Joint Forecasting of Panoptic Segmentations with Difference Attention

Colin Graber[1] Cyril Jazra[1] Wenjie Luo[2] Liangyan Gui[1] Alexander Schwing[1]
[1] University of Illinois at Urbana-Champaign [2] Waymo

Abstract

Forecasting of a representation is important for safe and effective autonomy. For this, panoptic segmentations have been studied as a compelling representation in recent work. However, recent state-of-the-art on panoptic segmentation forecasting suffers from two issues: first, individual object instances are treated independently of each other; second, individual object instance forecasts are merged in a heuristic manner. To address both issues, we study a new panoptic segmentation forecasting model that jointly forecasts all object instances in a scene using a transformer model based on 'difference attention.' It further refines the predictions by taking depth estimates into account. We evaluate the proposed model on the Cityscapes and AIODrive datasets. We find difference attention to be particularly suitable for forecasting because the difference of quantities like locations enables a model to explicitly reason about velocities and acceleration. Because of this, we attain state-of-the-art on panoptic segmentation forecasting metrics.

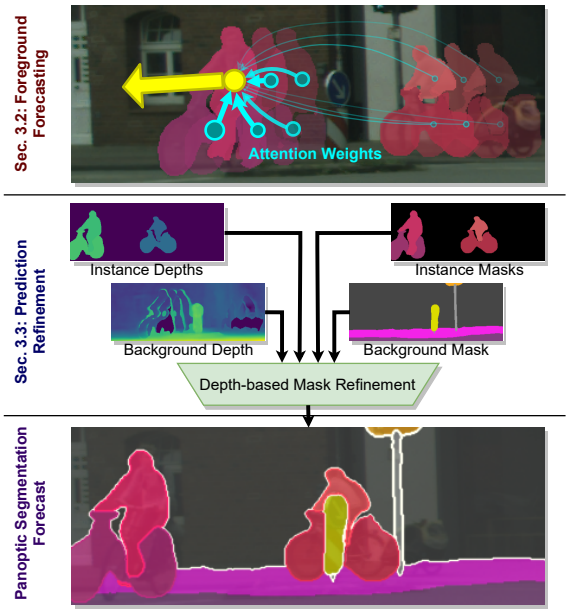

Figure 1. Our panoptic segmentation forecasting. We jointly reason about every instance in a scene to predict instance masks (top), and then reason about the relative depth of foreground and background components (middle) to produce an output (bottom).

1. Introduction

Forecasting is needed for safe and effective autonomous systems [9, 26]. For this reason, forecasting has been studied in many different domains from computer vision and robotics to machine learning. In common across domains is the discussion about what representations are useful for forecasting. Representations which have been studied range from trajectories [10, 11, 13, 32, 53] and bounding boxes [29, 30, 39, 50, 51] to semantic segmentation [6, 25, 28, 34, 36], instance segmentation [8, 19, 27], images [12, 24, 52] and recently also panoptic segmentations [14, 38].

Each representation has applications which benefit from their use. We focus on panoptic segmentations as they naturally disentangle 1) objects which change position in an image due to observer motion; from 2) object instances which change position due to both observer and instance motion.

However, the state-of-the-art on panoptic segmentation forecasting [14] is challenged by two key issues. First, foreground predictions of individual instances are made independently of each other. This is suboptimal because the movements of instances are clearly correlated, *e.g.*, when considering traffic patterns like the ones in the Cityscapes dataset [7]. Second, the method opted for a simple strategy to merge individual object instance segmentation forecasts with the background forecast. Specifically, in [14], object instance segmentation forecasts are always placed in front of the background segmentation forecast. This assumes that no background objects are located closer to the camera than any foreground entity, which is not true in practice.

In this work, we study a new method to address these two issues: 1) To jointly forecast object instance segmentations, we develop a modified attention module for transformer models. Specifically, instead of the inner-product attention in classical transformers, we propose "difference attention." This developed difference attention fits tasks like forecasting because it enables reasoning about velocities and acceleration, which is non-trivial with classical inner-product attention (see Fig. 1 top). 2) To properly reason about object and background placement, we develop a refinement head

which denoises background depth estimates and compares them against foreground predictions (see Fig. 1 middle).

We assess our method on the challenging Cityscapes [7] and AIODrive [46] datasets. We find difference attention and refinement to provide accurate results (see Fig. 1 bottom) which yield a new state-of-the-art of 37.6 PQ for mid-term forecasting on Cityscapes and 48.5 PQ on AIODrive. Code to reproduce results is available via https://github.com/cgraber/psf-diffattn.

2. Related work

Forecasting has been studied across communities [41].

Forecasting of non-semantic representations. Trajectories are arguably one of the representations for which forecasting has been studied most. Trajectories specify the future position of individual objects, either in 2D or 3D [10, 11, 32, 53]. For example, Hsieh *et al*. [18] disentangle position and pose of multiple moving objects – but only on synthetic data. Mittal *et al*. [33] forecast scene flow for point cloud data using self-supervision to reduce training data requirements. Kosiorek *et al*. [22] track instances to forecast their future. Several works have focused on anticipating future pose and location of specific object types, often people [13, 31]. However, arguably, a trajectory forecast provides little beyond position, velocity and acceleration.

To obtain more information, forecasting of future RGB frames has been studied [12, 24, 52]. Due to the high-dimensional space of the forecasts and because of the ambiguity in the forecasts, results often remain blurry, despite significant recent advances. For instance, recent work models uncertainty over future frames using, *e.g*., latent variables [44, 52] or treats foreground and background separately [47]. Moreover, Ye *et al*. [52] forecast future RGB frames by modeling each foreground object separately. Note, all these methods differ from ours in architecture and output: we forecast a semantic representation.

Closer to our work is AgentFormer [54]. It also uses transformers to forecast and introduces an identity encoding via agent-aware attention. Our work differs in that we predict panoptic segmentations while they predict birds-eye-view locations. Additionally, we develop difference attention and auxiliary losses which we find to aid forecasting.

Forecasting semantic segmentations. Recently, methods have been studied to estimate semantic segmentations for future, unobserved frames. Luc *et al*. [28] use a deep-net to estimate a future semantic segmentation given the current RGB frame and its semantics as input. Nabavi *et al*. [34] use recurrent models with semantic maps as input. Chiu *et al*. [6] further use a teacher net to provide an additional supervision during training. Šarić *et al*. [36] use learnable deformations to help forecast future semantics given the observed frames. Lin *et al*. [25] design an autoencoder which 1) compresses input feature pyramids into a low-resolution predictive feature map, 2) predicts this representation for a future frame, and 3) expands it back into a feature pyramid for decoding. However, importantly, these methods do not explicitly consider dynamics of the scene.

While Jin *et al*. [21] jointly predict flow and future semantic segmentations, recent work [37] explicitly warps deep features to obtain a future semantic segmentation. Similarly, Terwilliger *et al*. [40] use a long-short-term-memory (LSTM) module to estimate a flow field which is then used to warp the semantic output of a given input frame. However, by warping in output space, their model has a limited ability to cope with occlusions. While flow improves the modeling of the dynamic world, these methods only consider the dynamics at the pixel-level. Instead, we model dynamics at the object level.

Recent methods [17, 35, 43, 49] estimate future semantic segmentations by reasoning about shape, egomotion, and foreground motion separately. However, none of these methods reason explicitly about individual instances, while our method yields a full future panoptic segmentation forecast, *i.e*., a prediction for every instance.

Forecasting future instance segmentations. Recent methods which forecast an instance segmentation use a conv net or an LSTM module to regress to the deep features which correspond to the future instance segmentation [19, 27]. For example, Couprie *et al*. [8] use a conv net to forecast future instance contours together with an instance-wise semantic segmentation to estimate future instance segmentation. However, their method only estimates foreground and not background semantics.

Unlike these works, we predict both instance segmentation masks for foreground objects and background semantics for future time steps.

Forecasting panoptic segmentations. In recent years, panoptic segmentation has become a popular scene understanding task [3–5, 23, 45]. Very recently [14, 38], it has been proposed as a useful representation for forecasting because it naturally disentangles 1) objects which move in an image just because of observer motion; from 2) object instances which move due to both observer and instance motion.

The state-of-the-art [14] forecasts the future position of individual object instances independently of each other via an encoder-decoder architecture which is executed separately for every object instance. Moreover, the obtained instance forecasts are combined in a heuristic manner by simply pasting objects in front of background without considering depth information of background objects.

In contrast, we propose a method for panoptic segmentation forecasting which jointly forecasts all detected object instances at once via a tailored transformer attention. This helps to benefit from correlations between instances. Moreover, we study how to combine the individual forecasts in a differentiable way. We discuss our method next.

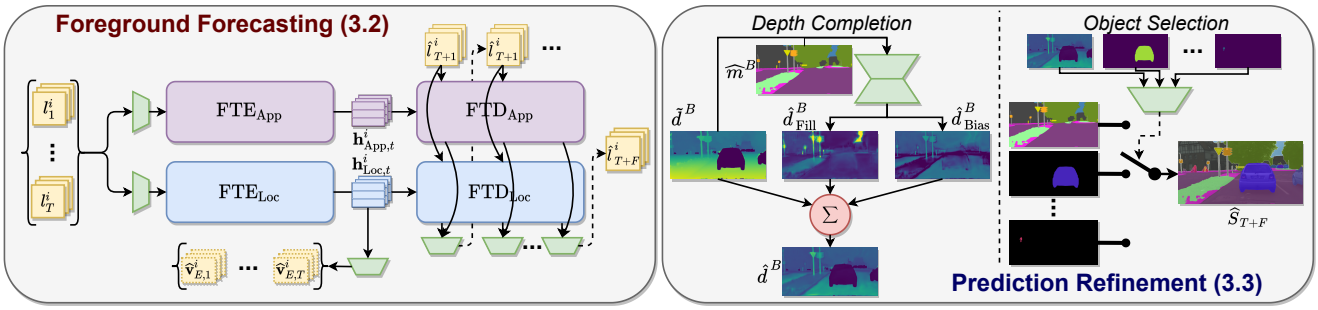

Figure 2. **Method overview**. The foreground forecasting component (Sec. 3.2) predicts future location and appearance jointly for all instances using our newly introduced difference attention transformer. This is followed by a prediction refinement stage (Sec. 3.3), which first completes/denoises input reprojected depths and then uses these along with predicted foreground instance depths to select which object is closest to the camera.

3. Method

In this section, we describe our method for joint forecasting of multiple object instances with the developed difference attention. We start by formalizing the forecasting task and by providing an overview of our approach. Next, we describe the developed difference attention (Sec. 3.1). We use this in our foreground forecasting module, which models interactions between individual instances (Sec. 3.2). Finally, we present the refinement head, which refines the initial foreground instance predictions by considering the background predictions and the depth (Sec. 3.3). An overview of our approach is presented in Fig. 2.

Forecasting task. Given T RGB images $I_1, \dots, I_T$ of height H and width W, panoptic segmentation forecasting aims to predict the panoptic segmentation $\widehat{S}_{T+F}$ corresponding to an unobserved future frame I_{T+F} at a fixed number of timesteps F from the last observation recorded at time T. Each pixel in $\widehat{S}_{T+F}$ is assigned a class $c \in \{1, \dots, C\}$ and an instance ID. In addition to these inputs, we assume access to camera poses $o_1, \dots, o_T$ and depth maps $d_1, \dots, d_T$ for all input frames. We study the use of both camera poses from odometry sensors, and camera poses estimated using visual SLAM [2]. Following [14], we obtain depth maps from input stereo image pairs [15].

Overview. To address forecasting, we follow the paradigm introduced by Graber *et al*. [14]. Specifically, they divide the task into two components: 1) the foreground component, which focuses on 'things' object instances annotated within the dataset; and 2) the background component, which focuses on all annotated 'stuff' object classes. These two components are modeled differently because the causes for the displacement of the corresponding objects in the image plane differ. Specifically, background objects such as buildings and poles shift due to camera motion, while foreground objects like cars and pedestrians move due to both camera motion as well as their own individual motion.

For a fair evaluation, we utilize the same background model as Graber *et al*. [14], who lift background semantics into a 3D point cloud using the estimated input depth, transform the depth based on the target frame camera information, project to the image plane, and refine the projected semantics using a semantic segmentation model. See Appendix A for more details.

However, the approach developed by Graber *et al*. [14] has two primary drawbacks which we correct in this work:

First, their approach to forecast the foreground components of the scene uses an RNN-based encoder-decoder model which models the trajectory of each instance independently of all other instances. This is sub-optimal: in many cases, the movement of individual entities is correlated, *e.g*., due to the flow of traffic. To enable modeling of this correlation, we develop a difference attention module which we detail in Sec. 3.1. It is particularly suitable for forecasting because of its innate ability to reason about the velocities of inputs. We use this difference attention transformer to jointly reason about the future trajectories of all entities in a scene, which we detail in Sec. 3.2.

Second, Graber *et al*. [14] combine foreground and background predictions by "stacking" all predicted foreground instances on top of the predicted background. This approach assumes that no background objects are located closer to the camera than any foreground entity, which is not true in practice. Hence, in Sec. 3.3, we introduce our model to combine foreground and background predictions in a per-pixel fashion by reasoning about their depths.

3.1. Difference Attention for Transformers

To better address forecasting, we develop a difference attention module for transformers. We find this difference attention to be particularly suitable for forecasting because the difference of quantities like locations enables a model to easily reason about velocities and acceleration. In contrast, classical transformer attention is based on inner products which do not naturally encode these quantities.

Formally, the difference attention module operates on two d-dimensional inputs $\mathbf{X}_{\text{self}} \in \mathbb{R}^{M_1 \times d}$ and $\mathbf{X}_{\text{other}} \in \mathbb{R}^{M_2 \times d}$ of lengths M_1 and M_2, respectively, reasons about

the differences between these inputs, and outputs representation $\mathbf{Y} \in \mathbb{R}^{M_1 \times d}$ which encodes these differences. For this, we first compute entity scores

$$\mathbf{Z} = \mathbf{Q}\mathbf{K}_R^T - \mathbf{1}_{M_1 \times 1} \text{diag}\left(\mathbf{K}_B \mathbf{K}_R^T\right)^T, \quad (1)$$

where $\mathbf{1}_{M_1 \times 1}$ is the $M_1 \times 1$ matrix filled with ones. $\mathbf{Q}$ is computed from $\mathbf{X}_{\text{self}}$ and $\mathbf{K}_B$ and $\mathbf{K}_R$ are computed from $\mathbf{X}_{\text{other}}$ with MLPs, *i.e.*, $\mathbf{Q} = f_Q(\mathbf{X}_{\text{self}})$, $\mathbf{K}_R = f_{K_R}(\mathbf{X}_{\text{other}})$, and $\mathbf{K}_B = f_{K_B}(\mathbf{X}_{\text{other}})$. Intuitively, this operation allows the entity score computation to be a function of the difference between the two inputs $\mathbf{X}_{\text{self}}$ and $\mathbf{X}_{\text{other}}$. This is useful for forecasting, as the offset of input locations and their change over time is necessary to understand motion.

Given these entity scores $\mathbf{Z}$, we compute the final attended representation $\mathbf{Y}$ which corresponds to $\mathbf{X}_{\text{self}}$ via

$$\mathbf{Y} = \text{softmax}\left(\mathbf{Z}/\sqrt{d}\right)\mathbf{V}_O - \mathbf{V}_S, \quad (2)$$

where $\mathbf{V}_O = f_{V_O}(\mathbf{X}_{\text{other}})$ and $\mathbf{V}_S = f_{V_S}(\mathbf{X}_{\text{self}})$. Intuitively, this enables the final output $\mathbf{Y}$ to encode the differences between the two inputs $\mathbf{X}_{\text{self}}$ and $\mathbf{X}_{\text{other}}$. This is again suitable for forecasting, as it enables representations to encode the velocity of an instance, which is critical for reasoning about future motion. We now discuss how we use this difference attention for foreground forecasting.

3.2. Foreground Forecasting

Our forecasting model is tasked with predicting a panoptic segmentation $\widehat{S}_{T+F}$ for time $T+F$. This is done by forecasting representations for the N instances in the scene, followed by a final refinement. We represent each instance at all times during forecasting using three components $l_t^i := \{\mathbf{x}_t^i, \mathbf{r}_t^i, p_t^i\}$: a 5-dimensional vector $\mathbf{x}_t^i := [x_0, y_0, x_1, y_1, d]$ representing the upper-left and lower-right corners of the bounding box enclosing instance i as well as the estimated distance of the instance from the camera at time t, a feature tensor $\mathbf{r}_t^i \in \mathbb{R}^{256 \times 14 \times 14}$ representing the visual appearance of the instance at time t, and a binary value $p_t^i \in \{0, 1\}$ which indicates whether instance i is present in frame I_t. Additionally, given background prediction logits $\widehat{m}^B$ and background reprojected depths $\tilde{d}^B$, the final output of the forecasting model is

$$\widehat{S}_{T+F} = \text{Ref}(\text{FD}(\text{FE}(\{l_t^i, c^i, o_t\}_{1:T}^{1:N}), \{o_t\}_{T+1:F}), \widehat{m}^B, \tilde{d}^B). \quad (3)$$

Here, the forecasting encoder FE operates on input representations l_t^i, classes c^i, and odometry o_t $\forall i \in \{1, \ldots, N\}, t \in \{1, \ldots, T\}$ and computes embeddings $\mathbf{h}_{\text{Loc},t}^i$ and $\mathbf{h}_{\text{App},t}^i$ which encode locations and appearances, respectively. The forecasting decoder FD processes these embeddings to autoregressively compute embeddings $\tilde{\mathbf{h}}_{\text{Loc},t}^i$ and $\tilde{\mathbf{h}}_{\text{App},t}^i$, which are used to produce outputs $\widehat{l}_t^i$. These outputs are subsequently combined with background semantics $\widehat{m}^B$ and depths $\tilde{d}^B$ using refinement model Ref to produce the final panoptic segmentation $\widehat{S}_{T+F}$. We discuss the encoder and decoder which use difference attention next, and we detail refinement in Sec. 3.3.

Forecasting Transformer Encoder. The encoder FE produces two embeddings for every instance i at every time t: the first, $\mathbf{h}_{\text{Loc},t}^i \in \mathbb{R}^{d_e}$ where d_e is the size of the embedding, contains information about its location as well as its observed motion; the second, $\mathbf{h}_{\text{App},t}^i \in \mathbb{R}^{256 \times 14 \times 14}$, contains information about its appearance. These are obtained using two newly developed forecasting transformer encoders. The use of transformers for this task permits to jointly reason about every instance both as a function of time and as a function of the other instances present in the scene.

The first transformer encoder produces in parallel $\forall i, t$ the location encoding

$$\{\mathbf{h}_{\text{Loc},t}^i\}_{1:T}^{1:N} = \text{FE}_{\text{Loc}}(\{l_t^i, c^i, o_t\}_{1:T}^{1:N}). \quad (4)$$

For this, it uses all input instances at every point in time, *i.e.*, $\{l_i^t\}_{1:T}^{1:N}$, as well as classes c^i and odometry o_t. Different from classical transformer encoders, FE_{Loc} is trained via auxiliary losses to natively reason about both the velocity of each instance across time as well as the motion of each instance relative to each other. Hence, the embedding $\mathbf{h}_{\text{Loc},t}^i$ is trained to encode information about the velocity, which we show improves the ability of the decoder to anticipate the instances' future motion.

The second transformer encoder, which produces the appearance encoding

$$\mathbf{h}_{\text{App},t}^i = \text{FE}_{\text{App}}(\{l_t^i, c^i, o_t\}_{1:T}^{1:N}), \quad (5)$$

maintains the spatial structure of the input appearance features. This is beneficial for predicting a spatial output.

Both the location and the appearance components of the forecasting transformer encoder are comprised of the same general structure: first, a feature representation for every instance is produced as a function of its location, its appearance, its object class, the current camera motion, and the current time. Second, these feature representations are processed using our customized transformer encoders FTE_{Loc} and FTE_{App}. Letting $\beta \in \{\text{Loc}, \text{App}\}$ denote the modules for the location encoder and the appearance encoder, respectively, this is formally described as

$$\{\mathbf{h}_{\beta,t}^i\}_{1:T}^{1:N} = \text{FE}_\beta(\{l_t^i, c^i, o_t\}_{1:T}^{1:N}) \quad (6)$$

$$\Leftrightarrow \begin{cases} \bar{\mathbf{x}}_{\beta,t}^i = f_\beta(l_t^i, c_i, o_t, t) & \forall i, t \\ \{\mathbf{h}_{\beta,t}^i\}_{1:T}^{1:N} = \text{FTE}_\beta(\{\bar{\mathbf{x}}_{\beta,t}^i\}_{1:T}^{1:N}) \end{cases}, \quad (7)$$

where f_{Loc} uses multilayer perceptrons and f_{App} uses convolutional nets which are described fully in Appx. B. Note, depending on β, Eq. (6) refers to either Eq. (4) or Eq. (5). They perform the computations given in Eq. (7). All features $\{\bar{\mathbf{x}}_{\beta,t}^i\}$ are used as input into the transformer FTE_β.

For FTE_{Loc}, all self-attention modules use the difference attention formulation introduced in Sec. 3.1. This design facilitates the ability of the model to reason about the velocity of the entities, which can be represented by differences in input embeddings which correspond to the same instance at different points in time, as well as the relative offsets between different entities. We find that the use of this form of attention leads to improved forecasting results.

The appearance transformer encoder FTE_{App} is built using convolutional transformers. Specifically, it consists of a transformer whose linear projections have been replaced with convolutional layers. This enables a spatially meaningful representation at all stages during encoding.

For more about attention computation see Appendix C.

Forecasting Transformer Decoder. The decoder utilizes the representations produced by the encoder to predict the future location $\widehat{\mathbf{x}}_i^t$, the future appearance $\widehat{\mathbf{r}}_i^t$, and the future presence $\widehat{p}_i^t$ of each object i for future time steps $t \in \{T+1, \ldots, T+F\}$. Predictions are computed autoregressively, starting with the most recent input locations $\widehat{\mathbf{x}}_i^T := \mathbf{x}_i^T$ and appearance features $\widehat{\mathbf{r}}_i^T := \mathbf{r}_i^T$.

For future time step $t \in \{T+1, \ldots, T+F\}$, both the location decoder FD_{Loc} and the appearance decoder FD_{App} take the following structure, with $\beta \in \{\text{Loc}, \text{App}\}$:

$$\{\tilde{\mathbf{h}}_{\beta,t}^i\}^{1:N} = \text{FD}_\beta(\{\widehat{l}_t^i\}_{T:t-1}^{1:N}, \{c^i\}^{1:N}, \{o_t\}_{T+1:t}, \{\mathbf{h}_{\beta,t}^i\}_{1:T}^{1:N})$$

$$\Leftrightarrow \begin{cases} \tilde{\mathbf{x}}_{\beta,t}^i &= \tilde{f}_\beta(\widehat{l}_t^i, c^i, o_t, t) \\ \{\tilde{\mathbf{h}}_{\beta,t}^i\}_{T+1:T+F}^{1:N} &= \text{FTD}_\beta(\{\tilde{\mathbf{x}}_{\beta,t}^i\}_{T+1:T+F}^{1:N}) \end{cases} \quad \forall i, t \,.$$

Similar to their corresponding encoder modules, the location transformer decoder FTD_{Loc} uses difference attention, the appearance transformer decoder FTD_{App} is a convolutional transformer, and both utilize agent-aware attention.

Final location, appearance, and presence predictions are obtained from the embeddings produced at each time via

$$\widehat{\mathbf{x}}_t^i = f_{\text{LocOut}}(\tilde{\mathbf{h}}_{\text{Loc},t}^i) + \widehat{\mathbf{x}}_{t-1}^i, \tag{8}$$

$$\widehat{p}_t^i = f_{\text{POut}}(\tilde{\mathbf{h}}_{\text{Loc},t}^i), \tag{9}$$

$$\widehat{\mathbf{r}}_t^i = f_{\text{AppOut}}(\tilde{\mathbf{h}}_{\text{App},t}^i), \tag{10}$$

where f_{LocOut} and f_{POut} are multilayer perceptrons and f_{AppOut} is a convolutional network.

Training. The foreground model is trained by providing it with input location and appearance features, predicting the future states of each of these, and regressing against pseudo-ground-truth future locations $\mathbf{x}_t^{*i}$, appearance features $\mathbf{r}_t^{*i}$, and presences p_t^{*i} which are obtained by running instance detection and tracking on future frames. We formally specify the losses in Appendix D.

In addition, we train the forecasting location encoder to estimate the velocity $\widehat{\mathbf{v}}_{\text{E},t}^i$ of each instance via

$$\widehat{\mathbf{v}}_{\text{E},t}^i = f_{\text{vel}}(\mathbf{h}_{\text{Loc},t}^i), \tag{11}$$

where f_{vel} is a multilayer perceptron. This auxiliary prediction task requires the encoder to include information about the motion of each instance within the representation it produces. We find this to lead to better forecasting results.

3.3. Prediction Refinement

To address the aforementioned second shortcoming of [14], we develop a refinement which combines foreground and background predictions as a function of their estimated depth. This allows foreground instances to be placed behind background objects, which yields more natural predictions.

While this would be easy if the depth signal was reliable, the only depth signal we have for the background is the depth of the reprojected points that are used as input for the background prediction model. These depths are both noisy and incomplete, *i.e.*, not every location will correspond to a reprojected point from an earlier frame. Hence, the refinement model has two primary jobs: first, it needs to complete as well as denoise the input depth; second, it needs to select which object is closest based on these depths as well as the depths of foreground instances.

Formally, the refinement head is provided with predicted foreground locations[1] $\widehat{\mathbf{x}}^i$, appearances $\widehat{\mathbf{r}}^i$, and presences $\widehat{p}^i$ for N instances. Given these components, if $p^i = 0$, then instance i is discarded, as the model anticipates that the object is not in frame I_{T+F} due to occlusions or leaving the scene; otherwise, the prediction mask $\widehat{m}^i$ is obtained via

$$\widehat{m}^i = \text{MaskOut}(\widehat{\mathbf{x}}^i, \widehat{\mathbf{r}}^i), \tag{12}$$

where MaskOut predicts a fixed-size mask using Mask R-CNN's mask head and then pastes it into the location specified by $\widehat{\mathbf{x}}^i$. The prediction head additionally uses estimated instance depths $\{\widehat{d}^i\}^{1:N}$, predicted background semantic logits $\widehat{m}^B \in \mathbb{R}^{H \times W \times C_{\text{BG}}}$, where C_{BG} is the number of background classes, the reprojected background depths $\tilde{d}^B \in \mathbb{R}^{H \times W}$, and a binary mask $Q \in \{0,1\}^{H \times W}$ which indicates for each pixel whether or not we have an input background depth. It outputs an object selection map $\widehat{P} \in \{0, \ldots, N\}^{H \times W}$ which specifies, for every pixel, whether the background is in front (represented by value 0) or one of the instances is in front (represented by values 1 through N). We get the final panoptic segmentation via

$$\widehat{S}_{T+F} = \mathbf{1}[\widehat{P} = 0] \arg\max(\widehat{m}^B) + \mathbf{1}[\widehat{P} > 0](\widehat{P} + C_{\text{BG}}).$$

The refinement head is composed of two modules: the first produces completed/denoised background depth prediction $\widehat{d}^B$, and the second uses this alongside the foreground instance information to compute the object selection map $\widehat{P}$. We describe both components next.

Depth completion model. We formulate the depth completion model using two outputs. The first, $\widehat{d}_{\text{Fill}}^B \in \mathbb{R}^{H \times W}$,

[1] For readability, we drop subscript $T{+}F$ for predictions in this section.

represents an initial estimation of the depths for all input locations which are missing a depth, *i.e.*, where $Q = 0$. The second, $\widehat{d}^B_{\text{Bias}} \in \mathbb{R}^{H \times W}$, represents an offset added to the input depths in order to refine and denoise them. Given these predictions, the output of this module is

$$\widehat{d}^B = Q\tilde{d}^B + (1 - Q)\widehat{d}^B_{\text{Fill}} + \widehat{d}^B_{\text{Bias}}, \tag{13}$$

where $\widehat{d}^B_{\text{Fill}}$ and $\widehat{d}^B_{\text{Bias}}$ are obtained using small convolutional networks specified in Appendix E.

Object selection model. Given the completed/denoised background depth prediction $\widehat{d}^B$, object selection determines for every output pixel which object is closest to the camera. We require that this module be fully differentiable such that gradients computed from its outputs can be propagated through to the depth completion model.

First, we describe the motivation behind our formulation. Let $\{d_0, \ldots, d_N\}$ be a set of non-negative depths from which we want to find the lowest value. Assume d_0 is the smallest depth. Then, for all $i, j \in \{0, \ldots, N\}$, $-d_0 d_j \geq -d_i d_j$. In other words, comparing all pairs of depths through multiplication permits to find the smallest.

To encode this, we compute the aggregate depth tensor $\mathbf{D} \in \mathbb{R}^{H \times W \times (N+1)}$ whose 0-th column is the completed background depth $\widehat{d}^B$ and whose i-th column for $i \in \{1, \ldots, N\}$ is $\mathbf{1}[\widehat{m}^i \geq 0.5]d^i + \mathbf{1}[\widehat{m}^i < 0.5]d_{\text{fgmax}}$, where $\mathbf{1}[\cdot]$ is the indicator function applied to all spatial locations in $\widehat{m}^i$ and d_{fgmax} is a large constant. We also construct a value tensor $\mathbf{V} \in \mathbb{R}^{H \times W \times (N+1)}$ whose n-th column is computed by applying a convolutional net to the background logits $\widehat{m}^B$ and foreground probabilities $\widehat{m}^i$. We get

$$\tilde{P}_{i,j} = \text{fullsoftmax}\left(-\mathbf{D}_{i,j}\mathbf{D}^T_{i,j}\right)\mathbf{V}_{i,j}, \tag{14}$$

where $\tilde{P} \in \mathbb{R}^{H \times W \times (N+1)}$ are object selection scores for each pixel, $\widehat{P}_{i,j} = \arg\max \tilde{P}_{i,j}$, and fullsoftmax is the softmax operation which normalizes across both dimensions of its input matrix, *i.e.*, for matrix $\mathbf{X} \in \mathbb{R}^{A \times B}$ we define

$$\text{fullsoftmax}(\mathbf{X})_{i,j} = \frac{e^{\mathbf{X}_{i,j}}}{\sum_{i,j} e^{\mathbf{X}_{i,j}}}. \tag{15}$$

Training. For training, we compute the input instance masks $\widehat{m}^i_{T+F}$ using the pseudo-ground-truth locations $\mathbf{x}^{*i}_t$. We then obtain completed background depths and compute final object selection scores $\tilde{P}_{T+F}$. This is compared to the ground-truth object selection P^*_{T+F} using cross-entropy. We additionally apply a squared norm loss to the predicted depth bias $\widehat{d}^B_{\text{Bias}}$ such that the model is encouraged to trust the input depths where possible.

4. Experiments

We demonstrate that the proposed difference attention and refinement lead to a new state-of-the-art for panoptic segmentation forecasting. We additionally show the contribution each component makes to the final improvement via ablations. In addition, we demonstrate how these improvements carry over to related dense forecasting tasks. Following prior work [14], we test our forecasting model on the Cityscapes dataset [7]. We additionally run experiments on the recently-introduced AIODrive dataset [46].

4.1. Cityscapes

Data. The Cityscapes dataset contains 5,000 sequences of 30 frames each, where ground-truth panoptic segmentations are provided for the 20th frame of each sequence. Here, we evaluate our forecasting model on panoptic segmentation forecasting. Additional results for instance segmentation and semantic segmentation forecasting can be found in Appendix H and Appendix I. We consider two types of forecasting: short-term and mid-term forecasting, each looking 3 and 9 frames into the future respectively. In both cases, we take every third frame as input to our model, hence matching the methods used in prior work [14, 27, 28, 37].

Metrics. Following prior work [14], we consider three metrics: *segmentation quality* (SQ), *recognition quality* (RQ), and *panoptic quality* (PQ). First, we match predicted and target segments, where true positive matches require the intersection over union (IoU) of the two segments to be at least 0.5. SQ corresponds to the average IoU of true matched positive segments. RQ corresponds to the F1 score computed over matches. Finally, PQ is the product of SQ and RQ. These metrics are computed for each individual class and then averaged over all classes.

Baselines. We compare against the baselines introduced in [14]. *Panoptic Deeplab (Oracle)* applies the Panoptic Deeplab model [3] on the target frame, and represents an upper bound on performance due to its access to oracle future information. *Panoptic Deeplab (Last Seen Frame)* applies this model to the most recently observed frame, which represents a model assuming no camera or instance motion. *Flow* computes optical flow [20] from the last two observed frames and then uses it to warp the panoptic segmentation obtained from the last observed frame. *Hybrid Semantic/Instance Forecasting* fuses a semantic segmentation forecast [40] with an instance segmentation forecast [27] to create a panoptic segmentation for the target frame. Finally, IndRNN-Stack is the model introduced by Graber *et al*. [14] which forecasts individual instances using an RNN encoder-decoder model and stacks all foreground components on top of all background components.

Results. The results for all models on the panoptic segmentation forecasting task are presented in Tab. 1. The proposed approach achieves state-of-the-art across both short- and mid-term settings on all metrics when compared to methods which don't access future information.

Fig. 3 presents a visual comparison. IndRNN-Stack is

	Short term: $\Delta t = 3$									Mid term: $\Delta t = 9$								
	All			Things			Stuff			All			Things			Stuff		
	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
Panoptic Deeplab (Oracle)†	60.3	81.5	72.9	51.1	80.5	63.5	67.0	82.3	79.7	60.3	81.5	72.9	51.1	80.5	63.5	67.0	82.3	79.7
Panoptic Deeplab (Last seen frame)	32.7	71.3	42.7	22.1	68.4	30.8	40.4	73.3	51.4	22.4	68.5	30.4	10.7	65.1	16.0	31.0	71.0	40.9
Flow	41.4	73.4	53.4	30.6	70.6	42.0	49.3	75.4	61.8	25.9	69.5	34.6	13.4	67.1	19.3	35.0	71.3	45.7
Hybrid [40] (bg) and [27] (fg)	43.2	74.1	55.1	35.9	72.4	48.3	48.5	75.3	60.1	29.7	69.1	39.4	19.7	66.8	28.0	37.0	70.8	47.7
IndRNN-Stack [14]	49.0	74.9	63.3	40.1	72.5	54.6	55.5	76.7	69.5	36.3	71.3	47.8	25.9	**69.0**	36.2	43.9	72.9	56.2
Ours	**50.2**	**75.7**	**64.3**	**42.4**	**74.2**	**56.5**	**55.9**	**76.8**	**70.0**	**37.6**	**71.4**	**49.5**	**28.6**	**69.0**	**40.1**	**44.1**	**73.2**	**56.4**

Table 1. **Panoptic segmentation forecasting evaluated on the Cityscapes validation set**. † has access to the RGB frame at time $T + F$. Higher is better for all metrics.

Figure 3. Mid-term panoptic segmentation forecasts on Cityscapes. Unlike IndRNN-Stack, our approach is able to properly place foreground instances behind background objects (left two columns). Additionally, our approach models interactions between objects, leading to additional improvements (right column).

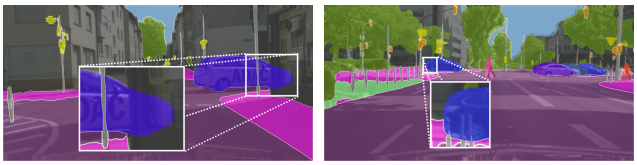

Figure 4. **Failure cases.** Left: the car is incorrectly predicted to be in front of the building on the right. Right: the car is incorrectly predicted to be in front of a few poles.

	$\Delta t = 3$			$\Delta t = 9$		
	PQ	SQ	RQ	PQ	SQ	RQ
Ours	**50.2**	**75.7**	**64.3**	**37.6**	71.4	**49.5**
1) w/o difference attention	49.9	75.6	64.0	36.8	**71.6**	48.3
2) w/o auxiliary encoder loss	49.1	75.3	63.0	36.5	71.5	47.9
3) w/o refinement	49.9	75.6	63.9	36.4	71.0	48.0
4) w/ ORB-SLAM odometry	49.6	**75.7**	63.5	37.2	71.5	49.0
w/ ground truth future odometry	50.5	75.8	64.7	39.7	72.0	52.1

Table 2. **Validating our design choices** using Cityscapes. Higher is better for all metrics. All approaches use predicted future odometry unless otherwise specified.

not capable of placing foreground instances behind background objects, which leads to missing segmentations such as poles in the left column and the street sign in the middle column. Our approach properly reasons about the depth of these objects and places the poles in front of the car and the street sign in front of the cyclist. Additionally, since IndRNN-Stack predicts instances independently, it can make trivial errors such as predicting a cyclist floating away from their bicycle (right column). Our approach, which models interactions among instances and can reason about the fact that cyclists should always move with their bicycles, does not make this error. Additional visualizations comparing these models are presented in Appendix J.

Limitations. Fig. 4 presents a few sequences where our model mispredicts the relative location of foreground and background components. The noisiness of the input point clouds can introduce error in depth reasoning, especially for far away objects which have similar depth. The fact that we only use one depth value for a foreground instance can introduce errors for larger objects. Similar to IndRNN-Stack, the method struggles with instance detection and tracking errors as we assume these inputs to be correct.

Ablations. Tab. 2 summarizes results studying the impact of modeling decisions. 1) *w/o difference attention* uses standard dot product attention for all transformers in place of the difference attention module developed in Sec. 3.1. Our full model's superior performance over 1) demonstrates that the difference attention model is able to better reason about instance motion. 2) *w/o auxiliary encoder loss* trains the forecasting model without applying a loss to the velocity output from Eq. (11). This leads to worse results, and shows that the auxiliary loss helps bias the encoder representations to encode motion information useful for forecasting. 3) *w/o refinement* does not use the refinement head, and instead stacks foreground predictions on top of background predictions, following Graber *et al*. [14]. This leads to missed

	All			Things			Stuff			All			Things		
	PQID	SQID	RQID	PQID	SQID	RQID	PQID	SQID	RQID	PQ	SQ	RQ	PQ	SQ	RQ
Pan. Deeplab †	64.0	84.9	74.4	60.9	79.5	76.3	64.6	85.9	74.0	64.0	84.9	74.4	60.9	79.5	76.3
Pan. Deeplab∗	37.5	**75.8**	47.5	16.1	**70.8**	21.5	41.4	76.7	52.2	37.7	75.5	47.8	17.5	**68.8**	23.8
Flow	40.8	75.7	51.6	19.1	69.9	25.7	44.7	76.7	56.3	40.9	75.4	51.7	19.8	68.5	26.9
IndRNN-Stack	45.1	73.9	57.4	24.1	68.7	32.8	49.0	74.9	61.9	45.3	73.8	57.7	25.1	68.1	34.5
Ours	45.1	73.9	57.2	25.8	69.9	34.3	48.6	74.7	61.4	45.2	73.8	57.5	26.6	69.0	35.8
IndRNN-Stack‡	48.3	75.1	61.1	24.7	68.9	33.3	**52.6**	**76.2**	**66.1**	48.5	**75.0**	61.3	25.8	68.3	35.1
Ours‡	**48.5**	75.1	**61.3**	**26.2**	69.3	**34.9**	**52.6**	76.1	**66.1**	**48.7**	**75.0**	**61.5**	**26.9**	**68.8**	**36.2**

Table 3. Metrics computed on the AIODrive dataset (0.5 second forecast). †: Oracle; ∗: Applied to last seen frame; ‡: use ground-truth depth inputs. IndRNN-Stack and our models use ground-truth odometry input.

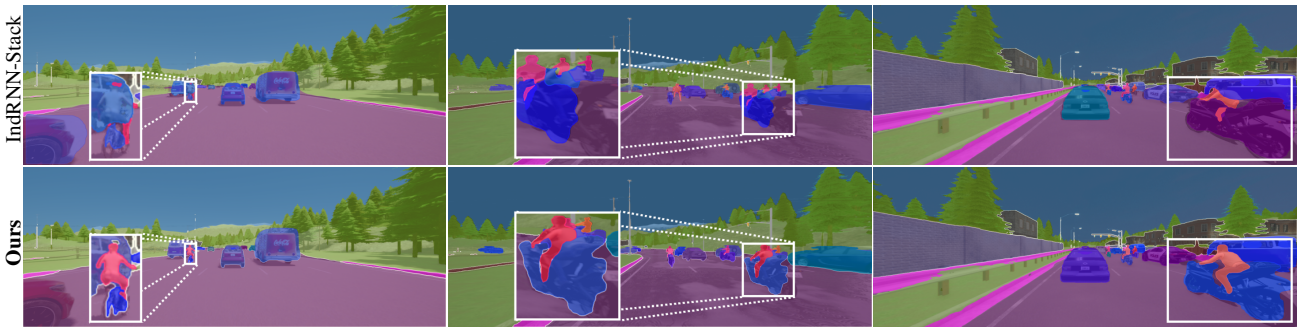

Figure 5. Panoptic segmentation forecasts on the AIODrive dataset. Our approach better captures the relationships between cyclists/their vehicles than IndRNN-Stack, which cannot model these relationships. Also, our model produces higher-fidelity pedestrian instance masks.

background objects which are occluded by foreground predictions, hence a drop in results. 4) *w/ ORB-SLAM odometry* uses input odometry obtained from [2], and shows that our method also works with odometry obtained from image data. The final ablation demonstrates that access to more accurate future camera motion leads to improvements.

4.2. AIODrive

Data. The AIODrive dataset [46] contains a large number of synthetically generated traffic scenarios and provides many inputs and annotations, including stereo images, LiDAR, ground-truth depth maps, panoptic segmentations, and more. The use of a simulator to obtain data and annotations results in AIODrive containing panoptic segmentation annotations, including instance tracks, for all frames. Here, we use the subset of the labels corresponding to Cityscapes classes, consisting of 2 "things" and 11 "stuff" classes. We use 5 frames of input and forecast the 5th frame into the future (corresponding to a 0.5s forecast). Additional details can be found in Appendix G.

Metrics. In addition to previously used metrics, we introduce metrics which account for object identity. Specifically, we evaluate using PQID, SQID, and RQID, which require matches computed between predicted and ground-truth objects to have the same instance ID. These metrics are more appropriate for the forecasting setting due to the fact that the previously used metrics can mark matches between different instances as true positives, meaning the motion of an instance was incorrectly predicted but the metric did not properly evaluate this. Note that we cannot compute these metrics on Cityscapes, as that data only contains annotations for a single frame per sequence.

Results. The results for all models on the panoptic segmentation forecasting task on AIODrive are presented in Tab. 3. Because the All PQID is averaged over 2 "things" classes and 11 "stuff" classes, this metric is biased towards "stuff" performance. Hence, All PQID is comparable between IndRNN-Stack and our model. However, the differences are much clearer on the "things" metrics, as our approach is better able to reason about the motion of individual object instances. Furthermore, there is a small drop in performance between PQID and PQ, indicating that some of the true positive matches found when computing PQ are between incorrect ground-truth instances. Fig. 5 shows results for our method and IndRNN-Stack on AIODrive. Our approach produces better forecasts for cyclists and their bikes, due to the use of difference attention.

5. Conclusion

We introduce a new model for panoptic segmentation forecasting. It uses difference attention which we find to be more suitable to forecasting than standard attention as it can reason about velocities and acceleration. A new refinement head also merges predictions based on depth. This improves prior work on all panoptic forecasting metrics.

Acknowledgements: This work is supported in part by NSF #1718221, 2008387, 2045586, 2106825, MRI #1725729, NIFA 2020-67021-32799 and Cisco Systems Inc. (CG 1377144 - thanks for access to Arcetri).

References

[1] Apratim Bhattacharyya, Mario Fritz, and Bernt Schiele. Bayesian prediction of future street scenes using synthetic likelihoods. *ICLR*, 2019. 7

[2] Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *arXiv preprint arXiv:2007.11898*, 2020. 3, 8, 5

[3] Bowen Cheng, Maxwell D. Collins, Yukun Zhu, Ting Liu, Thomas S. Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020. 2, 6

[4] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *arXiv*, 2021. 2

[5] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. 2021. 2

[6] Hsu-Kuang Chiu, Ehsan Adeli, and Juan Carlos Niebles. Segmenting the future. *IEEE Robotics and Automation Letters*, 2020. 1, 2

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 2, 6

[8] Camille Couprie, Pauline Luc, and Jakob Verbeek. Joint future semantic and instance segmentation prediction. In *ECCV workshops*, 2018. 1, 2

[9] Kenneth James Williams Craik. *The Nature of Explanation*. Cambridge University Press, 1943. 1

[10] Qi Dai, Vaishakh Patil, Simon Hecker, Dengxin Dai, Luc Van Gool, and Konrad Schindler. Self-supervised object motion and depth estimation from video. In *CVPR Workshops*, 2020. 1, 2

[11] Sebastien Ehrhardt, Oliver Groth, Aron Monszpart, Martin Engelcke, Ingmar Posner, Niloy Mitra, and Andrea Vedaldi. RELATE: Physically plausible multi-object scene synthesis using structured latent spaces. In *NeurIPS*, 2020. 1, 2

[12] Hang Gao, Huazhe Xu, Qi-Zhi Cai, Ruth Wang, Fisher Yu, and Trevor Darrell. Disentangling propagation and generation for video prediction. In *ICCV*, 2019. 1, 2

[13] Colin Graber and Alexander Schwing. Dynamic neural relational inference. In *CVPR*, 2020. 1, 2

[14] Colin Graber, Grace Tsai, Michael Firman, Gabriel Brostow, and Alexander G. Schwing. Panoptic segmentation forecasting. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7

[15] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *CVPR*, 2020. 3, 6

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 6

[17] Lukas Hoyer, Patrick Kesper, Anna Khoreva, and Volker Fischer. Short-term prediction and multi-camera fusion on semantic grids. In *ICCV Workshops*, 2019. 2

[18] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *NeurIPS*, 2018. 2

[19] Jian-Fang Hu, Jiangxin Sun, Zihang Lin, Jian-Huang Lai, Wenjun Zeng, and Wei-Shi Zheng. Apanet: Auto-path aggregation for future instance segmentation prediction. *IEEE TPAMI*, 2021. 1, 2

[20] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017. 6

[21] Xiaojie Jin, Huaxin Xiao, Xiaohui Shen, Jimei Yang, Zhe Lin, Yunpeng Chen, Zequn Jie, Jiashi Feng, and Shuicheng Yan. Predicting scene parsing and motion dynamics in the future. In *NeurIPS*, 2017. 2

[22] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *NeurIPS*, 2018. 2

[23] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *CVPR*, 2021. 2

[24] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion GAN for future-flow embedded video prediction. In *ICCV*, 2017. 1, 2

[25] Zihang Lin, Jiangxin Sun, Jian-Fang Hu, Qizhi Yu, Jian-Huang Lai, and Wei-Shi Zheng. Predictive feature learning for future segmentation prediction. In *ICCV*, 2021. 1, 2, 6, 7

[26] Rodolfo R. Llinás. *I of the vortex: from neurons to self*. MIT Press, 2001. 1

[27] Pauline Luc, Camille Couprie, Yann LeCun, and Jakob Verbeek. Predicting future instance segmentation by forecasting convolutional features. In *ECCV*, 2018. 1, 2, 6, 7, 8, 9, 12

[28] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *ICCV*, 2017. 1, 2, 6

[29] Osama Makansi, Özgün Cicek, Yassine Marrakchi, and Thomas Brox. On Exposing the Challenging Long Tail in Future Prediction of Traffic Actors. In *ICCV*, 2021. 1

[30] Srikanth Malla, Behzad Dariush, and Chiho Choi. Titan: Future forecast using action priors. In *CVPR*, 2020. 1

[31] Karttikeya Mangalam, Ehsan Adeli, Kuan-Hui Lee, Adrien Gaidon, and Juan Carlos Niebles. Disentangling human dynamics for pedestrian locomotion forecasting with noisy supervision. In *WACV*, 2020. 2

[32] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *CVPR*, 2017. 1, 2

[33] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, 2020. 2

[34] Seyed Shahabeddin Nabavi, Mrigank Rochan, and Yang Wang. Future semantic segmentation with convolutional LSTM. In *BMVC*, 2018. 1, 2

[35] Xiaojuan Qi, Zhengzhe Liu, Qifeng Chen, and Jiaya Jia. 3D motion decomposition for RGBD future dynamic scene synthesis. In *CVPR*, 2019. 2

[36] Josip Šarić, Marin Oršić, Tonći Antunović, Sacha Vražić, and Siniša Šegvić. Single level feature-to-feature forecasting with deformable convolutions. In *German Conference on Pattern Recognition*, 2019. 1, 2, 7

[37] Josip Saric, Marin Orsic, Tonci Antunovic, Sacha Vrazic, and Sinisa Segvic. Warp to the future: Joint forecasting of features and feature motion. In *CVPR*, 2020. 2, 6, 7, 8

[38] Josip Šarić, Sacha Vražić, and Siniša Šegvić. Dense semantic forecasting in video by joint regression of features and feature motion. *IEEE TNNLS*, 2021. 1, 2

[39] Oliver Styles, Victor Sanchez, and Tanaya Guha. Multiple object forecasting: Predicting future object locations in diverse environments. In *WACV*, 2020. 1

[40] Adam Terwilliger, Garrick Brazil, and Xiaoming Liu. Recurrent flow-guided semantic forecasting. In *WACV*, 2019. 2, 6, 7, 8, 9, 12

[41] Eugene Valassakis. Future object segmentation for complex correlated motions. Master's thesis, UCL, 2018. 2

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan. N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2, 3

[43] Suhani Vora, Reza Mahjourian, Soeren Pirk, and Anelia Angelova. Future semantic segmentation using 3D structure. *arXiv:1811.11358*, 2018. 2

[44] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert. An uncertain future: Forecasting from static images using variational autoencoders. In *ECCV*, 2016. 2

[45] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021. 2

[46] Xinshuo Weng, Yunze Man, Jinhyung Park, Ye Yuan, Dazhi Cheng, Matthew O'Toole, and Kris Kitani. All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. *arXiv*, 2021. 2, 6, 8

[47] Yue Wu, Rongrong Gao, Jaesik Park, and Qifeng Chen. Future video synthesis with object motion prediction. In *CVPR*, 2020. 2

[48] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *ICML*, 2020. 2

[49] Jingwei Xu, Bingbing Ni, Zefan Li, Shuo Cheng, and Xiaokang Yang. Structure preserving video prediction. In *CVPR*, 2018. 2

[50] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *CVPR*, 2018. 1

[51] Yu Yao, Mingze Xu, Chiho Choi, David J Crandall, Ella M Atkins, and Behzad Dariush. Egocentric vision-based future vehicle localization for intelligent driving assistance systems. In *ICRA*, 2019. 1

[52] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *ICCV*, 2019. 1, 2

[53] Raymond A. Yeh, Alexander G. Schwing, Jonathan Huang, and Kevin Murphy. Diverse Generation for Multi-agent Sports Games. In *CVPR*, 2019. 1, 2

[54] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *ICCV*, 2021. 2, 3