a motion-maintaining prediction. These three kinds of predictions are produced integrally via a single graph operation. We represent the driving scene with a heterogeneous hierarchical graph containing nodes of two types. An agent node contains its dynamics feature encoded from its historical states, and a CCL node contains the CCL's sequential feature. We propose a hierarchical graph operator (HGO) with an edge-masking technology to regulate the information flow in graph operations and obtain the encoded scene feature for the trajectory decoder. Experiments on two large-scale real-world driving datasets show that our method realizes map-adaptive prediction and outperforms strong baselines.**

***Index Terms*—Map-adaptive trajectory prediction, connected vehicles, graph neural networks, heterogeneous interactions.**

I. Introduction

An autonomous vehicle has to predict the future trajectories of its neighboring traffic participants to navigate in highly dynamic driving situations [1]. However, trajectory prediction is non-trivial due to several challenges. First, the number of traffic participants can vary from case to case. Second, the road structure is quite complex and shapes the motion of vehicles to a great extent. Third, the interdependencies among vehicles and infrastructures are complicated [2]. Fourth, driving behavior is multimodal, meaning that given the same situation, there may exist different future motions. For example, a vehicle approaching an intersection can either go straight forward or turn left (right). Fifth, driving behavior is not always rational in some critical corner cases.

Most recent works have proposed to jointly consider the target agent's own dynamics, its interaction with surrounding agents, and the impacts of the infrastructure. Surrounding agents are the agents that can potentially impact the target vehicle's future motion. The selection of criteria varies from study to study. Existing works represent the agents and the map either separately or integrally and try to predict multimodal future motions of target agents [3]–[8]. However, scene representation and multimodality for trajectory prediction are far from being well explored. Most existing works predict a predefined number of possible future motions of target agents across all situations. Thus they are non-map-adaptive. A prediction set with a fixed number of options limits the model to generalize to complex map geometries (e.g., a method with three outputs cannot cover all options of a target vehicle approaching a four-way intersection). These limitations can be addressed by map-adaptive trajectory prediction methods, which can output an arbitrary number of predictions according to a set of goal paths in specific scenes [6] as long as the goal paths can be retrieved from the local lane graph. Map-adaptive methods also make driving modes more interpretable by associating them with target paths.

In this work, we attempt to represent the complex driving scene and predict the multimodal motions of a target vehicle in an integrated manner. We represent the driving scene with a heterogeneous hierarchical graph, wherein a node is either an agent or one of its CCLs and contains the corresponding feature. This representation can accommodate an arbitrary number of agents and their centerlines, thus tackling the first two challenges. We represent interdependencies among nodes with directed edges and propose a three-stage graph operator to encode the scene graph. An edge-masking technology is used to regulate information flow in different stages. The hierarchical graph operator is designed to cope with the third challenge. To handle the fourth and fifth challenges, we design an integrated multimodal predictor via graph operation and edge-masking that can simultaneously predict CCL-following, scene-reasoning, and motion-maintaining future trajectories of a target agent. The graph operation allows our predictor to output a variable number of trajectories according to the target agent's CCLs. Thus our method is map-adaptive.

The main contributions of this work can be summarised as follows:

- At the high level, this work designs a map-adaptive multimodal trajectory prediction framework that jointly considers the target agent's own dynamics, interaction with other agents, and the local roads.
- For scene representation and encoding, this work represents the complex driving scene with a single heterogeneous hierarchical graph. It proposes a hierarchical graph operator augmented with an edge-masking technology to encode the scene graph for trajectory prediction.

Xiaoyu Mo, Haochen Liu, and Chen Lv are with the School of Mechanical and Aerospace Engineering, Nanyang Technological University, 639798, Singapore. (e-mail: xiaoyu006@e.ntu.edu.sg, haochen002@e.ntu.edu.sg, lyuchen@ntu.edu.sg)

Yang Xing is with the Centre for Autonomous and Cyber-Physical Systems at Cranfield University, UK, MK43 0AL. (e-mail: yang.x@cranfield.ac.uk)

- For map-adaptive trajectory decoding, this work proposes a comprehensive CCL-enabled map-adaptive multimodal predictor implemented with a graph operation. It produces three kinds of predictions simultaneously: 1) a set of CCL-following trajectories that is adaptive to the road topology and can generalize to unseen road structures; 2) a scene-reasoning trajectory that estimates the target agent's future movements based on its understanding of the scene; 3) a motion-maintaining trajectory that covers the cases where the vehicle just maintains its motion.

II. Related Work

A. Graph-Based Scene Representations

A graph is a natural way to represent the relationships (e.g., either or both of inter-agent interaction and lanelet connectivity) among objects [9]. ReCoG represents the relationships among agents and the map via a star-like heterogeneous graph, wherein the map node contains features extracted from a raster image [10]. HEAT represents the relationships among different kinds of agents using a heterogeneous edge-enhanced graph. Map information is considered via another channel [11]. VectorNet represents the driving scene with a hierarchical heterogeneous graph, where a higher-level node contains either an agent's trajectory or a map feature, and a lower-level node is a vector in a sequence (polyline). The graph is constructed with the full connection. The fully-connected graph structure is inefficient since the number of edges in the graph grows quadratically with the number of nodes [12]. LaneGCN represents actors and the map separately. It constructs a lane graph from the high-definition map (HD map) preserving connectivity between lane nodes and saves four types of connections for downstream encoders [5]. Rather than representing agents as single nodes, LaneRCNN constructs a dynamics-related lane graph for each agent, with each node containing geometric, semantic, and agent information [8].

We represent the driving scene with a heterogeneous hierarchical graph containing both agent and CCL nodes. An agent's CCLs are only connected to the agent itself, and all the surrounding agents are connected only to the target agent. This star-like connection makes the number of edges grows linearly, rather than quadratically, with the number of nodes. An edge in the graph is assigned with a type label, and the connection of the graph can be stored in a sparse manner. Rather than saving multiple edge sets and applying a specific graph neural network (GNN) for each, we utilize an edge-masking strategy to regulate information flow in the graph and apply a GNN for a subset of edge types hierarchically.

B. Multimodal Trajectory Predictions

Many works have proposed to output multiple possible future trajectories of a target agent to capture the inherent multimodality of a moving agent's future motion. MTP uses a convolutional neural network (CNN) to produce a fixed number of trajectories for a target agent. It shows that setting the model to produce three modalities gives the best performance on their metrics [3]. MultiPath first clusters trajectories in the training set via K-means to obtain a set of anchor trajectories, then reformulates the task into anchor selection and regression [4]. CoverNet dynamically generates a trajectory set according to the target agent's current dynamic state, then follows the anchor selection and regression operations introduced in MultiPath [4], [13]. TNT first predicts a fixed number of targets (final points in a fixed prediction horizon) for a given agent, then estimates trajectories conditioned on the targets, and finally scores and selects a fixed number of trajectories as the final prediction [7]. Heatmaps are also used in some recent works for goal generation [14], [15].

Rather than selecting a fixed number of anchors or targets for all driving scenarios, GoalNet predicts a variable number of trajectories that are adaptive to the road structure [6]. Specifically, it predicts a set of goal-based and goal-free trajectories of a target vehicle, where the goal is a forward path that the target vehicle can potentially choose to follow in the next seconds. It combines the traffic information in the goal paths and the target vehicle's dynamics to output goal-free predictions. However, GoalNet ignores the vehicles behind the target and those in other lanes, which can have a critical impact on the target vehicle's behavior. For example, the behavior of a target vehicle operating an unprotected left turn will be affected by the vehicles in the lane on its left side. However, GoalNet does not consider this information. In addition, the goal-free prediction, although also named motion-maintaining, is not purely based on the target vehicle's past motion, thus unable to cover the cases where a vehicle just maintains its motion. Motion-maintaining can be in either normal (e.g., cruising) or critical situations (e.g., brake failure).

To address the above-mentioned limitations of GoalNet, we propose a map-adaptive multimodal trajectory predictor that can predict CCL-following, scene-reasoning, and motion-maintaining trajectories of a target agent simultaneously based on a comprehensive representation of all nearby agents and CCLs. CCL-following predictions condition the target agent's future motions on individual CCLs, while the scene-reasoning prediction outputs a reasonable prediction based on the information available in the scene, in case a specific CCL cannot explain the target agent's behavior (e.g., parking, overtaking, and pulling onto the shoulder [6]) or there is no CCL can be retrieved from the lane graph (e.g., CCL selection method failure or lane graph error). The motion-maintaining prediction extrapolates the actor's current motion into the future to capture non-map-compliant behaviors in some critical cases for safety concerns.

III. Method

A. Input and Output.

This work aims to predict a set of multimodal trajectories of a target agent given dynamics of considered agents and the local map. At a time t, the input $\mathbf{X}_t$ contains historical states of considered agents and their CCLs:

$$\mathbf{X}_t = [H_t, C_t], \tag{1}$$

where $H_t = \{h_t^1, h_t^2, \ldots, h_t^n\}$ contains the historical states of n agents at time t, and $C_t = \{c_t^{1,1}, c_t^{1,2}, \ldots, c_t^{2,1}, c_t^{2,2}, \ldots, c_t^{n,m-1}, c_t^{n,m}\}$ contains

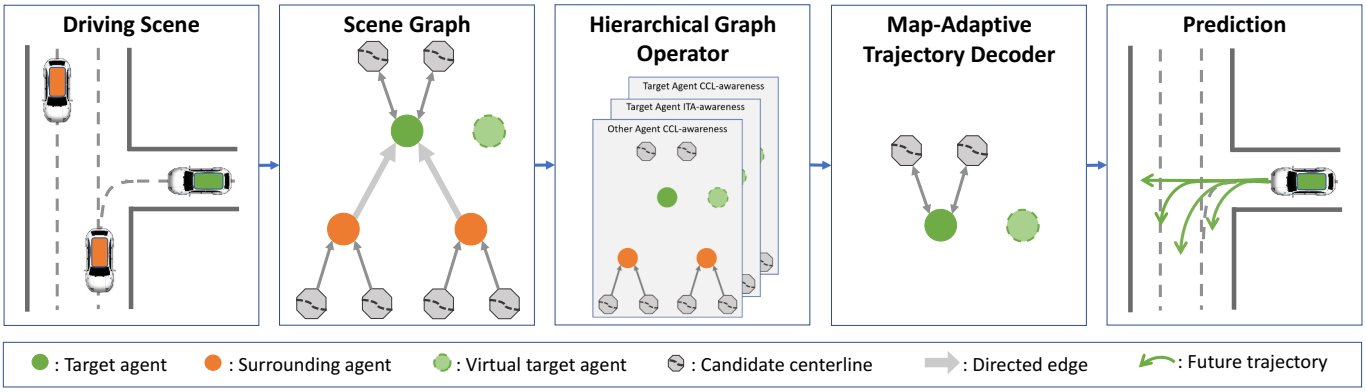

Fig. 1. **Overview of the proposed scheme.** Given a driving scene consisting of agents and the HD map, we first assign a variable number of CCLs to each agent according to its dynamics and the road structure. Then, we represent the driving scene with a heterogeneous hierarchical graph (scene graph) with an additional virtual target agent node. Next, we process the scene graph using our proposed hierarchical graph operator. Finally, we apply a map-adaptive trajectory decoder to predict a variable number of trajectories. These predictions of a target agent fall into three categories: CCL-following, scene-reasoning, and motion-maintaining predictions.

the CCLs of each agent. $h_t^i = [s_{t-T_h+1}^i, s_{t-T_h+2}^i, \ldots, s_t^i]$ is the historical states of agent i over a traceback horizon T_h, where $s_t^i = [x_t^i, y_t^i, v_{x_t}^i, v_{y_t}^i]$ is the state (position and velocity) of agent i at time t. $c_t^{i,j} = [(x,y)_1^{i,j}, (x,y)_2^{i,j}, \ldots, (x,y)_L^{i,j}]$ is the j_{th} CCL of agent i at time t that contains L way-points. Note that the number of considered agents n and the number of CCLs of an agent m vary from case to case. Without loss of generality, we number the target agent with one. Then the output is a set of future trajectories of the target agent:

$$F_t = \{f_t^{1,1}, f_t^{1,2}, \ldots, f_t^{1,m}, f_t^{1,m+1}, f_t^{1,m+2}\}, \tag{2}$$

where $f_t^{1,j} = [(x_{t+1}^{1,j}, y_{t+1}^{1,j}), \ldots, (x_{t+T_f}^{1,j}, y_{t+T_f}^{1,j})]$ is j_{th} sequence of predicted XY coordinates of the target agent over a prediction horizon T_f. The first m predictions are based on the target agent's m CCLs. $f_t^{1,m+1}$ is the scene-reasoning prediction and $f_t^{1,m+2}$ is the motion-maintaining prediction. The predicted trajectories are further described in Sub.Sec. III-E.

B. Heterogeneous Hierarchical Scene Graph

In this work, we represent the driving scene as a heterogeneous hierarchical graph, where the nodes fall into two categories: agents and CCLs. The hierarchical graph contains two layers, where the lower layer is the agent-CCL graph and the upper layer is the inter-agent interaction graph. The agent-CCL graph is a star-like graph with the agent at the center and all its CCLs linked to it (indicated by dark gray arrows in the second block of Fig. 1). The interaction graph is another star-like graph with the target agent at the center and all neighboring nodes linked to it (indicated by light gray arrows in the second block of Fig. 1). In addition to the objects in the driving scene, we introduce a virtual target agent node (light green node with a dashed border in the second block of Fig. 1) for motion-maintaining prediction. The virtual node is isolated in the graph and has no CCL nodes to form a sub-graph. We also assume that each node in the graph has a self-loop for information preservation. But, for clarity, these self-loops are not plotted.

There are many advantages of this representation: 1) the graph representation can accommodate an arbitrary number of objects; 2) the heterogeneous graph can comprehensively represent different kinds of objects; 3) the star-like graph structure is sparse so that it is more efficient compared to graphs with dense connectivity [7]; 4) the hierarchical structure allows information flow from local to global; 5) the introduced virtual node preserves the target agent's dynamics for motion-maintaining prediction.

Candidate centerlines selection. The main idea of centerline selection is operating depth-first search (DFS) on the lane graph. More description of CCL selection can be found in Sub.Sec. IV-A.

Graph construction. We construct a heterogeneous hierarchical graph to represent the interaction among agents and CCLs. We use $\mathcal{A}$ and $\mathcal{C}$ to contain the indices of agent and CCL nodes in the graph. These two types of objects (agents and CCLs) are further divided into four types of nodes (target agent, surrounding agent, target agent's CCL, and surrounding agents' CCL). In addition to these nodes, we introduce a virtual target node in the constructed graph to integrate motion-maintaining prediction. For an agent node, the raw node feature is the agent's historical states. For a CCL node, the raw node feature is a sequence of XY coordinates of this CCL. A directed edge pointing from node j to node i means that node j has an impact on node i, and there will be information flow from node j to node i in graph operations. Each edge is associated with an edge type determined by the edge's source and target nodes. The edge set is represented as:

$$E = \{e_{ij}\}_{(j \in \mathcal{N}_i)}, \quad i = 1, \ldots, N, \tag{3}$$

where e_{ij} is a directed edge from node j (the source node) to node i (the target node), $\mathcal{N}_i$ is the neighborhood of node i, and N is the total number of nodes in the graph. Self-loops e_{ii} are included in the edge set because a node is also counted as a neighbor of itself. An example of the constructed graph is shown in the second block of Fig. 1. Tab. I lists the node and edge types in this heterogeneous hierarchical graph.

C. Agent and CCL Encoders

Since there are two kinds of objects in our scene graph, i.e., vehicles and their CCLs, we introduce one shared encoder for

TABLE I
NODES AND EDGES IN THE SCENE GRAPH

TarAg	Target agent node
VirTarAg	Virtual target agent node
SurAg	Surrounding agent node
TarCCL	Target agent's CCL node
SurCCL	Surrounding agent's CCL node
TarAg-Loop	Self-loop of the *TarAg* node
VirTarAg-Loop	Self-loop of the *VirTarAg* node
SurAg-Loop	Self-loop of the *SurAg* node
TarCCL-Loop	Self-loop of the *TarCCL* node
SurCCL-Loop	Self-loop of the *SurCCL* node
SurCCL→SurAg	Edge from *SurCCL* node to *SurAg* node
SurAg→TarAg	Edge from *SurAg* node to *TarAg* node
TarCCL→TarAg	Edge from *TarCCL* node to *TarAg* node
TarAg→TarCCL	Edge from *TarAg* node to *TarCCL* node

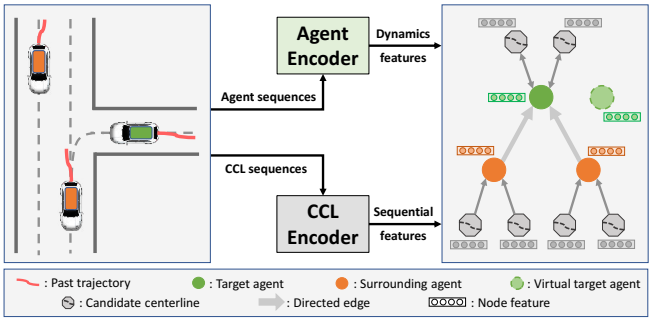

Fig. 2. **Agent and CCL encoders.** Given the agents' historical states and CCLs, we apply agent encoder and CCL encoder to extract sequential dependencies in corresponding sequences. Then we take the extracted features as node features of the scene graph.

each type. We assume that the CCLs are sequences of XY coordinates and the historical states of vehicles are sequences of their positions and velocities over the last two seconds.

1) Agent Dynamics Encoder: An agent is represented by a sequence of its historical states. We use a gated recurrent unit (GRU [16]) network to model its dynamics from its historical states:

$$d_t^i = \mathrm{GRU}_{\mathrm{agn}}(h_t^i), i \in \mathcal{A}, \tag{4}$$

where h_t^i is the historical sequence of vehicle node i at time t, $\mathrm{GRU}_{\mathrm{agn}}$ is the GRU network for agent dynamics encoding, and d_t^i is is the extracted dynamics feature.

2) Candidate Centerline Encoder: A CCL is represented by a sequence of XY coordinates. We use another GRU network to model the sequential dependencies in a centerline sequence:

$$q_t^j = \mathrm{GRU}_{\mathrm{ccl}}(c_t^j), j \in \mathcal{C}, \tag{5}$$

where c_t^j is the way-point sequence of CCL j at time t, $\mathrm{GRU}_{\mathrm{ccl}}$ is the GRU network for centerline encoding, and q_t^j is is the extracted sequential feature. Then the extracted features (d_t^i and q_t^j for all agents and CCLs) are reordered and renamed to r_t^i and taken as node features of the scene graph :$R_t = \{r_t^1, r_t^2, \ldots, r_t^i, \ldots, r_t^j, \ldots, r_t^{N-1}, r_t^N\}$. Please see Fig. 2 for an illustration of sequence encoding.

D. Hierarchical Graph Operator

In this subsection, we introduce the hierarchical graph operator (HGO) with the edge-masking technique (III-D1) designed to encode the scene graph. HGO consists of three stages, namely 1) surrounding agents' CCL-awareness, 2) the target agent's interaction-awareness, and 3) the target agent's CCL-awareness. The first stage (III-D2) allows the surrounding agents to gather information from their CCLs. The second stage (III-D3) then allows the target agent to model its interaction with the surrounding agents. The third stage (III-D4) then brings CCL-awareness to the target agent. Each stage is implemented with a separate Graph Attention Network (GAT [17]) with information flow regulated by the edge-masking technology. GAT is selected considering that: 1) its effectiveness has been proven in many graph learning tasks [18]; 2) it operates in the local neighborhood; 3) its attention mechanism allows modeling the importance of different factors. It is worth noting that HGO is open to being implemented with other GNNs. The information flow in HGO is shown in Fig. 3. Since GAT is utilized to implement the graph operators throughout this work, we first briefly introduce it before diving into the details.

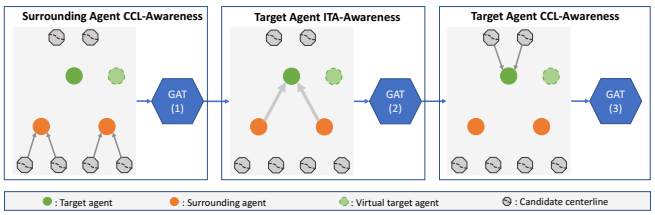

Fig. 3. **Information flow in the hierarchical graph operator (HGO).** The information flow is regulated by edge-masking. The first stage of HGO is for surrounding agent CCL-awareness. The second stage is for target agent interaction-awareness. The third stage is for target agent CCL-awareness. Each stage is implemented with a graph operation augmented with edge-masking.

Graph Attention Network. In this work, we want to model the effects of a target vehicle's surrounding agents and CCLs on its future motion. We represent the interdependencies among them as a graph and use GATs to update node features in each graph operation. For a node i, a GAT layer first computes attention coefficients over its neighborhood. Then it updates the feature of node i via a linear combination of the features of its neighboring nodes according to the normalized attention coefficients. In a GAT operation, information only flows along directed edges so that the information flow can be regulated by manipulating the edge set. This leads us to propose edge-masking for information flow regulation. For details of the GAT layer, please refer to [17].

1) Edge-Masking: Data masking is a technique to hide irrelevant information from a model so that it can be trained on more relevant data for a specific task. This technique is widely used in language and vision models and shows promising improvements [19]. Inspired by data masking, we propose edge-masking, a technique that hides irrelevant edges of a graph before processing it with a GNN, to train task-specific GNNs. In the setting of GNNs, information flows from one node to another through the edge between them. The information in node j can only be passed to node i through GNN operations if there is an edge from j to i. So we can regulate information flow from nodes to nodes (which can be of different types) by hiding irrelevant edges via edge-masking

to train GNNs for specific sub-tasks. This is different from HetGNN [20], which applies a GNN for each type of edge connection. Using the edge-masking technique, we only need to save one edge set with several edge masks for each graph operation.

2) Surrounding Agents' CCL-Awareness: Before modeling interactions between the target and its surrounding agents, we first let these surrounding agents gather information from their own CCLs. This operation, when modeling inter-agent interactions in the following stage, gives the target agent a broader view of the road structure and possible motions of its surrounding agents. We apply a GAT to the entire graph with edge-masking to regulate information flow in this graph operation so that the information only flows from surrounding agents' CCL nodes to themselves. This graph operation is expressed by:

$$G_t^1 = \mathrm{GAT}_1(R_t, E_1), \tag{6}$$

where R_t contains node features for both agent and CCL nodes, E_1 is the edge set retrieved via masking for this stage, GAT_1 is the GAT for this stage, and G_t^1 is the output of this stage. Each surrounding agent node in G_t^1 is with CCL-awareness. All the other nodes, i.e., the target, the virtual target, and all the centerline nodes, remain isolated. The information flow regulated by edge-masking is shown in the first block of Fig. 3. Specifically, the edges of the following types are used in this graph operator: {*SurCCL→SurAg*, *TarAg-Loop*, *SurAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}. Note that these self-loops are included to avoid information loss.

3) Target Agent's Interaction-Awareness: In the second stage, we allow the target agent to gather information from its neighborhood. Since its neighboring agents are aware of their corresponding CCLs, this stage provides interaction-awareness to the target vehicle along with further road awareness from its neighbors. This graph operation is expressed by:

$$G_t^2 = \mathrm{GAT}_2(G_t^1, E_2), \tag{7}$$

where G_t^1 is the output of Eq. 6, E_2 is the edge set retrieved via masking for this stage, GAT_2 is the GAT for this stage, and G_t^2 is the output of this stage. This stage brings interaction-awareness (ITA-awareness) to the target agent node. All the other nodes, i.e., the surrounding agents, the virtual target, and all the CCL nodes, remain isolated. The information flow regulated by edge-masking is shown in the second block of Fig. 3. Specifically, the edges of the following types are used in this stage: {*SurAg→TarAg*, *TarAg-Loop*, *SurAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

4) Target Agent's CCL-Awareness: The third stage is to make the target agent aware of its options, that is, its CCLs. This graph operation is expressed by:

$$G_t^3 = \mathrm{GAT}_3(G_t^2, E_3), \tag{8}$$

where G_t^2 is the output of Eq. 7, E_3 is the edge set retrieved via masking for this stage, GAT_3 is the GAT for this stage, and G_t^3 is the output of this stage. This stage lets the target agent look at its CCLs with knowledge of surrounding agents' options and interactions. All the other nodes, i.e., the surrounding agents, the virtual target, and all the CCL nodes, remain isolated. The information flow regulated by edge-masking is shown in the third block of Fig. 3. Specifically, the edges of the following types are used in this stage: {*TarCCL→TarAg*, *TarAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

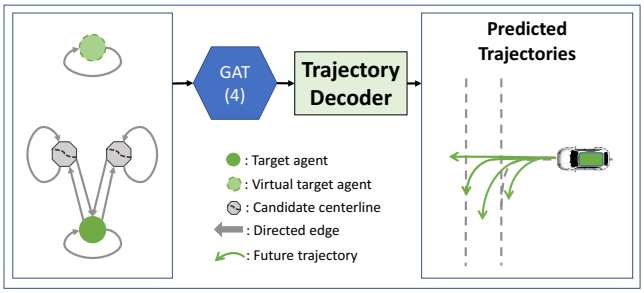

Fig. 4. **Map-adaptive trajectory decoder.** After encoding, we apply another GAT on the graph with edges shown in the left block of this figure. This is to distribute the target agent's feature to the CCL nodes and let the target agent node reconsider its options (CCLs). Then we apply a trajectory decoder to output the final multimodal prediction.

E. Map-Adaptive Trajectory Decoder

Our approach utilizes a variable number of CCLs to predict three kinds of future trajectories of a target vehicle. The number of CCLs depends on the lane geometry of the driving scene, and the predicted trajectories include CCL-following, scene-reasoning, and motion-maintaining predictions. This design is based on the following observations: 1) the road structure mainly shapes the motion of vehicles, and vehicles tend to follow centerlines in driving to keep a safe distance from each other; 2) there are some situations where a vehicle's behavior is determined by more than a single CCL; 3) A vehicle's motion can depend on its own dynamics in some cases.

A CCL-following prediction (Eq. 9) emphasizes the impacts of a specific CCL on the target vehicle's future motion and estimates its future motion under the current situation ($\mathbf{X}_t$) if it is going to follow the CCL ($c_t^{1,i}$). Map-adaptive prediction is realized by iterating over a set of CCLs:

$$f_t^{1,i} = \mathcal{F}_{ccl}(\mathbf{X}_t, c_t^{1,i}), \quad i = 1, \ldots, m. \tag{9}$$

When a vehicle does not follow a CCL (e.g., parking and overtaking), the scene-reasoning prediction (Eq. 10) estimates the target agent's future movements based on its understanding of the scene and covers the unimodal prediction [21]. It also guarantees that the designed model can output an interaction-aware prediction, regardless of whether the target vehicle's CCLs are available or not.

$$f_t^{1,m+1} = \mathcal{F}_{scn}(\mathbf{X}_t). \tag{10}$$

The motion-maintaining prediction (Eq. 11) covers the cases where the target vehicle maintains its current motion (h_t^1):

$$f_t^{1,m+2} = \mathcal{F}_{mot}(h_t^1). \tag{11}$$

It is worth noting that these three prediction types are designed for different purposes. However, in normal driving, vehicles tend to follow a CCL and maintain their motions, so these three types can generate similar predictions in some cases.

To handle the variable number of CCLs, we adopt the graph representation and a GNN in this predictor. Fig. 4 shows an illustration of this predictor. The graph structure used by this predictor is shown in the left block of Fig. 4, which is a heterogeneous graph containing three types of nodes: a target node, a virtual target node, and a set of CCL nodes of the target vehicle. The graph structure is also obtained via edge-masking. Throughout all the previous encoding stages with our information flow regulation strategy, the node features are updated and contain corresponding features for three types of predictions. The target node contains the overall information of the scene; the virtual target node contains its own dynamics; the target vehicle's CCL nodes contain corresponding CCL features. Since we focus on the target agent, all other agents and their CCL nodes are ignored in this stage. Given the number of the target vehicle's CCLs m, our predictor will output $m+2$ predictions. The calculation within this decoder is expressed by:

$$F_t = \mathrm{MLP}_{\mathrm{pred}}\left(\mathrm{GAT}_{\mathrm{pred}}(G_t^3, E_4), Mask_{tar}\right), \qquad (12)$$

where G_t^3 is the output of Eq. 8, E_4 is the edge set retrieved via masking for this stage, $\mathrm{GAT}_{\mathrm{pred}}$ is the GAT used for prediction, $Mask_{tar}$ is used to select the target agent node, the target CCL nodes, and the virtual target node from the output of $\mathrm{GAT}_{\mathrm{pred}}$, $\mathrm{MLP}_{\mathrm{pred}}$ is the trajectory decoder implemented with a multi-layer perceptron, and F_t is the predicted future trajectories of the target agent. F_t contains m CCL-following predictions, one scene-reasoning prediction, and one motion-maintaining prediction. Specifically, the edges of the following types are used in this graph operation: {*TarCCL→TarAg*, *TarAg→TarCCL*, *TarAg-Loop*, *TarCCL-Loop*, *VirTarAg-Loop*}.

IV. Real-World Dataset Validation

A. Datasets

We evaluate our method on the Argoverse motion forecasting dataset [22] and the NuScenes dataset [23]. Both provide two seconds of history states of traffic participants and HD maps. The former requires methods to predict the target vehicle's motions in the next three seconds, while the latter requires predictions of six seconds. The Argoverse dataset provides centerline segments and their connectivity. It also provides a map API (Application Programming Interface) to get a vehicle's CCLs given its latest trajectory. For more details of this map API, please refer to their released code. For the NuScenes dataset, given a vehicle, we first find the lane closest to its current position as the start lane. Then we search the lane graph from the start lane for up to three steps to obtain CCLs. For the six-seconds prediction horizon, we stop extending a CCL if its length exceeds 100 meters. Since this work focuses on trajectory prediction for a single target agent, we use a target-centric coordinate system for all the trajectories and waypoints. This makes our method robust to coordinate transformation since data represented in other coordinate systems can always be transformed to be target-centric. We split both datasets according to instructions provided by the datasets except for selecting only vehicles with a full six-second future trajectory in the NuScenes dataset for training.

B. Comparison with Existing Methods

On the Argoverse dataset. We compare our model with existing models proposed in recent years and two official baselines provided by the Argoverse dataset on the Argoverse motion forecasting benchmark (test set). We adopt Minimum Average Displacement Error (minADE) and Minimum Final Displacement Error (minFDE) in meters to evaluate prediction performances as previous works [7], [10]. Since the benchmark allows up to K=6 predictions, we replace our map-adaptive decoder with a decoder that predicts six possible future trajectories of a target agent for a fair comparison. We report the results in terms of minADE, minFDE, and miss rate (MR). Miss rate is the number of scenarios where none of the predicted trajectories of a target agent are within 2.0 meters of the ground truth according to endpoint error. The MR is listed, but the comparison is mainly on minADE and minFDE. It can be seen from Tab. II that our model significantly outperforms Argoverse baselines (Argo and Argo (NN)). Our model also outperforms Jean (the first place winner of the Argovese motion forecasting challenge 2019), LaneRCNN, TNT, DenseTNT, and GOHOME in terms of minADE (K=6) and minFDE (K=6). Our model is competitive with LaneGCN with only a small gap in prediction accuracies but a much greater reduction in memory cost and inference time. As we show in Tab. III, our method reduces both the required model size and data storage to a great extent and therefore improves inference speed. In Tab. III, we show that LaneGCN uses 3.7 million (M) of parameters and requires 32 gigabytes (G) to save data for training, validation, and testing. It needs 0.08 seconds to operate on a batch of 32 data pieces on a GPU (GeForce RTX 2080 Ti), while our method needs only 0.02 seconds, reducing the inference time by 75%.

TABLE II
PERFORMANCE ON ARGOVERSE BENCHMARK (TEST SET)

Methods	K=1			K=6		
	minADE	*minFDE*	*MR*	*minADE*	*minFDE*	*MR*
Argo [22]	2.96	6.81	0.81	2.34	5.44	0.69
Argo (NN) [22]	3.45	7.88	0.87	1.71	3.29	0.54
Jean [24]	1.86	4.18	0.63	0.93	1.49	0.19
LaneGCN [5]	1.71	3.78	0.59	0.87	1.36	0.16
LaneRCNN [8]	1.68	3.69	0.59	0.90	1.45	0.12
DenseTNT [15]	-	-	-	0.94	1.49	0.11
TNT [7]	2.17	4.96	0.71	0.91	1.44	0.17
GOHOME [14]	-	3.65	0.57	0.94	1.45	0.11
Ours	1.88	4.18	0.65	0.89	1.40	0.17

TABLE III
COMPARISON WITH LANEGCN

Method	#Param	Dataset size	Inference time
LaneGCN	3.7M	20.0G, 3.4G, 8.7G	0.08 sec/batch
Ours	0.6M	7.0G, 1.1G, 3.5G	0.02 sec/batch

On the NuScenes dataset. We compare our method with MTP and CoverNet on the NuScenes dataset and indirectly compare it with GoalNet since both methods are used as baselines in GoalNet. The results are reported in Tab. IV,

where $\min_5$ADE is the minADE in meters over 5 predictions. The data in the first row are reported in GoalNet [6], and the data in the second row are obtained by running MTP, CoverNet, and our method on the NuScenes dataset.

TABLE IV
COMPARISON WITH GOALNET

Methods	MTP	CoverNet	GoalNet	(v.s. MTP)	(v.s. CoverNet)
$\min_5$ADE	2.10	2.44	1.75	-0.35	-0.69
Methods	MTP	CoverNet	**Ours**	(v.s. MTP)	(v.s. CoverNet)
$\min_5$ADE	2.49	2.39	1.70	-0.79	-0.69

C. Ablative Studies

We conduct ablative studies to show the importance of each stage in the HGO and the superiority of the proposed approach compared to a single GAT (G). In Tab. V, r, if checked, means that the model uses RNN (GRU) to encode sequences; each of the graph operators (g1, g2, and g3), if checked, means that the model uses the checked operator(s); G, if checked, means that the model uses a single GNN to encode the entire hierarchical heterogeneous graph. All the ablative models are implemented with the proposed map-adaptive multimodal predictor (III-E). The number of predicted trajectories adapts to the number of CCLs of the target agent. They differ from each other mainly in the encoding part.

The results of the implemented models on the Argoverse validation set are shown in Tab. V. We run each model five times and report the mean and standard deviation of its minADE and minFDE. It can be observed that as we progressively add more graph operators, both minADE and minFDE decrease, and the proposed method shows the most stable performance. This demonstrates the effectiveness of our framework. Besides, there are some more observations we can draw from Tab. V: 1) comparing row 2 and row 3 with row 1, we can see that both the target agent's interaction-awareness (g2) and overall CCL-awareness (g3) improve the performance, respectively; 2) comparing row 4 with row 2, we can observe that the surrounding agents' CCL-awareness (g1) is useful for trajectory prediction; 3) comparing row 5 with row 2 and row 3, we can see that combining the target agent's interaction-awareness (g2) and its overall CCL-awareness (g3) improves the performance; 4) comparing row 5 with row 6, we can observe that our framework with hierarchical graph operator (g2g3) outperforms the model using a single GAT (G) to model a heterogeneous hierarchical graph, despite both having access to the same data in the graph; 5) comparing row 7 with the rest of the rows, we can tell that the proposed hierarchical graph operator outperforms all its ablations. This shows the effectiveness of the proposed HGO.

We find that all three factors (surrounding agents' CCL-awareness, the target agent's interaction-awareness, and its CCL-awareness) positively affect the prediction performance. However, an inappropriate encoder can impair performance. For example, cl-r-G performs worse than cl-r-g2g3 despite using the same input. This is because cl-r-G does not distinguish node and edge types in the heterogeneous interaction graph, while our method can do so via edge-masking. Running both cl-r-g1g2g3 and cl-r-G on the same device, we observe that the former uses 19% more time to make inferences on a batch of 32 data than the latter.

TABLE V
ABLATIVE STUDY RESULTS (ARGOVERSE VALIDATION SET)

Methods	r	g1	g2	g3	G	minADE	minFDE
cl-r	✓					0.883 ± 0.008	1.716 ± 0.017
cl-r-g2	✓		✓			0.827 ± 0.010	1.561 ± 0.019
cl-r-g3	✓			✓		0.838 ± 0.015	1.602 ± 0.033
cl-r-g1g2	✓	✓	✓			0.817 ± 0.017	1.533 ± 0.042
cl-r-g2g3	✓		✓	✓		0.798 ± 0.014	1.487 ± 0.036
cl-r-G	✓				✓	0.817 ± 0.011	1.549 ± 0.029
cl-r-g1g2g3	✓	✓	✓	✓		0.783 ± 0.003	1.448 ± 0.006

D. Visualization

Fig. 5 shows successful prediction cases for all three types, where our method can predict a variable number of trajectories of a target agent according to its CCLs. The predicted trajectories fall into three types: CCL-following, scene-reasoning, and motion-maintaining. The left part of Fig. 5 shows a successful case of the motion-maintaining prediction, where the target vehicle has only one CCL. It can be seen that the motion-maintaining prediction extrapolates its historical track, while the other two predictions are affected by the single CCL. The scene-reasoning and CCL-following predictions are different in this situation because the CCL-following prediction emphasizes the effect of a specific CCL and a shorter CCL may lead to a slow-down in the prediction. The middle part shows a successful case of the scene-reasoning prediction, where the target vehicle has no available CCL. It can be seen that the proposed model can still generate an interaction-aware prediction according to its understanding of the whole scene and the scene-reasoning prediction is closest to the ground truth. The right part shows the successful case of the CCL-following prediction where the target vehicle has two CCLs. It can be seen that CCL-following predictions are aligned with CCLs, and one of them is closest to the ground truth. Three cases in Fig. 5 together show that our method can predict a variable number of trajectories according to different situations (different numbers of CCLs). Fig. 5 also demonstrates the necessity to produce three kinds of predictions (CCL-following, scene-reasoning, and motion-maintaining predictions).

E. Training and Implementation

The implemented models are trained for 50 epochs using Adam as the optimizer to minimize the MTP [3] loss without classification loss. The learning rate of Adam starts from 0.002 and decreases by half at the ends of epochs: 1, 6, 12, 18, 24, and 30. For the proposed model, the raw agent and CCL features are first embedded into a 32-dimensional space separately before being sent to their corresponding GRU encoders. The hidden sizes of both GRU encoders are 64. The GATs in the hierarchical graph operator are single-layer GATs with three attention heads, and their hidden sizes are all set to 128. The GAT in the CCL-enabled prediction header uses only a single attention head. A two-layer MLP is used to finally produce the trajectories. LeakyReLU is used as the activation function between layers.

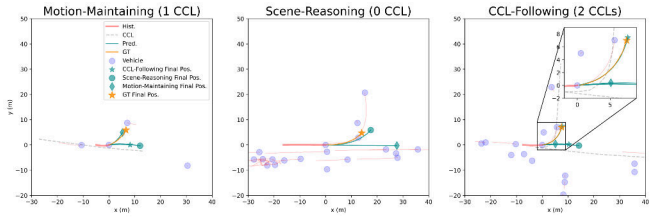

Fig. 5. **Visualized prediction results.** This figure showcases successful predictions of three prediction types. Light blue dots (Vehicle) show the agents' current positions, and red curves show their corresponding historical trajectories (Hist.). Dashed curves (CCL) show the CCLs of the target agent. Yellow curves (GT) and stars (GT Final Pos.) show the target agent's ground truth trajectory and final position over the prediction horizon, respectively. Green stars (CCL-Following Final Pos.) show the final positions of the CCL-following predictions. Green dots (Scene-Reasoning Final Pos.) show that of the scene-reasoning prediction. Green diamonds (Motion-Maintaining Final Pos.) show that of the motion-maintaining predictions. Green curves (Pred.) followed by predicted final positions are their corresponding trajectories.

To use this method in the real world, the historical states of surrounding agents and the target vehicle must be available. This information can be obtained via inter-vehicular communications, onboard perception, or centralized information hubs. For the CCL information, our method requires HD maps (providing lane graphs) of the local area so that the CCLs can be obtained via graph search.

V. Conclusion

In this work, we propose a map-adaptive multimodal trajectory prediction framework that can predict an agent's CCL-following, scene-reasoning, and motion-maintaining trajectories in an integrated manner. We represent the driving scene using a heterogeneous hierarchical graph and design a hierarchical graph operator (HGO) with an edge-masking technology to encode the driving scene. Experiments on real-world datasets show that the proposed framework achieves competitive performance. Compared to LaneGCN, our method reduces the inference time by three quarters. In addition to map-compliant predictions, our method covers the critical corner case where a vehicle's future motion purely depends on its own motion for safety considerations.

Although this method achieves accurate multimodal trajectory prediction, its reliability is affected by the CCL selection strategy. In the future, we plan to develop an adaptive CCL selection strategy and incorporate probability prediction for each possible motion into this framework. We also suggest extending the hierarchical graph operator by adding a stage to capture the interactions among surrounding agents.

References

[1] X. Li, G. Rosman, I. Gilitschenski, C.-I. Vasile, J. A. DeCastro, S. Karaman, and D. Rus, "Vehicle trajectory prediction using generative adversarial network with temporal logic syntax tree features," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3459–3466, 2021.

[2] X. Jia, L. Sun, M. Tomizuka, and W. Zhan, "Ide-net: Interactive driving event and pattern extraction from human data," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3065–3072, 2021.

[3] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2090–2096.

[4] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," in *Conference on Robot Learning*. PMLR, 2020, pp. 86–99.

[5] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *European Conference on Computer Vision*. Springer, 2020, pp. 541–556.

[6] L. Zhang, P.-H. Su, J. Hoang, G. C. Haynes, and M. Marchetti-Bowick, "Map-adaptive goal-based trajectory prediction," in *Conference on Robot Learning*. PMLR, 2021, pp. 1371–1383.

[7] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid *et al.*, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*. PMLR, 2021, pp. 895–904.

[8] W. Zeng, M. Liang, R. Liao, and R. Urtasun, "Lanercnn: Distributed representations for graph-centric motion forecasting," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 532–539.

[9] X. Mo, Z. Chen, and H.-T. Zhang, "Effects of adding a reverse edge across a stem in a directed acyclic graph," *Automatica*, vol. 103, pp. 254–260, 2019.

[10] X. Mo, Y. Xing, and C. Lv, "Recog: A deep learning framework with heterogeneous graph for interaction-aware trajectory prediction," *arXiv preprint arXiv:2012.05032*, 2020.

[11] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[12] 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.

[13] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "Covernet: Multimodal behavior prediction using trajectory sets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 074–14 083.

[14] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "Gohome: Graph-oriented heatmap output for future motion estimation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9107–9114.

[15] J. Gu, C. Sun, and H. Zhao, "Densetnt: End-to-end trajectory prediction from dense goal sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 303–15 312.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *International Conference on Learning Representations*, 2018.

[18] Q. Lv, M. Ding, Q. Liu, Y. Chen, W. Feng, S. He, C. Zhou, J. Jiang, Y. Dong, and J. Tang, "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 1150–1160.

[19] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 000–16 009.

[20] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[21] X. Mo, Y. Xing, and C. Lv, "Graph and recurrent neural network-based vehicle trajectory prediction for highway driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 1934–1939.

[22] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[23] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[24] J. Mercat, T. Gilles, N. El Zoghby, G. Sandou, D. Beauvois, and G. P. Gil, "Multi-head attention for multi-modal joint vehicle motion forecasting," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9638–9644.

2023-04-26

Map-adaptive multimodal trajectory prediction using hierarchical graph neural networks

Mo, Xiaoyu

IEEE

Mo X, Xing Y, Liu H, Lv C. (2023) Map-adaptive multimodal trajectory prediction using hierarchical graph neural networks. IEEE Robotics and Automation Letters, Volume 8, Issue 6, June 2023, pp. 3685-3692

https://doi.org/10.1109/LRA.2023.3270739