PTP: Parallelized Tracking and Prediction with Graph Neural Networks and Diversity Sampling

Xinshuo Weng*, Ye Yuan* and Kris Kitani

***Abstract*—Multi-object tracking (MOT) and trajectory prediction are two critical components in modern 3D perception systems that require accurate modeling of multi-agent interaction. We hypothesize that it is beneficial to unify both tasks under one framework in order to learn a shared feature representation of agent interaction. Furthermore, instead of performing tracking and prediction sequentially which can propagate errors from tracking to prediction, we propose a parallelized framework to mitigate the issue. Also, our parallel track-forecast framework incorporates two additional novel computational units. First, we use a feature interaction technique by introducing Graph Neural Networks (GNNs) to capture the way in which agents interact with one another. The GNN is able to improve discriminative feature learning for MOT association and provide socially-aware contexts for trajectory prediction. Second, we use a diversity sampling function to improve the quality and diversity of our forecasted trajectories. The learned sampling function is trained to efficiently extract a variety of outcomes from a generative trajectory distribution and helps avoid the problem of generating duplicate trajectory samples. We evaluate on KITTI and nuScenes datasets showing that our method with socially-aware feature learning and diversity sampling achieves new state-of-the-art performance on 3D MOT and trajectory prediction. Project website is: http://www.xinshuoweng.com/projects/PTP.**

***Index Terms*—Computer Vision for Transportation; Visual Tracking; Deep Learning for Visual Perception**

I. Introduction

TRACKING and forecasting are critical components in 3D perception systems for autonomous driving [1], [2] and assistive robots [3]–[5]. Historically, 3D multi-object tracking (MOT) [6]–[9] and trajectory forecasting [10]–[18] have been studied separately. As a result, perception systems often perform 3D MOT and forecasting separately in a cascaded order, where tracking is performed first to obtain past trajectories, followed by trajectory forecasting to predict future trajectories. However, this cascaded pipeline with separately trained modules can lead to sub-optimal performance, as information is not shared across two modules during training. As tracking and prediction are mutually dependent, it would be beneficial to optimize them jointly. For example, a better MOT module can lead to better performance of its downstream forecasting module while a more accurate motion model learned by prediction can improve data association in MOT. Our goal is to jointly optimize MOT and forecasting modules and learn a better shared feature representation for both modules.

Manuscript received: October 15th, 2020; Revised January 13, 2021; Accepted March 1, 2021.

This paper was recommended for publication by Editor Cesar Cadena Lerma upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by Qualcomm.

*The first two authors contributed equally to this work.

All three authors are with Robotics Institute, Carnegie Mellon University. {xinshuow, yyuan2, kkitani}@cs.cmu.edu.

Digital Object Identifier (DOI): see top of this page.

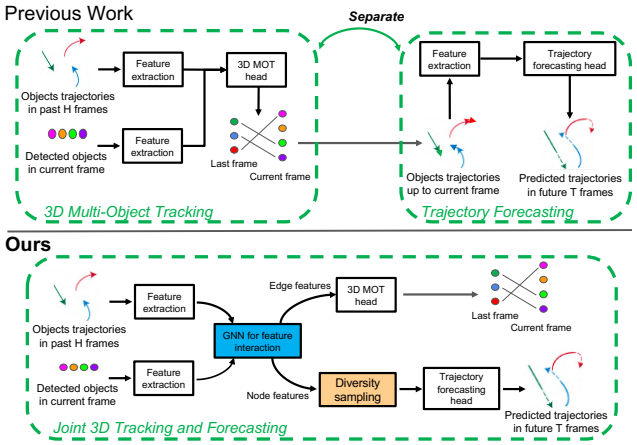


Fig. 1. **(Top)** Previous work: 3D MOT and trajectory forecasting performed separately and connected as a sequential process. **(Bottom)** Ours: Joint and parallelized framework for MOT and forecasting. Two key innovations: (1) feature interaction using GNNs to obtain socially-aware features in the presence of multiple agents; (2) diversity sampling to improve sample efficiency and produce diverse and accurate trajectory samples.

In addition to joint optimization, we also propose to parallelize MOT and forecasting in our framework. Instead of performing two modules in a sequential order as shown in Fig. 1 (top), our framework performs MOT and forecasting in *parallel* as shown in Fig. 1 (bottom). By parallelizing MOT and forecasting heads, i.e., forecasting does not explicitly depend on the MOT results, we can prevent association errors made in MOT from degrading forecasting performance. One might argue that the forecasting module in our parallelized framework cannot utilize association information in the current frame. However, we believe that the forecasting module in our framework can implicitly use association information of the current frame encoded in the shared features. We will show in the experiments that our novel parallelized framework outperforms prior cascaded track-forecast framework.

Modeling interaction for 3D MOT is crucial in the presence of multiple agents but is often overlooked in prior work. Prior work in 3D MOT often extracts the feature of each object *independently*, i.e., each object's feature only depends on the object's own inputs (image crop or location). As a result, there is no interaction between objects. We found that *independent feature extraction leads to inferior discriminative feature learning*, and object dependency is the key to obtaining discriminative features. Intuitively, the features of the same object over two frames should be as similar as possible and the features of two different objects should be as different as possible to avoid confusion during data association. This can only be achieved if object features are obtained in a context-aware process, i.e., modeling interactions between objects.

To model object interaction in 3D MOT, we use a feature interaction mechanism as shown in Fig. 1 (Bottom) by in-

troducing Graph Neural Networks (GNNs). Specifically, we construct a graph with each node being an object in the scene. Then, at each layer of the GNNs, each node can update its feature by aggregating features from other nodes. This node feature aggregation is useful because the resulting object features are no longer isolated and can be adapted according to other objects. We observed in our experiments that, after a few GNN layers, affinity matrix becomes more discriminative than the affinity matrix obtained without interaction. In addition to modeling interaction to improve 3D MOT, interaction modeling can also provide socially-aware context to improve trajectory forecasting [12], [17], [19]. To the best of our knowledge, we are the first to use GNNs to model interaction in a unified framework for both 3D MOT and trajectory forecasting tasks.

As future trajectories of objects are uncertain and multi-modal due to unobserved factors such as intent, prior work in trajectory forecasting often learns the future trajectory distribution with generative models such as conditional variational autoencoders (CVAEs; [14]) and conditional generative networks (CGANs; [11]). At test time, these methods randomly sample a set of future trajectories from the generative model without considering the correlation between samples. As a result, the samples can be very similar and only cover a limited number of modes, leading to poor sample efficiency. This inefficient sampling strategy is harmful in real-time applications because producing a large number of samples can be computationally expensive and lead to high latency. Moreover, without covering all the modes in the trajectory distribution and considering all possible futures, the perception system cannot plan safely, which is detrimental to safety-critical applications such as autonomous driving.

To improve sample efficiency in trajectory forecasting, we depart from the random sampling in prior work and employ a diversity sampling technique that can generate accurate and diverse trajectory samples from a pretrained CVAE model. The idea is to learn a separate sampling network which maps each object's feature to a set of latent codes. The latent codes are then decoded into trajectory samples. In this way, the produced samples are correlated (unlike random sampling where the samples are independent), which allows us to enforce structural constraints such as diversity onto the samples. Specifically, we use determinantal point processes (DPPs; [20]) to optimize the diversity of the samples. Our contributions are summarized as follows:

1) A parallelized framework for 3D MOT and trajectory forecasting to avoid compounding errors and improve performance of both modules via joint optimization;
2) A GNNs-based feature interaction mechanism that is the first applied to a unified MOT and forecasting framework to improve socially-aware feature learning;
3) Introducing diversity sampling to multi-agent trajectory forecasting which can produce more accurate and diverse trajectory samples.

II. Related Work

3D Multi-Object Tracking. Recent work tackles 3D MOT in an online fashion using a tracking-by-detection pipeline, where performance is mainly affected by two factors: 3D detection quality and discriminative feature learning. To obtain discriminative features, prior work focuses on feature engineering, among which the motion and appearance features are the most popular ones. [8], [21], [22] use Convolutional Neural Networks (CNNs) to extract 2D appearance features. To learn 3D appearance features from point clouds, [7] proposes a PointNet-based [23] 3D MOT network. To leverage motion features, filter-based [6], [24] and learning-based methods [21] have been proposed. Although prior work has achieved impressive performance by feature engineering, they extract feature from each object independently and ignore object interactions. Different from prior work, [25] is the first introducing GNNs to model interaction in 3D MOT, which significantly improve discriminative feature learning. Different from [25] which only shows the success of introducing GNNs to 3D MOT, we embed GNNs in a unified 3D MOT and trajectory forecasting framework to improve feature learning for both tasks.

Trajectory Prediction is to predict a sequence of ground positions of target objects in the future. Prior work mostly investigates the target object of people [10]–[13], [26]–[28] and vehicles [14]–[17], [29]–[31]. As the future is multi-modal, [11]–[13] use probabilistic models for trajectory prediction. Also, as agent behavior can be influenced by others, [12], [13], [17] introduce GNNs to learn interaction-aware features. However, most prior works study trajectory forecasting separately from the highly-related 3D MOT task, while we consider both forecasting and tracking in a unified framework.

Joint 3D Detection, Tracking and Prediction. A few prior works attempt joint optimization for different combinations of the three modules. [8], [22], [32] jointly optimize detector and tracker. [33], [34] achieve joint detection and prediction, while skipping the tracking module. [35] shows results for detection, tracking and forecasting. However, similar to [33], [34], only detection and prediction are jointly optimized in [35], while tracking results are obtained by post-processing. Perhaps concurrent work [36] is the closest to us which also jointly optimizes detection, tracking and forecasting. However, same as prior work, [36] processes three modules in a sequential order. To the best of our knowledge, we are the first to propose a parallelized tracking and prediction framework to avoid compounding errors, which also has joint optimization of tracking and prediction modules.

Graph Neural Networks was proposed in [37] to process graph-structured data using neural networks. The primary component of GNNs is node feature aggregation, with which the feature of a node can be updated by interacting with other nodes. Recently, significant success has been achieved by introducing GNNs to applications such as semantic segmentation [38], [39], action recognition [40]–[43], object tracking [25], [44]. Inspired by prior work, the goal of this work is to introduce existing GNNs techniques to a different practical application – joint MOT and trajectory forecasting, especially in our novel parallelized MOT and forecasting framework.

Diversity Sampling. Stemming from the M-Best MAP problem [45], diverse M-Best solutions [46] and multiple choice learning [47] are able to produce a diverse ensemble of so-

lutions and models. Also, submodular function maximization [48] has been used for diverse selection of garments from fashion images. Determinantal point processes (DPPs) [20] are also popular probabilistic models for subset selection due to its ability to measure the global diversity and quality within a set. Prior work has applied DPPs for document and video summarization [49], object detection [50], and grasp clustering [51]. Sample diversity has also been an active research topic in generative modeling. A majority of this line of research aims to improve the diversity of the data distribution learned by deep generative models, including works that try to alleviate the mode collapse problem in GANs [52]–[55] and the posterior collapse problem in VAEs [56]–[58]. Recent work [59], [60] uses DPPs to improve sample diversity in single-agent trajectory prediction and evaluated on a toy dataset. Different from [59], [60], we apply diversity sampling to multi-agent trajectory forecasting and evaluate on real large-scale driving datasets.

III. Approach

We aim to achieve 3D MOT and bird's eye view trajectory forecasting in parallel. Let $\mathcal{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_M\}$ denote the set of past trajectories of M tracked objects. Each past trajectory $\boldsymbol{o}_i = [\boldsymbol{o}_i^{-H}, \ldots, \boldsymbol{o}_i^{-1}]$ consists of the associated detections of the i-th tracked object in the past H frames. The associated detection at frame $t \in \{-H, \ldots, -1\}$ is a tuple $\boldsymbol{o}_i^t = [x, y, z, l, w, h, \theta, I]$, where (x, y, z) denotes the object center in 3D space, (l, w, h) denotes the object size, θ is the heading angle, and I is the assigned ID. Let $\mathcal{D} = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N\}$ denote the set of unassociated detections of N objects in the current frame obtained by a 3D object detector. Each unassociated detection $\boldsymbol{d}_j = [x, y, z, l, w, h, \theta]$ is defined similarly to the past associated detections $\boldsymbol{o}_i^t$ except without the assigned ID I. The goal of 3D MOT is to associate the current detection $\boldsymbol{d}_j \in \mathcal{D}$ with the past object trajectory $\boldsymbol{o}_i \in \mathcal{O}$ and assign an ID to $\boldsymbol{d}_j$. For trajectory forecasting, the objective is to predict the future trajectories $\mathcal{F} = \{\boldsymbol{f}_i, \ldots, \boldsymbol{f}_M\}$ for all M tracked objects in the past. Each future trajectory $\boldsymbol{f}_i = [\boldsymbol{f}_i^1, \ldots, \boldsymbol{f}_i^T]$ consists of the x and z positions (i.e., 2D position on the ground in a top-down view) of the i-th object in future T frames, i.e., $\boldsymbol{f}_i^t = [x, z]$ where $t \in \{1, \ldots, T\}$.

The entire network of our method to achieve the parallelized MOT and forecasting is shown in Fig. 1 (bottom), which consists of five modules: (1) a feature extractor to encode the feature for the object trajectories in the past and the detections in the current frame; (2) a feature interaction mechanism using GNNs to update the object features based on the features of other objects; (3) a 3D MOT head that computes the affinity matrix for data association between the tracked objects in the past and detected objects in the current frame; (4) a trajectory forecasting head that learns a CVAE to generate future trajectories based on the GNN features and past trajectories; (5) a diversity sampling that can optimize the diversity of the trajectory samples.

A. Feature Extraction

To utilize motion and location information from object trajectories in the past and detections in the current frame, we learn feature extractors as shown in Fig. 2 (Left). Given the trajectory $\boldsymbol{o}_i = [\boldsymbol{o}_i^{-H}, \ldots, \boldsymbol{o}_i^{-1}]$ for a tracked object i, we obtain its feature by applying a two-layer LSTM. The LSTM aims to model the temporal dynamics in the data and outputs a feature $\boldsymbol{u}_i$ with 64 dimensions. For a detected object j in the current frame, we use a 2-layer Multi-Layer Perceptron (MLP) to map the detection $\boldsymbol{d}_j$ to a 64-dimensional feature $\boldsymbol{v}_j$. Note that the feature extractors for tracked object $\boldsymbol{o}_i$ and detected object $\boldsymbol{d}_j$ are different as $\boldsymbol{o}_i$ and $\boldsymbol{d}_j$ have different time horizon. The obtained features $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are then used as initial node features $\boldsymbol{u}_i^0$ and $\boldsymbol{v}_j^0$ at layer 0 of the GNNs (see Sec. III-B) for feature interaction.

Note that our feature extractor is shared and optimized for both tracking and forecasting, which is different from prior work that extracts features twice separately in 3D MOT and trajectory forecasting as shown in Fig. 1 (top). As a result, our method reduces system complexity and we will also show in experiments that our parallelized tracking and forecasting framework improves feature learning.

B. Graph Neural Network for Feature Interaction

Graph Construction. After feature extraction, we have M features $\{\boldsymbol{u}_1^0, \ldots, \boldsymbol{u}_M^0\}$ for tracked objects in the past and N features $\{\boldsymbol{v}_1^0, \ldots, \boldsymbol{v}_N^0\}$ for detected objects in the current frame. We then construct an L-layer undirected Graph Neural Network (GNN) where each layer includes nodes of the M tracked objects and N currently detected objects as shown in Fig. 2 (right). We use undirected graph because the interaction between objects should be mutual. As the node feature is updated at each layer, let us denote the node features for the tracked objects at layer l as $\mathcal{U}^l = \{\boldsymbol{u}_1^l, \ldots, \boldsymbol{u}_M^l\}$. Similarly, we define the node feature for the currently detected objects at layer l as $\mathcal{V}^l = \{\boldsymbol{v}_1^l, \ldots, \boldsymbol{v}_N^l\}$.

In addition to node feature definition, we also define a set of edges at every layer of the graph to relate node features. To make GNN learning efficient, we restrict edge connections to be sparse, i.e., edges are not defined between every pair of nodes. Specifically, we utilize prior knowledge about social interaction in the presence of multiple agents: interactions primarily happen between objects that are close to each other. Therefore, we construct the edge between two nodes if and only if these two nodes' box centers have distance less than a threshold (C meters) in 3D space. As a result, we have a sparse edge connection as shown in Fig. 2 (right). Note that the edge connections are dynamic across time so that GNNs can model interaction in different scenes with varying numbers of objects, though the edge connections are fixed across layers of GNNs at the same time step.

Node Feature Aggregation. To model feature interaction in GNN, we iteratively update the node features by aggregating features from the neighborhood nodes (i.e., nodes connected by an edge) in each layer. Specifically, we employ the node feature aggregation rule proposed in GraphConv [61]:

$$\boldsymbol{u}_i^{l+1} = \sigma_1^l(\boldsymbol{u}_i^l) + \sum\nolimits_{j \in \mathcal{N}(i)} \sigma_2^l(\boldsymbol{v}_j^l) + \sum\nolimits_{g \in \mathcal{N}(i)} \sigma_3^l(\boldsymbol{u}_g^l), \quad (1)$$

where $\boldsymbol{u}_i^l$ and $\boldsymbol{u}_i^{l+1}$ are the node features for a tracked object i at layer l and $l+1$. $\mathcal{N}(i)$ denotes a set of neighborhood

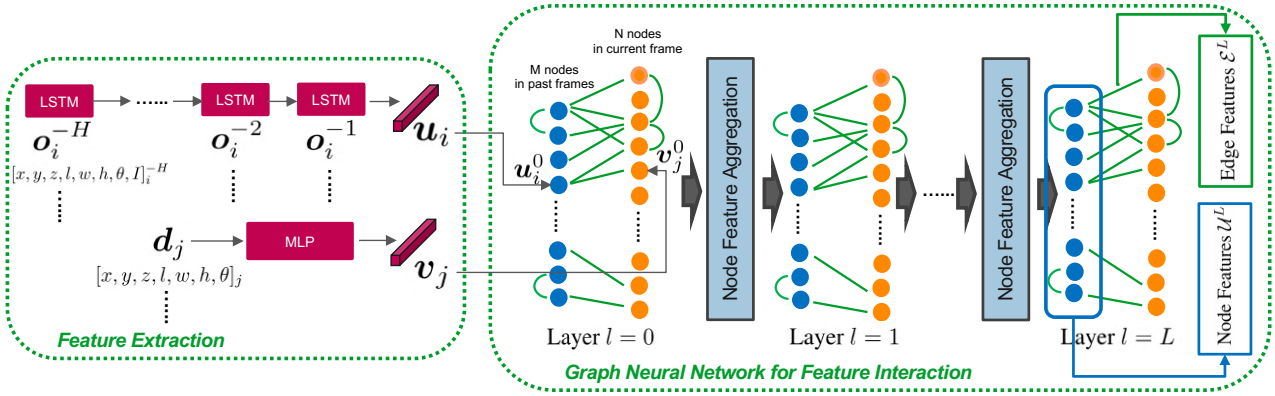


Fig. 2. **(Left)** To leverage the location and motion cues, we extract the feature from object trajectories $\mathcal{O}=\{\boldsymbol{o}_1,\ldots,\boldsymbol{o}_M\}$ in the past using an LSTM model and extract the feature from detections $\mathcal{D}=\{\boldsymbol{d}_1,\ldots,\boldsymbol{d}_N\}$ in the current frame using a MLP. **(Right)** The GNN-based feature interaction mechanism is used to update object node feature $\mathcal{U}^l=\{\boldsymbol{u}_1^l,\ldots,\boldsymbol{u}_M^l\}$ and $\mathcal{V}^l=\{\boldsymbol{v}_1^l,\ldots,\boldsymbol{v}_N^l\}$ at GNN layer l and iteratively through all GNN layers. At the final layer, we use the node features for tracked objects $\mathcal{U}^L$ for 3D MOT task (see Sec. III-C), and use the edge features $\mathcal{E}^L$ (computed from $\mathcal{U}^L$ and $\mathcal{V}^L$) for trajectory forecasting task (see Sec. III-D and III-E).

nodes that are connected to the node i by an edge, and $\boldsymbol{v}_j^l$, $\boldsymbol{u}_g^l$ with $j,g\in\mathcal{N}(i)$ are the neighborhood node features at layer l, where $\boldsymbol{u}_g^l$ is the neighborhood node feature in the past frames and $\boldsymbol{v}_j^l$ is the neighborhood node feature in the current frame. Moreover, σ_1^l, σ_2^l, σ_3^l are linear layers at layer l, whose weights are not shared across layers. Note that a ReLU operator is applied to the node feature after feature aggregation at each layer except for the final layer. Intuitively, the above node aggregation rule means that each node feature is updated by aggregating the transformed features of its own and its connected nodes. In addition to updating the node feature $\boldsymbol{u}_i^l$ for tracked objects, we also update the node feature $\boldsymbol{v}_j^l$ for detected objects:

$$\boldsymbol{v}_j^{l+1}=\sigma_1^l(\boldsymbol{v}_j^l)+\sum\nolimits_{i\in\mathcal{N}(j)}\sigma_2^l(\boldsymbol{u}_i^l)+\sum\nolimits_{g\in\mathcal{N}(j)}\sigma_3^l(\boldsymbol{v}_g^l). \quad (2)$$

Based on the above rules, the updated node features for tracked objects $\mathcal{U}^{l+1}$ and for detected objects $\mathcal{V}^{l+1}$ will affect each other through feature interaction in the following layers. After several layers of feature interaction, we use the node features at the final layer L for tracked objects $\mathcal{U}^L$ as inputs to our forecasting head (see section III-D). Due to feature interaction, we believe that the final node features $\mathcal{U}^L$ have contained enough information from both the trajectories in the past frames and detections in the current frame.

Edge Feature. As each entry of the affinity matrix in MOT represents similarity of two objects, it is natural to use edges relating two object nodes to compute the affinity matrix. To learn the similarity, we first define the edge feature between two connected nodes as the difference of their node features:

$$\boldsymbol{e}_{ij}^l=\boldsymbol{u}_i^l-\boldsymbol{v}_j^l, \quad (3)$$

where $\boldsymbol{u}_i^l$ is the node feature of tracked object i in the past frames and $\boldsymbol{v}_j^l$ is the node feature of detected object j in the current frame. The two features are related by an edge feature $\boldsymbol{e}_{ij}^l$ at layer l. Note that we only compute the feature for edges relating a tracked object and a detected object as MOT only associates objects across frames (not objects in a same frame). We use the set of edge features $\mathcal{E}^L$ at the final GNN layer as inputs to 3D MOT head for data association. We will show in our experiments that using this simple subtraction between two node features as the edge feature is good enough to achieve S.O.T.A 3D MOT performance.

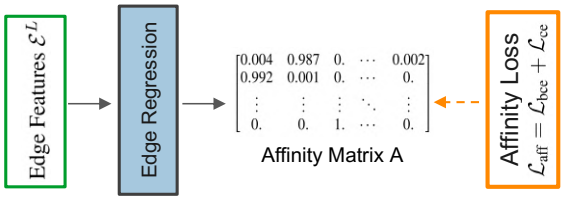


Fig. 3. 3D multi-object tracking head with an affinity loss.

C. 3D Multi-Object Tracking Head

To solve 3D MOT, we need to learn an affinity matrix A based on pairwise similarity of the features extracted from M tracked objects in the past and N detected objects in the current frame. As a result, affinity matrix A has a dimension of $M\times N$ where each entry A_{ij} represents the similarity score between the tracked object i and the detected object j.

Edge Regression. To learn the affinity matrix for data association, we employ an edge regression module as shown in Fig. 3, which consists of a two-layer MLP with a non-linear operator and a Sigmoid layer. To compute each entry in A_{ij}, the edge regression module uses an edge feature $\boldsymbol{e}_{ij}^L$ as input and outputs a scalar value between 0 to 1 as the pairwise similarity score:

$$A_{ij}=\text{Sigmoid}(\sigma_4(\text{ReLU}(\sigma_3(\boldsymbol{e}_{ij}^L)))), \quad (4)$$

where σ_3 and σ_4 are two linear layers. As a result, the computed affinity matrix A can be used to associate the objects using the Hungarian algorithm [62] during testing. For tracked and detected objects that cannot be associated, we employ the same birth and death memory as in [6] to create and delete identities. During training, we learn the network parameters by computing an affinity loss between the estimated affinity matrix A and its ground truth (GT).

Affinity Loss. As shown in Fig. 3, we employ an affinity loss $\mathcal{L}_{\text{aff}}$ to directly supervise the output A of 3D MOT head. Our affinity loss consists of two individual losses. First, as we know that the GT affinity matrix A^g can only have integer 0 or 1 on all the entries, we can formulate the prediction of the affinity matrix as a binary classification problem. Therefore,

our first loss is the binary cross entropy (BCE) loss $\mathcal{L}_{\text{bce}}$ that is applied on every entry of A:

$$\mathcal{L}_{\text{bce}} = \frac{-1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} A_{ij}^{g}\log A_{ij} + (1-A_{ij}^{g})\log(1-A_{ij}). \quad (5)$$

Second, we know that each tracked object $\boldsymbol{o}_i$ can only have either one matched detection $\boldsymbol{d}_j$ or no match at all. In other words, each row and column of the A^g can only be a one-hot vector or an all-zero vector. This motivates our second loss. We define the set of rows and columns in A^g that have a one-hot vector as $\mathcal{M}_{oh}$ and $\mathcal{N}_{oh}$, and then apply the cross entropy (CE) loss $\mathcal{L}_{\text{ce}}$ to these rows and columns. As an example, if the jth column $A^g_{\cdot j}$ in GT affinity matrix is a one-hot vector, then the loss $\mathcal{L}_{\text{ce}}$ for the jth column is defined as:

$$\mathcal{L}_{\text{ce}}^{\cdot j} = -\frac{1}{M}\sum_{i=1}^{M} A_{ij}^{g}\log\left(\frac{\exp A_{ij}}{\sum_{i=1}^{M}\exp A_{ij}}\right). \quad (6)$$

We can summarize the affinity loss $\mathcal{L}_{\text{aff}}$ for 3D MOT:

$$\mathcal{L}_{\text{aff}} = \mathcal{L}_{\text{bce}} + \mathcal{L}_{\text{ce}} = \mathcal{L}_{\text{bce}} + \sum_{i\in\mathcal{M}_{oh}}\mathcal{L}_{\text{ce}}^{i\cdot} + \sum_{j\in\mathcal{N}_{oh}}\mathcal{L}_{\text{ce}}^{\cdot j}, \quad (7)$$

where $\mathcal{L}_{\text{ce}}$ is computed by summing over the Eq. 6 for all rows and columns with a one-hot vector. Also, we use the same weight of 1 for two losses $\mathcal{L}_{\text{ce}}$ and $\mathcal{L}_{\text{bce}}$.

D. Trajectory Forecasting Head

Our trajectory forecasting head is a conditional generative model $p_\theta(\boldsymbol{f}_i|\boldsymbol{o}_i, \boldsymbol{u}_i^L)$, which learns the distribution of the i-th tracked object's future trajectory $\boldsymbol{f}_i$ based on its past trajectory $\boldsymbol{o}_i$ and node feature $\boldsymbol{u}_i^L$ at the last GNN layer. Note that the trajectory forecasting head does not explicitly depend on the MOT association results in the current frame, but instead uses the node feature $\boldsymbol{u}_i^L$ after feature interaction. This design prevents the association error in the current frame made by MOT from deteriorating the forecasting results, while still allowing the forecasting module to exploit the information in the current frame. This is because the node feature $\boldsymbol{u}_i^L$ already encodes object information in the current frame through interaction. As we share the generative model for all tracked objects $\mathcal{O}$, we drop the subscripts and superscripts for ease of notation and denote the generative model as $p_\theta(\boldsymbol{f}|\boldsymbol{o}, \boldsymbol{u})$. We adopt the CVAE [14] as our generative model and introduce a latent variable $\boldsymbol{z}$ to model unobserved factors (e.g., agent intentions) and capture the multi-modal distribution of the future trajectory $\boldsymbol{f}$. Based on the CVAE formulation, we introduce a variational lower bound $\mathcal{V}_{lb}(\boldsymbol{f}; \theta, \phi)$ of the log-likelihood function $\log p_\theta(\boldsymbol{f}|\boldsymbol{o}, \boldsymbol{u})$:

$$\begin{aligned}\mathcal{V}_{lb}(\boldsymbol{f}; \theta, \phi) = \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{f},\boldsymbol{o},\boldsymbol{u})}\left[\log p_\theta(\boldsymbol{f}|\boldsymbol{z}, \boldsymbol{o}, \boldsymbol{u})\right] \\ - \mathrm{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{f}, \boldsymbol{o}, \boldsymbol{u})\|p(\boldsymbol{z})\right),\end{aligned} \quad (8)$$

where $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a Gaussian latent prior, $q_\phi(\boldsymbol{z}|\boldsymbol{f}, \boldsymbol{o}, \boldsymbol{u}) = \mathcal{N}(\boldsymbol{\mu}, \mathrm{Diag}(\boldsymbol{\sigma}^2))$ is an approximated posterior (encoder distribution) and $p_\theta(\boldsymbol{f}|\boldsymbol{z}, \boldsymbol{o}, \boldsymbol{u}) = \mathcal{N}(\tilde{\boldsymbol{f}}, \alpha\boldsymbol{I})$ is a conditional likelihood (decoder distribution) with a coefficient α. We use two Recurrent Neural Networks (RNNs) as the encoder F_ϕ and decoder G_θ to respectively output the parameters of the encoder and decoder distributions: $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = F_\phi(\boldsymbol{f}, \boldsymbol{o}, \boldsymbol{u})$ and $\tilde{\boldsymbol{f}} = G_\theta(\boldsymbol{z}, \boldsymbol{o}, \boldsymbol{u})$. The detailed architectures for F_ϕ and G_θ are given in the supplementary materials. Based on above formulation, the loss for our trajectory forecasting head is $\mathcal{L}_{\text{cvae}} = -\mathcal{V}_{lb}$. As we jointly optimize the tracking and forecasting heads as well as the feature extractors and GNNs, we summarize the overall loss of our network as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{aff}} + \mathcal{L}_{\text{cvae}} = \mathcal{L}_{\text{aff}} - \mathcal{V}_{lb}, \quad (9)$$

Where the weights are 1 for $\mathcal{L}_{\text{aff}}$ and $\mathcal{L}_{\text{cvae}}$. Once the CVAE model is learned, we can produce the i-th agent's future trajectories $\boldsymbol{f}_i$ by randomly sampling a set of latent codes $\{\boldsymbol{z}_{i1}, \ldots, \boldsymbol{z}_{iK}\}$ from the latent prior and decode them using the decoder G_θ into future trajectory samples $\{\boldsymbol{f}_{i1}, \ldots, \boldsymbol{f}_{iK}\}$. However, as random sampling can lead to similar samples and low sample efficiency, we introduce a diversity sampling technique into multi-agent trajectory forecasting that can produce diverse and accurate samples, and improve sample efficiency.

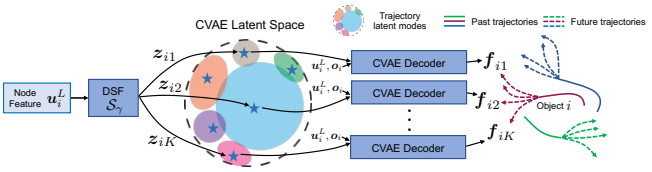


Fig. 4. **Trajectory Forecasting with Diversity Sampling.** To produce diverse trajectory samples from our pretrained CVAE model, we learn a diversity sampling function (DSF) $\mathcal{S}_\gamma$ to map each object's node feature $\boldsymbol{u}_i^L$ to a set of latent codes, which can cover not only the major mode but also other modes in the CVAE latent space. Then, we decode those codes into diverse and accurate future trajectories for object i.

E. Diversity Sampling Technique

To obtain diverse future trajectory samples for multi-agent trajectory forecasting, we introduce the diversity sampling technique. As shown in Fig. 4, we use a γ-parameterized Diversity Sampling Function (DSF) $\mathcal{S}_\gamma$ (a two-layer MLP) that maps the i-th object's GNN feature $\boldsymbol{u}_i^L$ to a set of latent codes: $\mathcal{S}_\gamma(\boldsymbol{u}_i^L) = \{\boldsymbol{z}_{i1}, \ldots, \boldsymbol{z}_{iK}\}$. We can then use the CVAE decoder G_θ to decode the latent codes into a set of future trajectories $\mathcal{Y}_i = \{\boldsymbol{f}_{i1}, \ldots, \boldsymbol{f}_{iK}\}$ for object i. In this way, the latent codes and trajectory samples are correlated and can be controlled by the parameters of the DSF $\mathcal{S}_\gamma$. Our goal is to optimize $\mathcal{S}_\gamma$ so that the trajectory samples $\mathcal{Y}_i$ are both diverse and accurate. To model diversity, we construct a DPP kernel $\boldsymbol{L}^i \in \mathbb{R}^{K\times K}$ for each object i based on the diversity and quality of the samples in $\mathcal{Y}_i$:

$$\boldsymbol{L}^i = \mathrm{Diag}(\boldsymbol{r}^i)\cdot\boldsymbol{S}^i\cdot\mathrm{Diag}(\boldsymbol{r}^i), \quad (10)$$

where the DPP kernel is formed by two components representing sample diversity and quality — a similarity matrix $\boldsymbol{S}^i \in \mathbb{R}^{K\times K}$ and a quality vector $\boldsymbol{r}^i \in \mathbb{R}^K$:

$$\begin{aligned}\boldsymbol{S}^i_{ab} &= \exp\left(-\omega\|\boldsymbol{f}_{ia} - \boldsymbol{f}_{ib}\|^2\right), \\ \boldsymbol{r}^i_k &= \exp\left(\max(-\|\boldsymbol{z}_{ik}\|^2 + R^2, 0)\right).\end{aligned} \quad (11)$$

The element $\boldsymbol{S}^i_{ab}$ of the similarity matrix measures similarity between trajectory samples $\boldsymbol{f}_{ia}$ and $\boldsymbol{f}_{ib}$ with a Gaussian kernel where ω is a scaling factor. Each element $\boldsymbol{r}^i_a$ in the quality vector defines the quality of sample $\boldsymbol{f}_{ia}$ based on how far its latent code $\boldsymbol{z}_{ik}$ is from the origin. If $\boldsymbol{z}_{ik}$ is very far, it means the sample has low likelihood and will be assigned a low quality score. Based on the DPP kernel $\boldsymbol{L}^i$, one can define a diversity loss to measure the diversity and quality within the trajectory samples $\mathcal{Y}_i$:

$$\mathcal{L}^i_{\text{dpp}} = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} = -\text{tr}\left(\boldsymbol{I} - (\boldsymbol{L}^i + \boldsymbol{I})^{-1}\right), \tag{12}$$

where λ_n is the n-th eigenvalue of $\boldsymbol{L}^i$, $\text{tr}(\cdot)$ is the trace operator and $\boldsymbol{I}$ is the identity matrix. As the diagonal elements of $\boldsymbol{L}^i$ are all ones, the sum of eigenvalues is fixed: $\sum \lambda_n = \text{tr}(\boldsymbol{L}^i) = K$. The optima of $\mathcal{L}^i_{\text{dpp}}$ is obtained when all eigenvalues are equal and $\boldsymbol{L}^i$ becomes an identity matrix, thus making $\boldsymbol{S}^i_{ab} = 0$ $(a \neq b)$ and $\boldsymbol{r}^i_k = 1$. This means the distance between trajectory samples is large and each sample has high likelihood. However, this optima is seldom obtained due to the trade-off between diversity and quality, i.e., samples far way from others often have low likelihood. Besides the diversity loss, we further introduce a reconstruction loss to encourage the set of trajectory samples $\mathcal{Y}_i$ to cover the ground-truth future trajectories $\hat{\boldsymbol{f}}$:

$$\mathcal{L}^i_{\text{recon}} = \min_k \|\boldsymbol{f}_{ik} - \hat{\boldsymbol{f}}\|^2. \tag{13}$$

To learn DSF, we freeze the parameters of other components (feature extractors, GNNs, and CVAE) and only optimize the parameters γ of the DSF with the following loss:

$$\mathcal{L}_{\text{dsf}} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{L}^i_{\text{dpp}} + \mathcal{L}^i_{\text{recon}}. \tag{14}$$

IV. Experiments

A. Datasets

We evaluate on standard driving datasets: KITTI [63] and nuScenes [64], which provide LiDAR point cloud and 3D bounding box trajectories. We do not evaluate on 2D MOT datasets such as MOTChallenges [65] as they do not provide LiDAR data or 3D bounding box ground truth and are thus not directly applicable to our method. To compare against prior state-of-the-art methods for 3D MOT alone or for trajectory forecasting only, we evaluate two modules separately. For KITTI, same as prior work, we report results on the car subset for comparison. For nuScenes, we evaluate our 3D MOT and trajectory forecasting for all categories (car, pedestrian, bicycle, motorcycle, truck, bus and trailer) and final performance is the mean over all categories.

B. Evaluating 3D Multi-Object Tracking

Evaluation Metrics. We use standard CLEAR metrics (including MOTA, MOTP, IDS) and new sAMOTA, AMOTA and AMOTP metrics [6]. Since we are evaluating 3D MOT methods, all above metrics need to be defined in 3D space using the criteria of 3D IoU or 3D distance. However, KITTI dataset only supports 2D MOT evaluation, i.e., metrics defined in 2D space for evaluating image-based MOT methods. Therefore, instead of using KITTI 2D MOT evaluation, we use 3D MOT evaluation code provided by [6].

Baselines. We compare against recent 3D MOT systems such as FANTrack [21], mmMOT [7], AB3DMOT [6] and GNN3DMOT [25]. To achieve fair comparison, we use the same 3D detections obtained by PointRCNN [66] on KITTI and by Megvii [67] on nuScenes for all methods. Also, for some baselines [21], [22], [25] that require 2D detections as inputs, we use the 2D projection of the 3D detections.

TABLE I
3D MOT Evaluation on the KITTI and nuScenes datasets.

Datasets	Methods	**sAMOTA**(%)↑	AMOTA(%)↑	AMOTP(%)↑	MOTA(%)↑	MOTP(%)↑	IDS↓
KITTI	mmMOT [7]	70.61	33.08	72.45	74.07	78.16	10
	FANTrack [21]	82.97	40.03	75.01	74.30	75.24	35
	AB3DMOT [6]	93.28	45.43	77.41	86.24	78.43	**0**
	GNN3DMOT [25]	93.68	45.27	**78.10**	84.70	**79.03**	**0**
	Ours (PTP)	**94.41**	**46.15**	76.83	**86.89**	78.32	3
nuScenes	FANTrack [21]	19.64	2.36	22.92	18.60	39.82	1593
	mmMOT [7]	23.93	2.11	21.28	19.82	40.93	572
	GNN3DMOT [25]	29.84	6.21	24.02	23.53	46.91	**401**
	AB3DMOT [6]	39.90	8.94	29.67	31.40	57.54	751
	Ours (PTP)	**42.36**	**9.84**	**33.48**	**32.06**	**63.61**	809

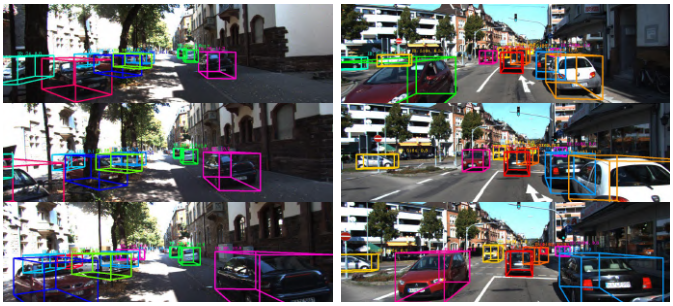
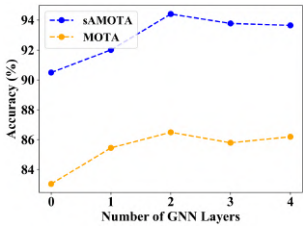


Fig. 5. 3D MOT results on sequence 0 (left) and 11 (right) of the KITTI test set. We show three frames (with an interval of 5 frames) for each sequence.

TABLE II
Effect of trajectory forecasting on 3D MOT.

Metrics	w/o forecasting	w/ forecasting
sAMOTA(%)↑	91.31	**94.41**
AMOTA(%)↑	43.68	**46.15**
AMOTP(%)↑	**76.94**	76.83
MOTA(%)↑	83.51	**86.89**
MOTP(%)↑	78.11	**78.32**
IDS↓	5	**3**



Results. We summarize the 3D MOT results on KITTI and nuScenes datasets in Table I. Our method consistently outperforms baselines in sAMOTA, AMOTA and MOTA, which are the primary metrics for ranking MOT methods. We hypothesize that this is because our method leveraging GNN obtains more discriminative features to avoid confusion in MOT association while all 3D MOT baselines ignore the interaction between objects. Moreover, joint optimization of the tracking and forecasting modules in parallel also helps. We will justify both hypotheses in the ablation study. We show qualitative results of our method on the KITTI dataset in Fig. 5, demonstrating reliable 3D MOT performance.

Ablation Study. We first verify if joint optimization of the 3D MOT and forecasting improves 3D MOT performance. In Table II, when we train MOT and trajectory forecasting heads together on the KITTI dataset, performance is higher in most metrics compared to the model without forecasting. This proves that parallelizing two modules is beneficial to 3D MOT. Second, as shown in the figure next to Table II, on the KITTI dataset we validate the effect of the number of GNN layers on 3D MOT. We can see that, performance is significantly increased when using two layers of GNN, and then starts to converge with more layers. As a result, we use two GNN layers for shared feature learning in our method.

C. Evaluating Trajectory Forecasting

Evaluation Metrics. We use two standard metrics: Average Displacement Error (ADE) [10] and Final Displacement Error (FDE) for accuracy evaluation. To evaluate diversity of the trajectory samples and penalize similar samples, we use the

TABLE III
TRAJECTORY FORECASTING EVALUATION ON KITTI AND NUSCENES.

Datasets	Metrics	Conv-Social [16]	Social-GAN [11]	TraPHic [15]	Graph-LSTM [19]	**Ours (PTP)**
KITTI-1.0s	ADE↓	0.607	0.586	0.542	0.478	**0.471**
	FDE↓	0.948	1.167	0.839	0.800	**0.763**
	ASD↑	1.785	0.495	1.787	1.070	**2.351**
	FSD↑	1.987	0.844	1.988	1.836	**4.071**
KITTI-3.0s	ADE↓	2.362	2.340	2.279	1.994	**1.319**
	FDE↓	3.916	4.102	3.780	3.351	**2.299**
	ASD↑	2.436	1.351	2.434	2.745	**5.843**
	FSD↑	2.973	2.066	2.973	4.582	**10.123**
nuScenes-1.0s	ADE↓	0.674	0.483	0.571	0.509	**0.378**
	FDE↓	0.784	0.586	0.640	0.618	**0.490**
	ASD↑	2.101	1.005	2.102	1.122	**5.665**
	FSD↑	2.430	1.475	2.432	1.603	**7.826**
nuScenes-3.0s	ADE↓	1.989	1.794	1.827	1.646	**1.017**
	FDE↓	3.015	2.850	2.760	2.445	**1.527**
	ASD↑	2.799	1.945	2.803	2.742	**8.323**
	FSD↑	4.174	3.610	4.184	4.970	**15.787**

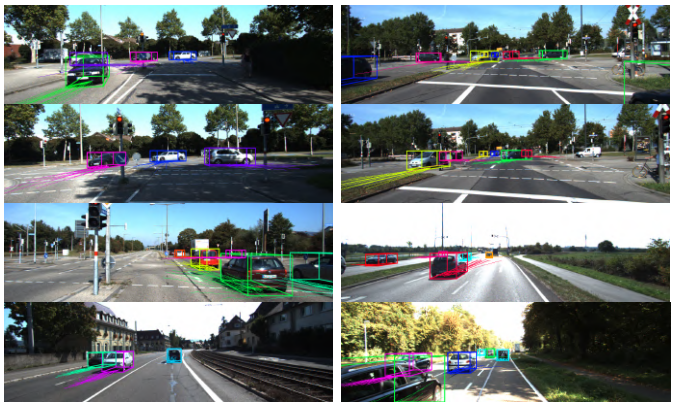
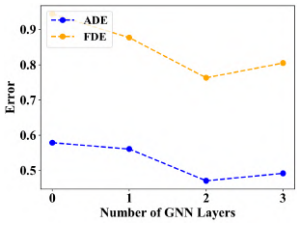


Fig. 6. Trajectory forecasting visualization on the KITTI dataset.

TABLE IV
EFFECT OF 3D MOT AND DIVERSITY SAMPLING ON FORECASTING.

Datasets	Metrics	w/o MOT+DSF	w/o DSF	+App	**Ours**
KITTI-1.0s	ADE↓	0.663	0.582	0.692	**0.471**
	FDE↓	1.121	0.978	1.085	**0.763**
	ASD↑	1.796	1.730	1.683	**2.351**
	FSD↑	3.168	3.052	3.194	**4.071**
KITTI-3.0s	ADE↓	1.729	1.564	2.104	**1.319**
	FDE↓	3.086	2.893	3.536	**2.299**
	ASD↑	3.196	3.416	2.951	**5.843**
	FSD↑	5.776	6.168	5.895	**10.123**



Average Self Distance (ASD) and Final Self Distance (FSD) metrics proposed in [59] for sample diversity evaluation.

Baselines. We compare against S.O.T.A. methods designed for vehicle trajectory forecasting [15], [16], [19] and person trajectory forecasting [11], among which [19] also leverages GNNs. As all baselines and our method are probabilistic approaches, we follow [11] and use 20 samples for all methods during evaluation. Following standard trajectory forecasting evaluation, which uses GT past trajectories (not estimated trajectories by a MOT module) to predict future trajectories, we feed GT trajectories in frames $t \in \{-H, \dots, 0\}$ to baselines and predict trajectories in frames $t \in \{1, \dots, T\}$. To have a relatively fair comparison to baselines, our method takes GT past trajectories in frames $t \in \{-H, \dots, -1\}$ and GT detections in frame 0 as inputs, and perform MOT in the current frame 0 and predict trajectories in frames $t \in \{1, \dots, T\}$ in parallel. As a result, the comparison is relatively fair as both baselines and our method have access to GT object information up to frame 0, though our method does not use GT object identity in the frame 0 (our method only uses GT detections in the frame 0 but does not know which past trajectory each detection is associated to). Note that, one can also view that the baseline methods perform sequential 3D MOT and trajectory forecasting (as shown in Fig. 1 top) but with a perfect object association in the frame 0 while our method does MOT and forecasting in parallel.

Results. We summarize trajectory forecasting results on the KITTI and nuScenes datasets in Table III. Our trajectory forecasting module, which (1) is jointly trained with a 3D MOT head in parallel, (2) uses GNNs for feature interaction and (3) uses diversity sampling, outperforms the baselines in both accuracy and diversity metrics. These results validate the advantage of our novel parallelized framework over prior cascaded track-forecast baselines (even using perfect tracking results in the frame 0 from GT, let alone using real-world 3D MOT modules with imperfect results). Also, our method outperforms baselines by a large margin in the long-horizon (i.e., 3.0s) prediction setting. This is because our method has a higher sample efficiency and can cover different modes of the future trajectory distribution. We show qualitative results of our method on the KITTI dataset in Fig. IV-B with plausible and diverse trajectory predictions drawn on the ground.

Ablation Study. We first verify if our parallelized joint tracking/forecasting optimization and diversity sampling function improves performance of the trajectory forecasting module on the KITTI dataset. In Table IV, we denote our model trained without the MOT head and DSF as w/o MOT+DSF, i.e., a standard multi-modal trajectory forecasting model based on GNNs and CVAE. Then, we add one module at a time. We first add the MOT head back, denoted as w/o DSF in Table IV, showing clear improvement in accuracy metrics and slight improvement in diversity metrics. We believe it is because that the auxiliary 3D MOT objective improves the shared feature learning, which is helpful to trajectory forecasting. Moreover, after adding the DSF module, denoted as **Ours** in Table IV, we see further improvement in both accuracy and diversity metrics. Also, another interesting ablation is to see if only using motion feature is the best option for our PTP framework compared to using both motion and appearance features as in [25]. Our results (+App) show that adding appearance leads to lower performance on ADE/FDE compared to w/o DSF, which we believe is because appearance feature is not meaningful to trajectory forecasting. In the figure next to Table IV, we also show the effect of GNN layers on forecasting performance. We observed that ADE/FDE are decreased when using GNNs (*e.g.*, 2 layers) compared to not using GNNs (*i.e.*, 0 layer), showing the effectiveness of GNN-based feature interaction for trajectory forecasting in our proposed parallelized framework.

V. CONCLUSION

We proposed a **P**arallelized 3D **T**racking and **P**rediction (PTP) framework that can avoid compounding errors and showed that it is beneficial to achieve both tasks under one unified framework through shared feature learning and joint optimization. Also, we incorporated two novel computational units into our approach: (1) a GNN-based feature interaction, which is introduced for the first time to a joint tracking and prediction framework; (2) a diversity sampling technique that improves sample efficiency for multi-agent trajectory forecasting. Through experiments, we established new state-of-the-art performance on both 3D MOT and trajectory forecasting, showing that the proposed method is effective.

REFERENCES

[1] S. Wang, D. Jia, and X. Weng, "Deep Reinforcement Learning for Autonomous Driving," *arXiv:1811.11329*, 2018.

[2] X. Weng, Y. Man, D. Cheng, J. Park, M. O'Toole, and K. Kitani, "All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds," 2020.

[3] S. Kayukawa, K. Higuchi, J. Guerreiro, S. Morishima, Y. Sato, K. Kitani, and C. Asakawa, "BBeep: A Sonic Collision Avoidance System for Blind Travellers and Nearby Pedestrians," *CHI*, 2019.

[4] A. Manglik, X. Weng, E. Ohn-bar, and K. M. Kitani, "Future Near-Collision Prediction from Monocular Video: Feasibility, Dataset , and Challenges," *IROS*, 2019.

[5] X. Sun, X. Weng, and K. Kitani, "When We First Met: Visual-Inertial Person Localization for Co-Robot Rendezvous," *IROS*, 2020.

[6] X. Weng, J. Wang, D. Held, and K. Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," *IROS*, 2020.

[7] W. Zhang, H. Zhou, S. Sun, Z. Wang, J. Shi, and C. C. Loy, "Robust Multi-Modality Multi-Object Tracking," *ICCV*, 2019.

[8] D. Frossard and R. Urtasun, "End-to-End Learning of Multi-Sensor 3D Tracking by Detection," *ICRA*, 2018.

[9] X. Weng and K. Kitani, "AutoSelect: Automatic and Dynamic Detection Selection for 3D Multi-Object Tracking," *arXiv:2012.05894*, 2020.

[10] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," *CVPR*, 2016.

[11] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *CVPR*, 2018.

[12] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, S. H. Rezatofighi, and S. Savarese, "Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks," *NeurIPS*, 2019.

[13] B. Ivanovic and M. Pavone, "The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs," *ICCV*, 2019.

[14] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents," *CVPR*, 2017.

[15] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "TraPHic: Trajectory Prediction in Dense and Heterogeneous Traffic Using Weighted Interactions," *CVPR*, 2019.

[16] N. Deo and M. M. Trivedi, "Convolutional Social Pooling for Vehicle Trajectory Prediction," *CVPRW*, 2018.

[17] X. Li, X. Ying, and M. C. Chuah, "GRIP: Graph-Based Interaction-Aware Trajectory Prediction," *ITSC*, 2019.

[18] Y. Yuan, X. Weng, Y. Ou, and K. Kitani, "AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting," *arXiv:2103.14023*, 2021.

[19] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting Trajectory and Behavior of Road-Agents Using Spectral Clustering in Graph-LSTMs," *arXiv:1912.01118*, 2019.

[20] A. Kulesza, B. Taskar, *et al.*, "Determinantal Point Processes for Machine Learning," *Foundations and Trends in Machine Learning*, 2012.

[21] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "FANTrack: 3D Multi-Object Tracking with Feature Association Network," *IV*, 2020.

[22] H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krähenbühl, T. Darrell, and F. Yu, "Joint Monocular 3D Vehicle Detection and Tracking," *ICCV*, 2019.

[23] I. Cherabier, C. Hane, M. R. Oswald, and M. Pollefeys, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *CVPR*, 2017.

[24] M. Simon, K. Amende, A. Kraus, J. Honer, T. Sämann, H. Kaulbersch, S. Milz, and H. M. Gross, "Complexer-YOLO: Real-Time 3D Object Detection and Tracking on Semantic Point Clouds," *CVPRW*, 2019.

[25] X. Weng, Y. Wang, Y. Man, and K. Kitani, "GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning," *CVPR*, 2020.

[26] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity Forecasting," *ECCV*, 2012.

[27] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes," *ECCV*, 2016.

[28] Y. Yuan and K. Kitani, "Ego-Pose Estimation and Forecasting as Real-Time PD Control," *ICCV*, 2019.

[29] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, "PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings," *ICCV*, 2019.

[30] N. Rhinehart, M. Kris, and P. Vernaza, "R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting," *ECCV*, 2018.

[31] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart, "Inverting the Pose Forecasting Pipeline with SPF2: Sequential Pointcloud Forecasting for Sequential Pose Forecasting," *CoRL*, 2020.

[32] Y. Wang, K. Kitani, and X. Weng, "Joint Object Detection and Multi-Object Tracking with Graph Neural Networks," *ICRA*, 2021.

[33] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-End Interpretable Neural Motion Planner," *CVPR*, 2019.

[34] S. Casas, W. Luo, and R. Urtasun, "IntentNet: Learning to Predict Intention from Raw Sensor Data," *CoRL*, 2018.

[35] W. Luo, B. Yang, and R. Urtasun, "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net," *CVPR*, 2018.

[36] M. Liang, B. Yang, W. Zeng, Y. Chen, R. Hu, S. Casas, and R. Urtasun, "PnPNet: End-to-End Perception and Prediction with Tracking in the Loop," *CVPR*, 2020.

[37] M. Gori, G. Monfardini, and F. Scarselli, "A New Model for Learning in Graph Domains," *IJCNN*, 2005.

[38] Y. Chen, M. Rohrbach, Z. Yan, S. Yan, J. Feng, and Y. Kalantidis, "Graph-Based Global Reasoning Networks," *CVPR*, 2019.

[39] L. Zhang, X. Li, A. Arnab, K. Yang, Y. Tong, and P. H. S. Torr, "Dual Graph Convolutional Network for Semantic Segmentation," *BMVC*, 2019.

[40] X. Wang and A. Gupta, "Videos as Space-Time Region Graphs," *ECCV*, 2018.

[41] M. Li, S. Chen, X. Chen, Y. Zhang, Y. Wang, and Q. Tian, "Actional-Structural Graph Convolutional Networks for Skeleton-based Action Recognition," *CVPR*, 2019.

[42] L. Shi, Y. Zhang, J. Cheng, and H. Lu, "Skeleton-Based Action Recognition with Directed Graph Neural Networks," *CVPR*, 2019.

[43] R. Zhao, K. Wang, H. Su, and Q. Ji, "Bayesian Graph Convolution LSTM for Skeleton Based Action Recognition," *ICCV*, 2019.

[44] J. Gao, T. Zhang, and C. Xu, "Graph Convolutional Tracking," *CVPR*, 2019.

[45] B. Seroussi and J.-L. Golmard, "An Algorithm Directly Finding the K Most Probable Configurations in Bayesian Networks," *International Journal of Approximate Reasoning*, 1994.

[46] D. Batra, P. Yadollahpour, A. Guzman-Rivera, and G. Shakhnarovich, "Diverse m-Best Solutions in Markov Random Fields," *ECCV*, 2012.

[47] S. Lee, S. P. S. Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra, "Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles," *NIPS*, 2016.

[48] W.-L. Hsiao and K. Grauman, "Creating Capsule Wardrobes from Fashion Images," *CVPR*, 2018.

[49] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse Sequential Subset Selection for Supervised Video Summarization," *NIPS*, 2014.

[50] S. Azadi, J. Feng, and T. Darrell, "Learning Detection with Diverse Proposals," *CVPR*, 2017.

[51] D.-A. Huang, M. Ma, W.-C. Ma, and K. M. Kitani, "How do We Use Our Hands? Discovering a Diverse Set of Common Grasps," *CVPR*, 2015.

[52] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode Regularized Generative Adversarial Networks," *ICLR*, 2017.

[53] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein Generative Adversarial Networks," *ICML*, 2017.

[54] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved Training of Wasserstein GANs," *NIPS*, 2017.

[55] M. Elfeki, C. Couprie, M. Riviere, and M. Elhoseiny, "GDPP: Learning Diverse Generations Using Determinantal Point Process," *ICML*, 2019.

[56] S. Zhao, J. Song, and S. Ermon, "InfoVAE: Balancing Learning and Inference in Variational Autoencoders," *AAAI*, 2019.

[57] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf, "Wasserstein Auto-Encoders," *ICLR*, 2018.

[58] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick, "Lagging Inference Networks and Posterior Collapse in Variational Autoencoders," *ICLR*, 2019.

[59] Y. Yuan and K. Kitani, "Diverse Trajectory Forecasting with Determinantal Point Processes," *ICLR*, 2020.

[60] Y. Yuan and K. M. Kitani, "Dlow: Diversifying Latent Flows for Diverse Human Motion Prediction," *ECCV*, 2020.

[61] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks," *AAAI*, 2019.
[62] H. W Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, 1955.
[63] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite," *CVPR*, 2012.
[64] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, and Q. Xu, "nuScenes: A Multimodal Dataset for Autonomous Driving," *CVPR*, 2020.
[65] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," *TPAMI*, 2016.
[66] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud," *CVPR*, 2019.
[67] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection," *CVPR*, 2019.