Whose Track Is It Anyway? Improving Robustness to Tracking Errors with Affinity-based Trajectory Prediction

Xinshuo Weng[1,3], Boris Ivanovic[3], Kris Kitani[1], Marco Pavone[2,3]

[1]Robotics Institute, Carnegie Mellon University
[2]Department of Aeronautics and Astronautics, Stanford University
[3]NVIDIA Research

Abstract

Multi-agent trajectory prediction is critical for planning and decision-making in human-interactive autonomous systems, such as self-driving cars. However, most prediction models are developed separately from their upstream perception (detection and tracking) modules, assuming ground truth past trajectories as inputs. As a result, their performance degrades significantly when using real-world noisy tracking results as inputs. This is typically caused by the propagation of errors from tracking to prediction, such as noisy tracks, fragments and identity switches. To alleviate this propagation of errors, we propose a new prediction paradigm that uses detections and their affinity matrices across frames as inputs, removing the need for error-prone data association during tracking. Since affinity matrices contain "soft" information about the similarity and identity of detections across frames, making prediction directly from affinity matrices retains strictly more information than making prediction from the tracklets generated by data association. Experiments on large-scale, real-world autonomous driving datasets show that our affinity-based prediction scheme[1] *reduces overall prediction errors by up to 57.9%, in comparison to standard prediction pipelines that use tracklets as inputs, with even more significant error reduction (up to 88.6%) if restricting the evaluation to challenging scenarios with tracking errors.*

1. Introduction

Tightly integrating multi-object tracking and trajectory prediction methods within an autonomy stack is a significant challenge due to the fact the most prior works develop multi-object tracking [5,14,17,28,38,39,41,42,49] and trajectory prediction [8, 12, 15, 16, 20, 32, 43, 46] approaches in isolation. Recently, [40,45] showed that feeding the outputs of a modern tracking method directly into a prediction model can significantly degrade performance (*e.g.*, up to 28× higher prediction error) in comparison to the common idealized prediction setting using perfect inputs (*i.e.*, ground truth trajectories). Unfortunately, this challenge of integration with an explicit consideration of robustness to tracking errors is still largely under-explored in the field.

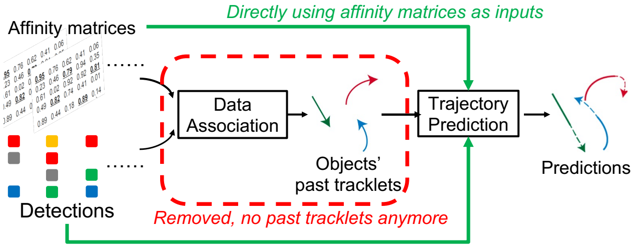

Figure 1. In contrast to standard tracking-prediction pipelines, our affinity-based prediction (Affinipred) skips data association, directly using detections and affinity matrices as prediction inputs.

Why does prediction accuracy significantly drop when using tracking results as inputs? Prior work [40, 45] has shown that tracking errors such as identity switches and fragments, which induce velocity or orientation estimation errors, are to blame. Typically, identity switches occur from mistakes in data association when two different objects are marked as the same due to ambiguity, *e.g.*, their appearance or geometric features are similar or they are very close. Fragments occur when data association fails to find a detection to match with an existing tracklet.

Now, is there an effective way to avoid these tracking errors from propagating to prediction? In this work, we argue that there is: skip data association altogether as shown in Fig. 1, since it may cause identity switches and fragments. Our key insight can be summarized as follows: Broadly, data association converts a "soft" affinity matrix (comprised of scalars between 0 and 1) to a boolean matrix (containing only 0 or 1) through a "hard" matching. By removing the

[1] Our project website is at https://www.xinshuoweng.com/projects/Affinipred.

data association step, we aim to preserve this "soft" information that is otherwise thrown away when "hard" matching is performed during data association. Because we skip the data association step, our method can be more accurate in the case where the affinity values are similar between two pairs of objects and it is ambiguous to perform correct matching and easy to make data association errors.

Contributions. We propose to directly use raw detections and intermediate tracking results, *i.e.*, affinity matrices, as inputs to prediction. To fully exploit the raw detections and affinity matrices for prediction, we propose:

(1) An **Affini**ty-based **pred**iction (Affinipred) framework that removes the need for input past trajectories and also the error-prone data association step.

(2) A transformer architecture that models joint attention between all detections at all input frames and can deal with a variable size of detections across frames.

(3) An affinity-based attention mechanism that directly incorporates full object affinity information across time.

In addition to the above contributions, Affinipred also leverages advancements from prior work by using a conditional variational autoencoder (CVAE) [18] to produce multi-modal predictions [21, 32], performing joint interaction modeling in the decoder to produce scene-consistent predictions [11,35], and incorporating maps in the inputs to capture environmental information [47, 48].

2. Related Work

Trajectory Prediction. Significant advancements in trajectory prediction [4, 15, 16, 19–21, 29–31] have been made in recent years. Yet, almost invariably, these works address trajectory prediction in an idealized setting, *i.e.*, using ground truth past trajectories as inputs for training and evaluation, with no direct accounting of perception errors. As a result, it is non-trivial to transfer models trained in this idealized setting to the real world, where we typically use noisy tracking results as inputs to prediction. Therefore, mitigating the propagation of errors from tracking to prediction is of critical importance, and serves as the motivation of this work. Different from prior work, our affinity-based prediction does not assume ground truth trajectories as inputs and provides significantly higher robustness to tracking errors.

Tracking-Prediction Integration. Although the majority of prior work focuses on trajectory prediction separately from tracking, there have been a few works which address the integration of tracking and prediction. For example, [23] proposed an end-to-end cascading detection, tracking and prediction network that is jointly optimized for these three tasks. Similarly, [43] proposed a parallelized tracking and prediction framework that can also be jointly optimized. Although these end-to-end methods yield increased performance, they do not explicitly account for tracking errors during prediction. Thus, if tracking results accumulated over frames contain errors, these errors will have a detrimental impact on prediction, as shown in [40, 45].

The closest, albeit concurrent, works to ours are [40, 45], which have both identified the impact of tracking errors on prediction and are seeking solutions. [45] propose enforcing prediction consistency at every frame to fix tracking errors, in turn improving prediction accuracy. Since data association is prone to errors, [40] propose to use multi-hypothesis data association (MHDA) [10, 25] for prediction. Similar to our idea, the use of MHDA also aims to preserve more information in the affinity matrices by generating multiple sets of tracking results, thereby increasing the likelihood of passing accurate tracking results as inputs to prediction. In short, both [40,45] improve prediction robustness by raising the quality of its inputs, i.e., the tracklets generated by tracking, but make no modifications to the prediction model. In contrast, we present a novel prediction framework that uses *affinity matrices* rather than tracklets as inputs, thereby completely removing the chances of errors occurring in data association and passing more information to prediction.

Joint Social-Temporal Modeling. In scenarios with many interacting agents, an agent's future behavior depends highly on its belief of other agents' behaviors. As a result, an important aspect of trajectory prediction is agent-agent interaction modeling, also known as social-temporal modeling. The most popular approach in prior work is to model social-temporal interactions in a cascaded order, i.e., first temporal modeling followed by social modeling, exemplified by RNNs and Graph Neural Networks (GNNs) [20,32,43], or Transformers and RNNs [6]. However, since social and temporal modeling are not performed jointly, social modeling loses access to agent information in previous frames (this information has already been compressed in temporal modeling). Accordingly, there are recent works which argue that it is beneficial to jointly model social-temporal interactions, *e.g.*, interleaving social and temporal transformers [24, 44, 46] or GNNs [26] in an iterative manner. We use a similar joint social-temporal modeling mechanism by allowing agent features at any frame to directly attend over features of other agents at any frame. Unlike prior work, we do not have access to agent identity information during joint social-temporal modeling so we use the joint affinity matrix to inject the "soft" identity information.

3D Multi-Object Tracking. The goal of online tracking is to match previously-computed tracklets with current detections to form new tracklets up to the current frame, and then incrementally obtain trajectories over the entire sequence. During this process, an intermediate affinity matrix is often computed where each entry represents the pairwise similarity between a past tracklet and current detection. A matching algorithm, such as the Hungarian algorithm [37], can then be used to perform "hard" matching, assigning detections to tracklets. Compared to the standard tracking-prediction pipeline that uses tracklets as inputs to prediction, we use the intermediate affinity matrices as inputs, preserving strictly more "soft" information for prediction.

3. Problem Formulation

The goal of multi-agent trajectory prediction is to learn a function $\mathcal{F}$ that can predict a sequence of future locations for every agent in the scene. Let $\mathcal{Y} = (\mathcal{Y}_1, \mathcal{Y}_2, \cdots, \mathcal{Y}_N) \in \mathbb{R}^{N \times K \times 2}$ denote N frames of K agents' future trajectories. At frame t, $\mathcal{Y}_t = (\boldsymbol{y}_t^1, \boldsymbol{y}_t^2, \cdots, \boldsymbol{y}_t^K) \in \mathbb{R}^{K \times 2}$ denotes the K agents' ground positions, where each position is represented as a tuple $\boldsymbol{y}_t^k = (u_t^k, v_t^k) \in \mathbb{R}^2$.

In prior work, the inputs to the function $\mathcal{F}$ are typically the past trajectories of the same K agents from the previous M frames $\mathcal{X} = (\mathcal{X}_{1-M}, \cdots, \mathcal{X}_0)$ and an optional map $\mathcal{M}$ of the scene. However, obtaining perfect past trajectories through tracking is challenging and data association errors can easily be propagated to prediction. Accordingly, we seek a different solution that instead uses a list of raw (unassociated) detections $\mathcal{D} = (\mathcal{D}_{1-M}, \cdots, \mathcal{D}_0)$ (Sec. 4.1) and their affinity matrices $\mathcal{A} = (\mathcal{A}_{2-M}^{1-M}$[2]$, \cdots, \mathcal{A}_0^{-1})$ between pairs of frames (Sec. 4.2), as shown in Fig. 2 (top left) and (bottom left). Since the future is uncertain and can evolve in many different ways, predictions for each agent should also be multi-modal. Thus, we aim to learn a function $\mathcal{F}$ that can map $(\mathcal{D}, \mathcal{A})$ (and optionally $\mathcal{M}$) and sampled latent variables $\mathcal{Z} = (\boldsymbol{z}_1, \cdots, \boldsymbol{z}_K)$ to future trajectories $\mathcal{Y}$:

$$\mathcal{Y} = \mathcal{F}(\mathcal{D}, \mathcal{A}, \mathcal{M}, \mathcal{Z}). \tag{1}$$

4. Affinity-based Prediction (Affinipred)

The network of our approach is illustrated in Fig. 2, it is comprised of five key components: (1) A past embedding layer that encodes features of all flattened detections and optionally a map in past frames; (2) An affinity construction module that converts individual affinity matrices between two frames into one joint affinity matrix between all detections from all frames; (3) A transformer encoder with affinity-based attention for interaction modeling between all detections that outputs context features, which can then be used to generate the prior p_θ; (4) A transformer decoder that predicts future trajectories autoregressively from embeddings, context and sampled latent variables; (5) Another transformer decoder that generates the posterior q_ϕ from ground truth (GT) future trajectories and context.

4.1. Input Representation: Detections

To avoid using potentially erroneous tracking results to perform prediction, we operate directly from raw detections. While this removes the need for data association, it poses two challenges: *incompleteness* and *loss of identity*.

Incompleteness. Due to a variety of reasons (occlusion, large distances, sensor failures, small object sizes, etc), state-of-the-art detectors often struggle to provide stable detections across all frames. This means that some objects, although present in the scene, may be not detected in some frames. For example, in Fig. 2 (top left), the blue object is detected in frames $t = -2$ and $t = 0$, but not in frame $t = -1$. Note that the object colors (denoting identity) in Fig. 2 are only used for illustration, and there is actually no identity information in the detections used by our method. Here, we denote raw detections at frame t as $\mathcal{D}_t = (\boldsymbol{d}_t^1, \boldsymbol{d}_t^2, \cdots, \boldsymbol{d}_t^{K_t})$, where K_t is the number of objects detected at time t (which can change across frames).

Loss of Identity. Since detections across frames are unassociated, we do not have a sequence of data for each object. As a result, standard sequence modeling techniques, e.g., RNNs, cannot be applied to extract trajectory-level features. Further, detections with the same index across frames (*i.e.*, $\boldsymbol{d}_t^k$ and $\boldsymbol{d}_{t-1}^k$) do not necessarily have the same identity.

Our Solution. To appropriately process detection inputs while accounting for the above challenges, we propose a joint social-temporal transformer architecture (Sec. 4.3 and 4.5) with affinity-based attention (Sec. 4.4). First, the joint social-temporal transformer models interactions between all detections across all frames so it can operate with a variable number of objects in each frame. Second, we inject affinity matrices into our affinity-based attention module (Sec. 4.4) as separate inputs that preserve "soft" similarity and identity information between all detections.

4.2. Input Representation: Affinity Matrices

Since affinity matrices provide strictly more information than the identity estimations produced by "hard" matching, affinity is a core facet of our method's input representation. Given K_{t-1} detections at frame $t-1$ and K_t detections at frame t, an affinity matrix $\mathcal{A}_t^{t-1} \in \mathbb{R}^{K_{t-1} \times K_t}$ has entries $a_{ij} \in \mathbb{R}$ that represent the similarity (or probability of corresponding to the same identity, if normalized) between $\boldsymbol{d}_{t-1}^i$ and $\boldsymbol{d}_t^j$, where $i \in [1, K_{t-1}]$ and $j \in [1, K_t]$ denote the row and column index of $\mathcal{A}_t^{t-1}$, respectively.

Affinity matrices are typically obtained as intermediate results from online multi-object tracking methods [41, 49]. However, there are two challenges in using them for prediction: (1) Since online tracking methods aim to solve data association incrementally over frames, affinity matrices are only computed between consecutive frames, *e.g.*, $\mathcal{A}_{-1}^{-2}$ and $\mathcal{A}_0^{-1}$, as shown in Fig. 2 (bottom left). As a result, we do not have direct affinity matrices between detections in non-consecutive frames; (2) Affinity matrices are typically computed between *tracklets* in the last frame and detections in the current frame, since tracklets are already associated up to the last frame. As a result, the estimated affinity matrix

[2] $\mathcal{A}_{2-M}^{1-M}$ denotes the affinity matrix between frame $2-M$ and $1-M$.

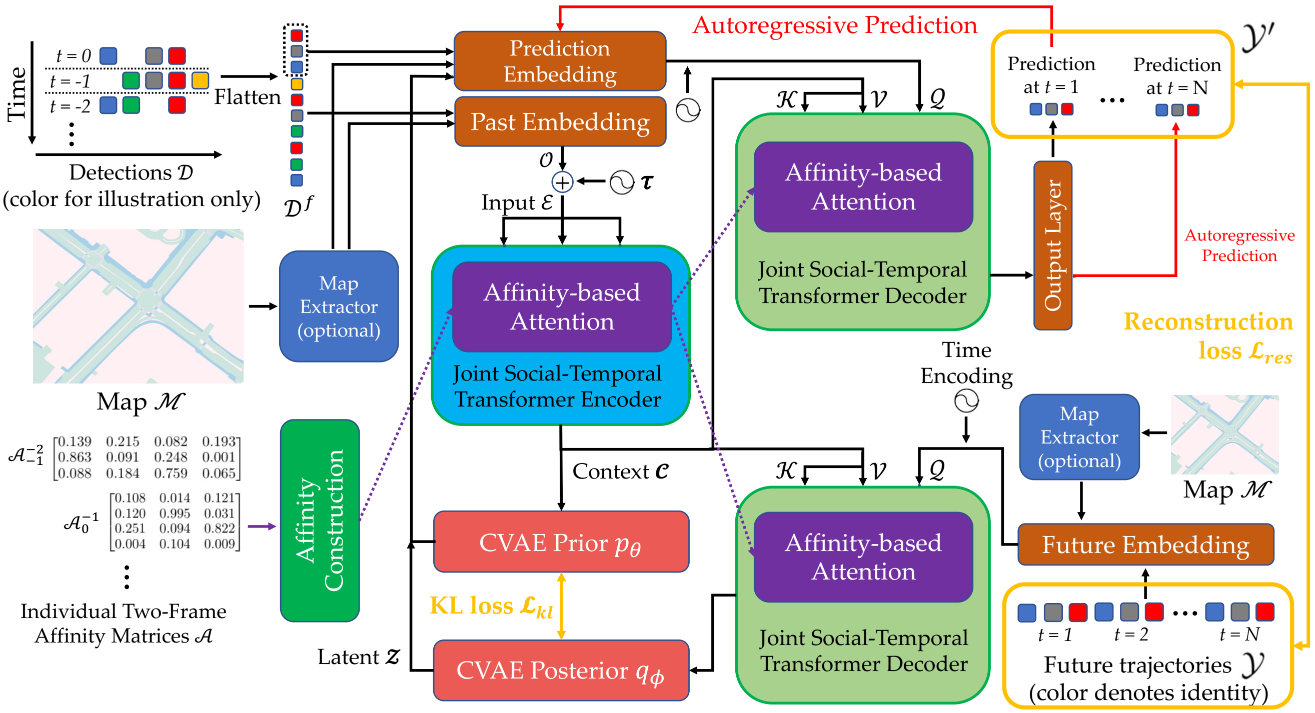
Time
t = 0
t = -1
t = -2
Flatten
Detections $\mathcal{D}$
(color for illustration only)
$\mathcal{D}^f$
Prediction Embedding
Past Embedding
Autoregressive Prediction
$\mathcal{Y}'$
Prediction at t = 1
Prediction at t = N
$\mathcal{O}$
τ
Input $\mathcal{E}$
$\mathcal{K}$
$\mathcal{V}$
$\mathcal{Q}$
Affinity-based Attention
Joint Social-Temporal Transformer Encoder
Joint Social-Temporal Transformer Decoder
Output Layer
Autoregressive Prediction
Reconstruction loss $\mathcal{L}_{res}$
Map Extractor (optional)
Map $\mathcal{M}$
Time Encoding
Context $\mathcal{C}$
$\mathcal{A}_{-1}^{-2}$ $\begin{bmatrix} 0.139 & 0.215 & 0.082 & 0.193 \\ 0.863 & 0.091 & 0.248 & 0.001 \\ 0.088 & 0.184 & 0.759 & 0.065 \end{bmatrix}$
$\mathcal{A}_{0}^{-1}$ $\begin{bmatrix} 0.108 & 0.014 & 0.121 \\ 0.120 & 0.995 & 0.031 \\ 0.251 & 0.094 & 0.822 \\ 0.004 & 0.104 & 0.009 \end{bmatrix}$
Individual Two-Frame Affinity Matrices $\mathcal{A}$
Affinity Construction
CVAE Prior p_θ
KL loss $\mathcal{L}_{kl}$
CVAE Posterior q_ϕ
Latent $\mathcal{Z}$
Future Embedding
t = 1
t = 2
t = N
Future trajectories $\mathcal{Y}$
(color denotes identity)

Figure 2. **Affinity-based Prediction**. Given inputs of past data (raw detections $\mathcal{D}$, map $\mathcal{M}$, and affinity matrices $\mathcal{A}$ between consecutive frames), our goal is to predict multiple samples of future trajectories $\widehat{\mathcal{Y}}$. To model joint social-temporal attention, all detections are flattened $\mathcal{D}^f$ before extracting $\mathcal{O}$ via the past embedding layer. To recover the order of time information in raw detections, a time encoding τ is added to $\mathcal{O}$ to form the input embedding $\mathcal{E}$. Then, a transformer encoder containing a series of self-attention blocks is applied to output context embeddings $\mathcal{C}$, with the "soft" object identity information contained in $\mathcal{A}$ injected through affinity-based attention. During training, future trajectories $\mathcal{Y}$ are also encoded via a transformer decoder to compute the posterior q_ϕ so that a KL loss $\mathcal{L}_{\text{kl}}$ can be applied between the prior p_θ and posterior q_ϕ. To produce multimodal predictions, latent variables $\mathcal{Z}$ are sampled from the prior (during testing) or the posterior (during training). By combining $\mathcal{Z}$ with detections at frame $t = 0$ (blue, gray, red objects), we can predict their future locations $\widehat{\mathcal{Y}}$ in an autoregressive manner via another transformer decoder and an output layer. Finally, a reconstruction loss $\mathcal{L}_{\text{res}}$ between $\mathcal{Y}$ and $\widehat{\mathcal{Y}}$ is also used to trains the network. For illustration purposes, we only show $M = 3$ past frames ($t = -2$, $t = -1$, and $t = 0$) in the figure.

might be inaccurate at a frame if the tracklet construction in the previous frame is inaccurate. We will discuss potential solutions to these challenges in Sec. 6.

4.3. Transformer Encoder

Past Embedding. We use a transformer encoder to model the joint social-temporal attentions between all detections in all frames. As transformers operate on sequences of data, we first flatten detections into a sequence $\mathcal{D}^f \in \mathbb{R}^{K_{\text{sum}} \times 2} = (\boldsymbol{d}_{1-M}^1, \cdots, \boldsymbol{d}_{1-M}^{K_{1-M}}, \boldsymbol{d}_{2-M}^1, \cdots, \boldsymbol{d}_0^1, \cdots, \boldsymbol{d}_0^{K_0})$, where $K_{\text{sum}} = \sum_{t=1-M}^{0} K_t$ is the number of detections in all past frames. Then, a past embedding (fully-connected) layer is used to convert $\mathcal{D}^f$ into a list of feature embeddings $\mathcal{O} = (\boldsymbol{o}_{1-M}^1, \cdots, \boldsymbol{o}_0^{K_0}) \in \mathbb{R}^{K_{\text{sum}} \times d_f}$.

Map Encoding. If available, a map can also (optionally) be included as an input to our model. Specifically, given an object's position $\boldsymbol{d}_t^k = (u_t^k, v_t^k)$, we crop a local map $\mathcal{M}_t^k$ around the object from the global map $\mathcal{M}$. Then, a convolutional neural network (CNN) encodes the local map as a vector $\boldsymbol{m}_t^k$ that is concatenated with the object's position before feeding into the past embedding layer.

Time Encoding. After flattening detections along the time dimension, the timestamp of each object is lost. To recover the order of time information for embeddings, we apply positional encoding as in [36], modified to instead encode time so that objects present at the same timestamp have the same time encoding. This time encoding $\boldsymbol{\tau}_t^k$ is added to the feature embedding $\boldsymbol{o}_t^k$ for each object to form the overall input embedding $\boldsymbol{e}_t^k$ to the transformer encoder.

Joint Social-Temporal Modeling. Inspired by the original transformer [36], we apply a series of self-attention blocks to the input embeddings $\mathcal{E} = (\boldsymbol{e}_{1-M}^1, \cdots, \boldsymbol{e}_0^{K_0})$, where each block contains a multi-head attention, layer normalization, feed-forward network, and another layer normalization. Each input embedding $\boldsymbol{e}_t^k$ is also projected into key, query, and value, which are used to compute attention (Sec. 4.4). After B attention blocks, the transformer encoder outputs context features $\mathcal{C} = (\boldsymbol{c}_{1-M}^1, \cdots, \boldsymbol{c}_0^{K_0})$.

Overall, our transformer encoder has two key characteristics: (1) Since the inputs are objects from all past frames, our transformer encoder allows an object at any frame to directly attend to any another object in any frame, enabling

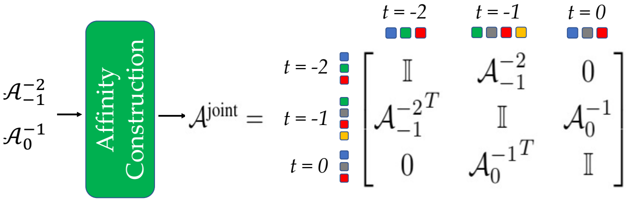

Figure 3. **Joint Affinity Construction**. To construct $\mathcal{A}^{\text{joint}}$, used in affinity-based attention, we first fill its diagonal with identity matrices $\mathbb{I}$, since object identities are fixed in the same timestamp. We then fill the off-diagonal blocks of $\mathcal{A}^{\text{joint}}$ with individual two-frame affinity matrices when available, otherwise they are left as zeros. In this example with $M = 3$ past frames, we can fill in affinity matrices $\mathcal{A}^{-1}_0$ (between frame -1 and 0) and $\mathcal{A}^{-2}_{-1}$ (between frame -2 and -1).

social attention across time, *i.e.*, joint social-temporal modeling. This differs from prior works which use a two-stage approach (*e.g.*, social then temporal or vice versa); (2) Since the inputs are flattened, our transformer encoder can operate on a variable number of objects across frames.

4.4. Affinity-based Attention

Joint Affinity Construction. To enable joint attention between all detections in all past frames, we first need to construct the joint affinity matrix $\mathcal{A}^{\text{joint}} \in \mathbb{R}^{K_{\text{sum}} \times K_{\text{sum}}}$ from individual two-frame affinity matrices $\mathcal{A}$. As an example, let $M = 3$ (three past frames). In this case, there are two individual affinity matrices, $\mathcal{A} = (\mathcal{A}^{-2}_{-1}, \mathcal{A}^{-1}_0)$. As shown in Fig. 3, the joint affinity $\mathcal{A}^{\text{joint}}$ is a (tridiagonal) block matrix composed of the affinity matrices $\mathcal{A}^{-2}_{-1}, \mathcal{A}^{-1}_0$, identity matrix $\mathbb{I}$, and zeros. The diagonal of $\mathcal{A}^{\text{joint}}$ is identity because the same object k at the same timestamp t must have the same identity to itself. The off-diagonal blocks of $\mathcal{A}^{\text{joint}}$ are filled with individual two-frame affinity matrices (for consecutive frames) and zeros otherwise.

Affinity-based Attention. Since an object's future trajectory depends on both its past motion and the motion of other objects, our affinity-based attention models both types of dependency. In general, these types of dependency may have different effects on prediction, so we use separate attention weights to model them by projecting the key $\mathcal{K} \in \mathbb{R}^{K_{\text{sum}} \times d_f}$ and query $\mathcal{Q} \in \mathbb{R}^{K_{\text{sum}} \times d_f}$ into two sets:

$$\mathcal{K}_{\text{self}} = \mathcal{K}\boldsymbol{W}^{\mathcal{K}}_{\text{self}}, \quad \mathcal{Q}_{\text{self}} = \mathcal{Q}\boldsymbol{W}^{\mathcal{Q}}_{\text{self}}, \tag{2}$$

$$\mathcal{K}_{\text{other}} = \mathcal{K}\boldsymbol{W}^{\mathcal{K}}_{\text{other}}, \quad \mathcal{Q}_{\text{other}} = \mathcal{Q}\boldsymbol{W}^{\mathcal{Q}}_{\text{other}}, \tag{3}$$

where $\mathcal{K}_{\text{self}}, \mathcal{Q}_{\text{self}} \in \mathbb{R}^{d_f \times d_f}$ are used to compute the attention of the object to itself (or to objects that have a high affinity value with it), and $\mathcal{K}_{\text{other}}, \mathcal{Q}_{\text{other}}$ are used to compute the attention of the object to others. We then combine these attention weights in an affinity-aware manner by using the joint affinity matrix $\mathcal{A}^{\text{joint}}$ as a mask:

$$\mathcal{W} = \mathcal{A}^{\text{joint}} \odot (\mathcal{Q}_{\text{self}}\mathcal{K}^T_{\text{self}}) + (1 - \mathcal{A}^{\text{joint}}) \odot (\mathcal{Q}_{\text{other}}\mathcal{K}^T_{\text{other}}), \tag{4}$$

where $\odot$ is an element-wise product and the combined attention weight $\mathcal{W} \in \mathbb{R}^{K_{\text{sum}} \times K_{\text{sum}}}$ is a matrix. Finally, we can compute the attended output as in [36]:

$$\mathcal{V}' = \text{softmax}\left(\frac{\mathcal{W}}{\sqrt{d_f}}\right)\mathcal{V}, \tag{5}$$

where the output embeddings $\mathcal{V}' \in \mathbb{R}^{K_{\text{sum}} \times d_f}$ are used as inputs to the rest of the operations in the attention block.

4.5. Transformer Decoder

Different from the transformer encoder that computes the self-attention between all detections, our transformer decoder computes the cross-attention between detections and future positions. As shown in Fig. 2, our transformer decoder is used in two places: (1) to model attention between detections and predicted trajectories in an autoregressive manner, and (2) to model attention between detections and GT future trajectories to compute the posterior. In both places, the only difference is that the query embeddings are projected from the predicted positions via the prediction embedding layer or from the GT future trajectories via the future embedding layer. Both embedding layers are fully-connected, similar to the past embedding layer as introduced in Sec. 4.3. Affinity-based attention and time encoding are also used in both transformer decoders, and the key and value embeddings are projected from the context feature $\mathcal{C}$ from the transformer encoder.

4.6. Multi-Modal Prediction

To produce multi-modal predictions, we apply the standard CVAE framework, sampling latent values $\mathcal{Z}$ at runtime and conditioning the model with them to produce multiple plausible joint realizations of agent futures.

CVAE Posterior. Following standard CVAE trajectory prediction works [9,43], the posterior $q_\phi(\boldsymbol{z}^k_t|\mathcal{D}, \mathcal{M}, \mathcal{Y})$ is computed with a two-layer Multi-Layer Perceptron followed by two separate fully-connected layers to compute the mean $\boldsymbol{\mu}^k_t$ and variance $\boldsymbol{\sigma}^k_t$ of a Gaussian for each input detection $\boldsymbol{d}^k_t$. Latent variables $\mathcal{Z} = (\boldsymbol{z}^1_{1-M}, \cdots, \boldsymbol{z}^{K_0}_0) \in \mathbb{R}^{K_{\text{sum}} \times d_z}$ can then be sampled for all detected objects during training, which are concatenated with the raw detections $\mathcal{D}^f$ and (optionally) map features to compute the input embeddings during autoregressive prediction. Finally, the posterior is computed using the outputs from the transformer encoder $\mathcal{C}$ and GT future transformer decoder, incorporating information about both the future $(\mathcal{Y}, \mathcal{M})$ and the past $(\mathcal{D}, \mathcal{M})$.

CVAE Prior. The network for computing the prior is similar to the network for computing posterior, except that its inputs are the context feature $\mathcal{C}$ that do not contain any information from the future GT trajectories. During testing, we can sample multiple latent variables $\mathcal{Z} = (\boldsymbol{z}^1_{1-M}, \cdots, \boldsymbol{z}^{K_0}_0)$ from the prior $p_\theta(\boldsymbol{z}^k_t|\mathcal{D}, \mathcal{M})$ to predict multiple samples of object future trajectories.

4.7. Training Details

Which agents to predict? Since raw detections are different from trajectory inputs in standard trajectory prediction, a key question is for which objects should we predict? For example, one can predict future locations for all K_{sum} objects detected in past frames. However, some detections in past frames may belong to the same object (although this is unknown in advance), so predicted trajectories may be duplicated, requiring subsequent matching and filtering.

Another way is to predict trajectories only for the K_0 objects detected in the current frame ($t = 0$). In this way, no duplicate predictions are made for objects with the same identity, however, some objects might not have predictions if they are not detected in frame $t = 0$ due to detector failure. In this work, we choose to make $K = K_0$ rather than K_{sum}, matching standard data processing procedures used in prior works, e.g., [15, 40], and challenges (nuScenes [7], ETH/UCY [22, 27]), to enable a fair comparison.

Detection Matching with GT. To train Affinipred, we use GT future trajectories as supervision signals for each detection that needs to be predicted. Accordingly, this requires matching the K_0 detections at frame $t = 0$ with GT future trajectories. Following the standard matching scheme used for official evaluation in the nuScenes challenge [7], we use a distance threshold d_{thres} and Hungarian matching [37]. During training, we filter out detections at frame $t = 0$ that are not successfully matched with GT (a small percentage), *i.e.*, objects that are most likely false positives.

Autoregressive Prediction. During both training and testing, we perform autoregressive prediction. Object positions are predicted sequentially, with predictions at t serving as inputs when predicting the next timestamp $t + 1$. Using Fig. 2 as an example, we begin with detections at frame $t = 0$ as inputs (blue, gray and red objects). Then, our output layer converts the transformer decoder output to predicted positions $\widehat{\mathcal{Y}}_1 = (\widehat{\boldsymbol{y}}_1^1, \cdots, \widehat{\boldsymbol{y}}_1^{K_0})$ at frame $t = 1$. By iteratively applying this process, we can obtain predictions in all N future frames $\widehat{\mathcal{Y}} = (\widehat{\mathcal{Y}}_1, \cdots, \widehat{\mathcal{Y}}_N)$.

Training Objectives. We optimize our Affinipred network by minimizing two loss functions: (1) A reconstruction loss between the K_0 predicted $\widehat{\mathcal{Y}}$ and GT trajectories $\mathcal{Y}$; and (2) a KL divergence loss between the CVAE prior and posterior for each object being predicted:

$$\begin{aligned}\mathcal{L} = \alpha\mathcal{L}_{\text{res}} + \beta\mathcal{L}_{\text{KL}} &= \alpha||\widehat{\mathcal{Y}} - \mathcal{Y}||_2 \\ &+ \beta\sum_{k=1}^{K_0}\text{KL}(q_\phi(\boldsymbol{z}_0^k|\mathcal{D}, \mathcal{M}, \mathcal{Y}) \;||\; p_\theta(\boldsymbol{z}_0^k|\mathcal{D}, \mathcal{M})),\end{aligned} \tag{6}$$

where $\alpha, \beta \in \mathbb{R}_{>0}$ are loss function weights. Here, only the prior and posterior of the K_0 objects are used because we only predict these objects detected at frame $t = 0$.

5. Experiments

Our affinity-based prediction scheme has two core goals: (1) Improve overall prediction performance in a realistic setting when using tracking results as inputs, and (2) Increase robustness to upstream tracking errors. To evaluate our method's efficacy in addressing these, we make use of standard prediction datasets, but use detection and tracking results as inputs rather than GT past trajectories. In addition to evaluating prediction accuracy for all detected objects ("global evaluation"), we follow [40] and evaluate prediction performance for objects with certain types of tracking errors ("targeted evaluation"), to evaluate our method's robustness to tracking errors. At a high level, we find that our affinity-based prediction significantly improves real-world prediction performance and robustness to tracking errors, compared to standard tracklet-based prediction.

5.1. Implementation Details

Hyper-parameters. We use $\alpha = \beta = 1$ in Eq. (6), $d_f = 256$ for embeddings, $d_z = 32$ for latent variables, and $d_{\text{thres}} = 2$ when matching detections with GT trajectories.

Network Architecture. We use a 4-layer CNN to extract 32-dimensional map features. We use one fully-connected layer in all four embedding layers (past, future, prediction embedding, and output). We use $B = 2$ attention blocks in both the transformer encoder and decoder, and 8 heads in the affinity-based multi-head attention.

Data Pre-processing. We apply linear interpolation and extrapolation to impute missing data in any of the past M and future N frames. We also apply three types of data augmentation to increase the robustness of prediction with respect to tracking errors. In particular, we inject identity switches (IDS), fragments (FRAG), and noise. For IDS, we swap the identity of two objects in the current frame with 30% probability if they are within 5 meters. For FRAG, we randomly drop data in past frames with 30% probability. Finally, we perturb the u, v positions of each object with a maximum magnitude of 0.4 of its width and length in previous frames.

5.2. Evaluation Methodology

Baselines. In all experiments, we compare against a standard tracking-prediction pipeline that is composed of a detector, tracker, and prediction model. On KITTI, these are PointRCNN [34], AB3DMOT [41], and PTP [43], respectively. On nuScenes, these are Megvii [50], AB3DMOT [41], and PTP [43], respectively. We also compare against MTP [40], a strong, recent work that tackles the same problem of improving prediction robustness to upstream tracking errors. Specifically, MTP [40] explicitly accounts for tracking errors in prediction via multi-hypothesis data association. To fairly compare with these baselines, we use the same detection and tracking methods to obtain the inputs.

Datasets. We use the standard KITTI [13] and nuScenes [7] autonomous driving datasets for evaluation. Since there is no official prediction evaluation server for KITTI, we evaluate on the KITTI tracking [1] validation set, using the standard train/val splits from [33]. We consider three main object categories during evaluation: cars, pedestrians and cyclists. To fairly compare against [40], we use $M = 10$ past frames and predict $N = 10$ future frames. To match objects with their GT trajectories, we compute 3D Intersection over Union (IoU) and use a standard threshold of 0.5 [41].

For nuScenes, we follow official prediction challenge guidelines [2]: (1) using the official train, val, test splits [3]; (2) evaluating only on vehicle classes; (3) using $M = 4$ past frames to predict $N = 12$ future frames; and (4) using a threshold distance of 2 meters when matching predictions with GT trajectories to compute metrics. Although we follow official nuScenes evaluation guidelines, it is important to note that the minADE$_S$, minFDE$_S$ values in Tables 1 through 4 are not comparable to values on the nuScenes leaderboard (which use GT past trajectories for evaluation).

Metrics. To evaluate prediction accuracy, we use the standard Minimum Average and Final Displacement Error metrics over S predicted samples (minADE$_S$, minFDE$_S$):

$$\text{minADE}_S = \min_s \frac{1}{K_0} \sum\nolimits_{k=1}^{K_0} \frac{1}{N^k_{\text{valid}}} \sum\nolimits_{t=1}^{N} \text{F}^k_m ||\widehat{\boldsymbol{y}}_{st}^{\,k} - \boldsymbol{y}^k_t||_2,$$

$$\text{minFDE}_S = \min_s \frac{1}{K_0} \sum\nolimits_{k=1}^{K_0} ||\widehat{\boldsymbol{y}}_{sT_k}^{\,k} - \boldsymbol{y}^k_{T_k}||_2,$$

where K_0 is the number of objects detected in the current frame, $\widehat{\boldsymbol{y}}_{st}^{\,k}$ is the predicted position of object k at frame t in the s^{th} sample. Since not all objects have complete GT future trajectories (e.g., they may leave the scene early), a frame-wise mask F^k_m is used for object k to compute errors only on the frames where GT exists. As a result, the number of total evaluated frames for object k is $N^k_{\text{valid}} \leq N$. Similarly, when computing minFDE$_S$, $T_k \leq N$ is the last frame of object k. Lastly, we evaluate with $S = 20$ on KITTI and $S = 10$ on nuScenes to have a fair comparison with [40].

Tracking Evaluation. In order to analyze prediction performance on objects with tracking errors, it is necessary to determine which objects have tracking errors. Following [40], we consider two common tracking errors: identity switches (IDS) and fragments (FRAG). We use the standard 3D tracking evaluation code released in [41] to identify which objects have IDS/FRAG errors per frame.

5.3. Results and Analysis

Targeted Evaluation. Table 1 summarizes the performance of all methods for objects with tracking errors. We can see that our method *significantly* outperforms all others on every dataset and tracking error type. For instance, on nuScenes objects with IDS, we reduce minADE$_S$ by 71.8% (from 3.923 to 1.106), and reduce minFDE$_S$ by 74.5% (from 6.210 to 1.584). Importantly, we can directly compare to MTP [40] because our method uses the same detection and tracking outputs (including intermediate affinity matrices) as MTP [40]. Thus, the set of considered IDS/FRAG objects is exactly the same for both methods.

Datasets	Targets	Methods	minADE$_S$	minFDE$_S$
KITTI	IDS	PointRCNN+AB3DMOT+PTP, S=20	2.820	4.514
		MTP [40], S=20	0.747	1.173
		MTP [40], S=400	0.707	1.093
		Affinipred (Ours), S=20	0.516	0.792
KITTI	FRAG	PointRCNN+AB3DMOT+PTP, S=20	1.621	2.155
		MTP [40], S=20	1.335	1.688
		MTP [40], S=400	1.305	1.627
		Affinipred (Ours), S=20	1.063	1.381
nuScenes	IDS	Megvii+AB3DMOT+PTP, S=10	8.345	13.892
		MTP [40], S=10	3.923	6.210
		MTP [40], S=200	3.321	5.052
		Affinipred (Ours), S=10	1.106	1.584
nuScenes	FRAG	Megvii+AB3DMOT+PTP, S=10	14.520	21.815
		MTP [40], S=10	8.476	12.105
		MTP [40], S=200	7.697	10.606
		Affinipred (Ours), S=10	4.486	5.600

Table 1. Prediction performance for objects with IDS/FRAG.

Our method's performance is further highlighted when comparing against a standard detection-tracking-prediction system that uses the same detection [34, 50] and tracking [41] methods. For example, on nuScenes objects with IDS, we reduce minADE$_S$ by 86.7% (from 8.345 to 1.106), and reduce minFDE$_S$ by 88.6% (from 13.892 to 1.584). All these results confirm that our affinity-based prediction has significantly improved the robustness of prediction with respect to IDS and FRAG, which is reasonable as the standard detection-tracking-prediction system does not explicitly account for tracking errors.

Global Evaluation. Table 3 summarizes the performance of methods for all detected objects. Similar to the trend in targeted evaluation, our method significantly outperforms a standard detection-tracking-prediction pipeline. For example, we reduce minADE$_S$ by 57.9% (from 2.320 to 0.977) on nuScenes. Also, our method significantly outperforms MTP [40], even when allowing it 20× more samples. The magnitude of error reduction our method achieves in Table 3 is less drastic than in Table 1 because global evaluation also includes objects whose future trajectories are easier to predict, *i.e.*, without any upstream detection or tracking errors.

Ablation Study. Table 2 summarizes the results, where the last row corresponds to our full method. In the second-last row, we replace our method's core advancement (the use of affinity and detection inputs) with tracklets, and observe a significant drop in both targeted and global performance. This signifies that our core contribution is effective at both improving overall prediction performance and improving model robustness to tracking errors.

We then disable GT matching, meaning future tracklets from tracking are used in place of GT future trajectories

Methods				IDS		FRAG		Global	
Extra/Inter.	Augmentation	GT matching	Affinity	minADE$_S$	minFDE$_S$	minADE$_S$	minFDE$_S$	minADE$_S$	minFDE$_S$
				3.408	5.124	8.085	10.246	1.625	2.960
✓				2.705	4.067	6.892	8.227	1.264	2.136
✓	✓			2.464	3.531	6.622	8.058	1.215	1.963
✓	✓	✓		1.792	2.642	4.988	6.193	1.143	1.882
✓	✓	✓	✓	1.106	1.584	4.486	5.600	0.977	1.628

Table 2. Ablation experiments on the nuScenes prediction test set. $S = 10$ samples are used to predict $N = 12$ future frames.

Datasets	Methods	minADE$_S$	minFDE$_S$
KITTI	PointRCNN+AB3DMOT+PTP, S=20	0.185	0.278
	MTP [40], S=20	0.162	0.238
	MTP [40], S=400	0.146	0.203
	Affinipred (Ours), S=20	0.129	0.194
nuScenes	Megvii+AB3DMOT+PTP, S=10	2.320	3.819
	MTP [40], S=10	1.585	2.512
	MTP [40], S=200	1.325	1.979
	Affinipred (Ours), S=10	0.977	1.628

Table 3. Prediction performance for all detected objects.

during training. Since tracklets are noisy, performance sensibly decreases in all settings. Finally, we disable data augmentation and imputation (linear extrapolation and interpolation), and the resulting decrease in performance confirms that these pre-processing steps are also important to improve prediction robustness, especially so for objects with IDS and FRAG errors. Interestingly, even without these data pre-processing steps and affinity-based inputs, our base prediction network performs similarly to MTP [40].

Training with Augmented GT. As we have seen in Table 2, augmenting the inputs with IDS, FRAG, and noises during training improves prediction accuracy and robustness to tracking errors. Seeing this, one may wonder if we can apply the same augmentations to a standard tracklet-based prediction method that is trained with GT trajectories. How would this compare to our Affinipred?

To answer this question, we train a tracklet-based version of our base prediction network (*i.e.*, the second row of Table 2) on nuScenes with different tracking error augmentations applied to GT trajectories (as described in Section 5.1). Table 4 summarizes the results and shows that, even with the combination of all three input data augmentations, a tracklet-based method trained with GT trajectories is also outperformed by our Affinipred, which is the case for both objects with ID switches and fragments.

Runtime Speed. The speed of our method depends on the number of objects present per input frame. On a single GeForce RTX 2080, our method can run at 4.7 FPS with an average of 30 detections per input frame. There are a number of ways to make our method faster beyond this, such as switching from autoregressive to batch predictions, optimizing the network architecture to reuse computation, and generally optimizing our (currently unoptimized PyTorch) codebase for performance.

Inputs	Augmentation			IDS		FRAG	
	IDS	FRAG	Noises	minADE$_S$	minFDE$_S$	minADE$_S$	minFDE$_S$
GT traj.				2.878	4.136	7.314	9.136
	✓			1.383	1.906	5.160	6.264
		✓		2.577	3.695	6.610	7.830
			✓	1.897	2.557	6.060	7.460
	✓	✓	✓	1.314	1.824	4.773	5.959
Affi. + Det.	✓	✓	✓	1.106	1.584	4.486	5.600

Table 4. Performance of a tracklet-based variant of our method trained with augmented GT trajectories compared to our affinity-based prediction scheme on nuScenes. Data imputation with extra-/interpolation is used for both methods.

6. Limitations and Conclusions

There are two main limitations (and thus areas of future work) in our affinity-based prediction method (Affinipred). First, how can one obtain affinity matrices between non-consecutive frames? Further, how can one obtain affinity matrices between two frames of *detections* rather than between past tracklets and current detections (Sec. 4.2)? Along these lines, an interesting area of future work is to design an affinity estimation network that can directly output the joint affinity matrix $\mathcal{A}^{\text{joint}}$. Second, is there an elegant way to predict the futures of all objects present in any past frame, rather than only those detected in the current frame? Such a capability would be useful for handling occlusions, being able to predict for objects that were in a past frame but are occluded in the current frame.

In conclusion, we present a novel affinity-based prediction scheme that only requires detections and their affinity matrices across frames as inputs, entirely removing the need for error-prone data association. Since affinity matrices contain "soft" information about the similarity and identity of detections across frames, making prediction using affinity matrices retains strictly more information than making prediction using tracklets generated by data association. Experiments on the KITTI and nuScenes autonomous driving datasets show that our affinity-based prediction scheme reduces overall prediction errors by up to 57.9% in comparison to standard tracklet-based prediction pipelines, with even more significant error reductions (up to 88.6%) when restricting evaluation to objects with tracking errors.

References

[1] KITTI Tracking Dataset. http://www.cvlibs.net/datasets/kitti/eval_tracking.php. 7

[2] nuScenes Prediction Challenge Guidelines. https://www.nuscenes.org/prediction?externalData=all&mapData=all&modalities=Any. 7

[3] nuScenes Prediction Data Split. https://github.com/nutonomy/nuscenes-devkit/blob/master/python-sdk/nuscenes/eval/prediction/splits.py. 7

[4] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016. 2

[5] Nuri Benbarka, Jona Schroder, and Andreas Zell. Score refinement for confidence-based 3D multi-object tracking. *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2021. 1

[6] Manoj Bhat, Jonathan Francis, and Jean Oh. Trajformer: Trajectory Prediction with Local Self-Attentive Contexts for Autonomous Driving. *Machine Learning for Autonomous Driving Workshop at Conf. on Neural Information Processing Systems*, 2020. 2

[7] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, and Qiang Xu. nuScenes: A Multimodal Dataset for Autonomous Driving. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020. 6, 7

[8] Defu Cao, Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Spectral Temporal Graph Neural Network for Multivariate Time-series Forecasting. *Proc. IEEE Conf. on Robotics and Automation*, 2021. 1

[9] Hao Cheng, Wentong Liao, Xuejiao Tang, Michael Ying Yang, Monika Sester, and Bodo Rosenhahn. Exploring Dynamic Context for Multi-path Trajectory Prediction. *Proc. IEEE Conf. on Robotics and Automation*, 2021. 5

[10] Ingemar J. Cox and Sunita L. Hingorani. An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and Its Evaluation for the Purpose of Visual Tracking. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 1996. 2

[11] Alexander Cui, Abbas Sadat, Sergio Casas, Renjie Liao, and Raquel Urtasun. LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving. *IEEE Int. Conf. on Computer Vision*, 2021. 2

[12] Stuart Eiffert, Kunming Li, Mao Shan, Stewart Worrall, Salah Sukkarieh, and Eduardo Nebot. Probabilistic Crowd GAN: Multimodal Pedestrian Trajectory Prediction using a Graph Vehicle-Pedestrian Attention Network. *IEEE Robotics and Automation Letters*, 2020. 1

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? the KITTI Vision Benchmark Suite. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012. 7

[14] Xusen Guo and Kai Huang. 3D Object Detection and Tracking on Streaming Data. *Proc. IEEE Conf. on Robotics and Automation*, 2020. 1

[15] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018. 1, 2, 6

[16] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs. *IEEE Int. Conf. on Computer Vision*, 2019. 1, 2

[17] Aleksandr Kim, Aljosa Osep, and Laura Leal-Taixé. EagerMOT: 3D Multi-Object Tracking via Sensor Fusion. *Proc. IEEE Conf. on Robotics and Automation*, 2021. 1

[18] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114*, 2013. 2

[19] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity Forecasting. *European Conf. on Computer Vision*, 2012. 2

[20] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. Social-BiGAT: Multimodal Trajectory Forecasting using Bicycle-GAN and Graph Attention Networks. *Conf. on Neural Information Processing Systems*, 2019. 1, 2

[21] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H.S. Torr, and Manmohan Chandraker. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017. 2

[22] Alon Lerner, Yiorgos Chrysanthou, and Dani Lischinski. Crowds by Example. *Computer Graphics Forum*, 2007. 6

[23] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. PnPNet: End-to-End Perception and Prediction with Tracking in the Loop. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020. 2

[24] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal Motion Prediction with Stacked Transformers. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2021. 2

[25] Yun Li Long, Hui Xu, and Wei An. An Algorithm for Tracking Multiple Targets. *IEEE Transactions on Automatic Control*, 1979. 2

[26] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-STGCNN: A Social Spatio-Temporal Graph Convolutional Neural Network for Human Trajectory Prediction. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020. 2

[27] Stefano Pellegrini, Andreas Ess, Konrad Schindler, and Luc Van Gool. You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. *IEEE Int. Conf. on Computer Vision*, 2009. 6

[28] Johannes Pöschmann, Tim Pfeifer, and Peter Protzel. Factor Graph based 3D Multi-Object Tracking in Point Clouds. *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2020. 1

[29] Nicholas Rhinehart, M Kris, and Paul Vernaza. R2P2: A ReparameteRized Pushforward Policy for Diverse, Precise Generative Path Forecasting. *European Conf. on Computer Vision*, 2018. 2

[30] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. *IEEE Int. Conf. on Computer Vision*, 2019. 2

[31] Alexandre Robicquet, Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Learning Social Etiquette: Human Trajectory Understanding In Crowded Scenes. *European Conf. on Computer Vision*, 2016. 2

[32] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data. *European Conf. on Computer Vision*, 2020. 1, 2

[33] Samuel Scheidegger, Joachim Benjaminsson, Emil Rosenberg, Amrit Krishnan, and Karl Granstr. Mono-Camera 3D Multi-Object Tracking Using Deep Learning Detections and PMBM Filtering. *IEEE Intelligent Vehicles Symposium*, 2018. 7

[34] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 6, 7

[35] Yichuan Charlie Tang and Ruslan Salakhutdinov. Multiple Futures Prediction. *Conf. on Neural Information Processing Systems*, 2019. 2

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Conf. on Neural Information Processing Systems*, 2017. 4, 5

[37] H W Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics*, 1955. 2, 6

[38] Sukai Wang, Yuxiang Sun, and Ming Liu. PointTrackNet: An End-to-End Network for 3D Object Detection and Tracking from Point Cloud. *Proc. IEEE Conf. on Robotics and Automation*, 2020. 1

[39] Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint Object Detection and Multi-Object Tracking with Graph Neural Networks. *Proc. IEEE Conf. on Robotics and Automation*, 2021. 1

[40] Xinshuo Weng, Boris Ivanovic, and Marco Pavone. MTP: Multi-hypothesis Tracking and Prediction for Reduced Error Propagation. *arXiv:2110.09481*, 2021. 1, 2, 6, 7, 8

[41] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2020. 1, 3, 6, 7

[42] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020. 1

[43] Xinshuo Weng, Ye Yuan, and Kris Kitani. PTP: Parallelized Tracking and Prediction with Graph Neural Networks and Diversity Sampling. *IEEE Robotics and Automation Letters*, 2021. 1, 2, 5, 6

[44] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. *European Conf. on Computer Vision*, 2020. 2

[45] Rui Yu and Zihan Zhou. Towards Robust Human Trajectory Prediction in Raw Videos. *arXiv:2108.08259*, 2021. 1, 2

[46] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. AgentFormer: Agent-Aware Transformers for Socio-Temporal Multi-Agent Forecasting. *IEEE Int. Conf. on Computer Vision*, 2021. 1, 2

[47] Wenyuan Zeng, Ming Liang, Renjie Liao, and Raquel Urtasun. LaneRCNN: Distributed Representations for Graph-Centric Motion Forecasting. *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 2021. 2

[48] Lingyao Zhang, Po-Hsun Su, Jerrick Hoang, Galen Clark Haynes, and Micol Marchetti-Bowick. Map-Adaptive Goal-Based Trajectory Prediction. *Conf. on Robot Learning*, 2020. 2

[49] Wenwei Zhang, Hui Zhou, Shuyang Sun, Zhe Wang, Jianping Shi, and Chen Change Loy. Robust Multi-Modality Multi-Object Tracking. *IEEE Int. Conf. on Computer Vision*, 2019. 1, 3

[50] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection. *IEEE Conf. on Computer Vision and Pattern Recognition*, 2019. 6, 7