Appendix: Panoptic Segmentation Forecasting

In this appendix, we first provide additional details on the classes contained within Cityscapes (Sec. A). This is followed by details deferred from the main paper for space reasons, including a description of how odometry is used by each component (Sec. B), formal descriptions of the background modeling approach (Sec. C), the aggregation method (Sec. D), used losses (Sec. E), and miscellaneous implementation details for all model components and baselines (Sec. F). This is followed by additional experimental results (Sec. G), including per-class panoptic segmentation metrics as well as metrics computed on test data for the tasks of panoptic, semantic, and instance segmentation forecasting. Finally, we present additional visualizations of model predictions (Sec. H).

Included in the supplementary material are videos visualizing results and a few of the baselines. These are described in more detail in Sec. H.

A. 'Things' and 'Stuff' class breakdown in Cityscapes

The data within Cityscapes are labeled using 19 semantic classes. The 'things' classes are those that refer to individual objects which are potentially moving in the world and consist of person, rider, car, truck, bus, train, motorcycle, and bicycle. The remaining 11 classes are the 'stuff' classes and consist of road, sidewalk, building, wall, fence, pole, traffic light, traffic sign, vegetation, terrain, and sky.

B. Odometry

Cityscapes provides vehicle odometry o_t at each frame t as $[v_t, \Delta\theta_t]$, where v_t is the speed and $\Delta\theta_t$ is the yaw rate. This odometry assumes the vehicle is moving on a flat ground plane with no altitude changes. For background forecasting (Sec. 3.1.2), a full 6-dof transform H_t from frame t and the target frame $T+F$ is required for projecting the 3d semantic point cloud $(\tilde{m}_t^B, \tilde{d}_t^B)$.

To derive H_t, we use the odometry readings $o_{1,\dots,T+F}$ from frame t to target frame $T+F$, the time difference between consecutive frames $\Delta t_{1,\dots,T+F}$, and the camera extrinsics $H_{\text{veh}}^{\text{cam}}$ (i.e., transformation from the vehicle's coordinate system to the camera coordinate system) provided by Cityscapes. First, we compute the 3-dof transform between consecutive frames t and $t+1$ from o_t and Δt_{t+1} by applying a velocity motion model [60], typically applied to a mobile robot. More specifically,

$$\theta_t = \Delta\theta_t \cdot \Delta t_{t+1}, \tag{11}$$

$$r_{t+1} = v_t/(\Delta\theta_t), \tag{12}$$

$$x_{t+1} = r_{t+1} \cdot \sin\theta_t, \tag{13}$$

$$y_{t+1} = r_{t+1} - r_{t+1} \cdot \cos\theta_t, \tag{14}$$

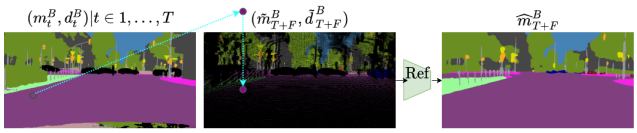

Figure 6. **'Stuff' forecasting is achieved in two steps.** First, for each input frame $t \in \{1,\dots,T\}$ given depth map d_t^B and egomotion, project the semantic map m_t^B for all the background pixels on to frame $T+F$. Note the sparsity of the projected semantic maps. Second, we apply a refinement network to complete the background segmentation.

where (x_{t+1}, y_{t+1}) is the location of the vehicle at time $t+1$ treating the vehicle at time t as the origin (x-axis is front, y-axis is left, and z-axis is up), and θ_t is the rotation of the vehicle at time t along the z-axis. We then extend (x, y, θ) into a 6-dof transform H_t^{t+1}, and apply the camera extrinsics $H_{\text{veh}}^{\text{cam}}$ to obtain the transform of the cameras from frame t to frame $t+1$ via,

$$H_{\text{veh},t} = \begin{bmatrix} \cos\theta_t & -\sin\theta_t & 0 & x_{t+1} \\ \sin\theta_t & \cos\theta_t & 0 & y_{t+1} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{15}$$

$$H_t^{t+1} = H_{\text{veh}}^{\text{cam}} H_{\text{veh},t}^{-1} (H_{\text{veh}}^{\text{cam}})^{-1}. \tag{16}$$

The final transform H_t^{T+F} from frame t to frame $T+F$ is obtained by concatenating the consecutive transforms H_t^{t+1} for $t \in \{1,\dots,T+F-1\}$.

For egomotion estimation (Sec. 3.1.4), o_t is used as input for frames $t \in \{1,\dots,T\}$ and predicted for frames $t \in \{T+1,\dots,T+F\}$. For 'things' forecasting (Sec. 3.1.1), the input egomotion vector consists of v_t, θ_t, x_{t+1}, y_{t+1}, and θ_{t+1}, i.e., the speed and yaw rate of the vehicle as well as how far the vehicle moves in this time step.

C. Detail on background modelling

Here we give more details on our background modelling, to ensure reproducibility.

The background model is summarized in Fig. 6. More formally, the background model estimates the semantic segmentation of an unseen future frame via

$$\widehat{m}_{T+F}^B = \text{ref}(\{\,\text{proj}(m_t, d_t, K, H_t, u_t)\,\}|_{t=1}^{T}), \tag{17}$$

where K subsumes the camera intrinsics, H_t is the 6-dof camera transform from input frame t to target frame $T+F$, m_t is the semantic estimate at frame t obtained from a pretrained semantic segmentation model, d_t is the input depth map at time t, and u_t denotes the coordinates of all the pixels not considered by the foreground model. Here, *proj* refers to the step that creates the sparse projected semantic

map at frame $T+F$ from an input frame t, and *ref* refers to a refinement model that completes the background segmentation.

The refinement model *ref* receives a reprojected semantic point cloud $(\tilde{m}_t^B, \tilde{d}_t^B)$ from applying *proj* to each input frame at time $t \in \{1, \dots, T\}$. For *proj*, each input frame comes with a per-pixel semantic label m_t and a depth map d_t obtained through a pre-trained model. We back-project, transform, and project the pixels from an input frame to the target frame. This process can be summarized as

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = H_t \begin{bmatrix} K^{-1} \begin{bmatrix} u_t \\ 1 \end{bmatrix} D_t \\ 1 \end{bmatrix}, \tag{18}$$

$$\begin{bmatrix} u_{T+F} \\ 1 \end{bmatrix} = K \begin{bmatrix} x_t/z_t \\ y_t/z_t \\ 1 \end{bmatrix}, \tag{19}$$

$$\tilde{m}_t^B(u_{T+F}) = m_t^B(u_t), \tag{20}$$

$$\tilde{d}_t^B(u_{T+F}) = z_t, \tag{21}$$

where u_t is a vector of the pixel locations for all background locations in the image at time t, K subsumes the camera intrinsics, D_t is a diagonal matrix whose entries contain depth d_t of the corresponding pixels, and u_{T+F} is the vector containing corresponding pixel locations for the image at time $T+F$. We maintain the per-pixel semantic class obtained from the frame m_t, and the projected depth. Note, in this process, if multiple pixels u_t from an input frame are projected to the same pixel u_{T+F} in the target frame, we keep the depth and semantic label of the one with the smallest depth value (i.e., closest to the camera).

D. Foreground and background aggregation steps

Alg. 1 describes our aggregation steps in detail. We initialize the output using the predicted background segmentation. After this, instances are sorted in reverse order of depth, and then they are placed one by one on top of the background.

Algorithm 1 Foreground and background aggregation

Input: Background semantics m^B_{T+F};
Foreground segmentations m^i_{T+F}, classes c^i,
depths $d^i_{T+F}, i = 1, \dots, N$;
for $(x, y) \in \{1, \dots, W\} \times \{1, \dots, H\}$ **do**
  $S_{T+F}(x, y) \leftarrow [m^B_{T+F}(x, y), 0]$
end for
$\sigma \leftarrow \text{ArgSortDescending}(d^1_{T+F}, \dots, d^N_{T+F})$
for $i \in \sigma$ **do**
  for $(x, y) \in \{1, \dots, W\} \times \{1, \dots, H\}$ **do**
    if $m^i_{T+F} = 1$ **then**
      $S_{T+F}(x, y) \leftarrow [c^i, i]$
    end if
  end for
end for
Return: future panoptic segmentation S_{T+F}

E. Losses

Here we formally describe the losses used to train the model.

E.1. 'Things' forecasting loss

For instance i we use $\mathcal{L}^i_{\text{fg}} :=$

$$\frac{1}{Z_i} \sum_{t=T}^{T+F} p(t,i) \left(\lambda \text{SmoothL1}(\hat{\mathbf{x}}^i_t, \mathbf{x}^i_t) + \text{MSE}(\hat{\mathbf{r}}^i_t, \mathbf{r}^i_t) \right), \tag{22}$$

where $Z_i = \sum_{t=T}^{T+F} p(t,i)$ is a normalization constant and $p(t,i)$ equals 1 if we have an estimate for instance i in frame t while being 0 otherwise. MSE refers to mean squared error, i.e., $\text{MSE}\left(\hat{\mathbf{r}}^i_t, \mathbf{r}^i_t\right)) := \frac{1}{J} \sum_{j=1}^{J} \left(\hat{\mathbf{r}}^i_t - \mathbf{r}^i_t\right)^2$, while SmoothL1 is given by

$$\text{SmoothL1}(\mathbf{a}, \mathbf{b}) := \frac{1}{J} \sum_{j=1}^{J} \text{SmoothL1Fn}(\mathbf{a}^j, \mathbf{b}^j) \tag{23}$$

$$\text{SmoothL1Fn}(a, b) := \begin{cases} \frac{1}{2}(a-b)^2, & \text{if } |a-b| < 1, \\ |a-b| - \frac{1}{2}, & \text{otherwise,} \end{cases} \tag{24}$$

where $\mathbf{a}$ and $\mathbf{b}$ are vector-valued inputs and a and b are scalars. We use $\lambda = 0.1$, which was chosen to balance the magnitudes of the two losses.

E.2. 'Stuff' forecasting loss

To train the refinement network we use the cross-entropy loss

$$\mathcal{L}_{\text{bf}} := \frac{1}{\sum_{x,y} \mathbf{1}^{\text{bg}}_t[x,y]} \sum_{x,y} \mathbf{1}^{\text{bg}}_t[x,y] \sum_c m^{i*}_t(x,y,c) \log\left(p^i_t(x,y)\right). \tag{25}$$

Here, $\mathbf{1}^{\text{bg}}_t[x,y]$ is an indicator function which specifies whether pixel coordinates (x,y) are in the background of frame t, and $m^{i*}_t(x,y,c) = 1$ if c is the correct class for pixel (x,y) and 0 otherwise. Other variables are as described in the main paper.

F. Further implementation details

F.1. 'Things' forecasting model

For instance detection, we use the pretrained MaskR-CNN model provided by Detectron2[1], which is first trained

[1] https://github.com/facebookresearch/detectron2

on the COCO dataset [39] and then finetuned on Cityscapes. For all detections, we extract the object bounding box and the $256 \times 14 \times 14$ feature tensor extracted after the ROIAlign stage. The detected instances are provided to DeepSort [65] to retrieve associations across time. We use a pre-trained model[2] which was trained on the MOT16 dataset [47]. The tracker is run on every 30 frame sequence once for every 'things' class (in other words, the tracker is only asked to potentially link instances of the same class as determined by instance detection).

Both GRU_{enc} and GRU_{dec} are 1-layer GRUs with a hidden size of 128. Both $\text{ConvLSTM}_{\text{enc}}$ and $\text{ConvLSTM}_{\text{enc}}$ are 2-layer ConvLSTM networks using 3×3 kernels and 256 hidden channels. $f_{\text{enc,b}}$ and f_{bbox} are 2-layer multilayer perceptrons with ReLU activations and a hidden size of 128. f_{bfeat} is a linear layer that produces a 16-dimensional output, which is then copied across the spatial dimensions to form a $16 \times 14 \times 14$ tensor and concatenated with the mask feature tensor along the channel dimension before being used as input. f_{mfeat} is a 1×1 convolutional layer producing a 8 channel output, which is followed by a ReLU nonlinearity and a linear layer which produces a 64-dimensional output vector. $f_{\text{enc,m}}$ and f_{mask} are 1×1 convolutional layers producing a 256 channel output. MaskOut is the mask head from MaskRCNN, and consists of 4 3×3 convolutional layers with output channel number 256, each followed by ReLU nonlinearities, a 2×2 ConvTranspose layer, and a final 1×1 convolutional layer that produces an 8 channel output. Each channel represents the output for a different class, and the class provided as input is used to select the proper mask. The Mask Head's parameters are initialized from the same pre-trained model as used during instance detection and are fixed during training.

During training, individual object tracks were sampled from every video sequence and from every 18-frame subsequence from within each video. The tracking model we used frequently made ID switch errors for cars which the ego vehicle was driving past; specifically, the tracker would label a car visible to the camera as being the same car that the recording vehicle drove past a few frames previously. This had the effect of causing some tracks to randomly "jump back" into the frame after having left. In an attempt to mitigate these errors, instance tracks belonging to cars were truncated if this behavior was detected in an input sequence. Specifically, if a given car track, after being located within 250 pixels from the left or right edge of a previous input frame, moved towards the center of the frame by more than 20 pixels in the current frame, it was discarded for this and all future frames. Since this led to incomplete tracks in some cases, car tracks located within 200 pixels from the left or right side of the frame were augmented by estimating their velocity from previous inputs and linearly extrapolating their locations until they were no longer present in the frame.

We trained for 200000 steps using a batch size of 32, a learning rate of 0.0005, and the Adam optimizer. Gradients were clipped to a norm of 5. The learning rate was decayed by a factor of 0.1 after 100000 steps. During training, ground-truth odometry was used as input; odometry predictions were used for future frames for all evaluations except for the ablation which is listed as having used ground-truth future odometry. Bounding box features and odometry were normalized by their means and standard deviations as computed on the training data. During evaluation, we filtered all sequences which did not detect the object in the most recent input frame (i.e., $T = 3$), as this led to improved performance.

F.2. 'Stuff' forecasting model

We use the model described by Zhu *et al*. [73] with the SEResNeXt50 backbone and without label propagation as our single frame semantic segmentation model[3]. For our refinement model, we use a fully convolutional variant of the HarDNet architecture [7] which is set up to predict semantic segmentations[4]. We initialize with pre-trained weights and replace the first convolutional layer with one that accepts a 60 channel input (3 input frames, each consisting of a 19 channel 1-hot semantic input and a 1 channel depth input). We trained a single model for both the short- and mid-term settings; hence, for each sequence in the training data, we use two samples: one consisting of the 5th, 8th, and 11th frames (to represent the mid-term setting) and one consisting of the 11th, 14th, and 17th frames (to represent the short-term setting), where frame 20 was always the target frame. During training, ground-truth odometry was used to create the point cloud transformations; during evaluation unless specified otherwise, the predicted odometry was used for future frames.

We trained the 'stuff' forecasting model for 90000 steps using a batch size of 16, a learning rate of 0.002, weight decay 0.0001, and momentum 0.9. Gradients were clipped to a norm of 5. The learning rate was decayed by a factor of 0.1 after 50000 steps. During training, we randomly resized inputs and outputs between factors of $[0.5, 2]$ before taking 800×800 crops. Input depths were clipped to lie between $[0.1, 200]$ and normalized using mean and standard deviation computed on the training set.

F.3. Egomotion estimation

The egomotion estimation model takes as input $o_t := [v_t, \theta_t], t \in \{1, \ldots, T\}$, where v_t is the speed and θ_t is the yaw rate of the ego-vehicle. It is tasked to predict

[2] https://github.com/nwojke/deep_sort

[3] https://github.com/NVIDIA/semantic-segmentation/tree/sdcnet

[4] https://github.com/PingoLH/FCHarDNet

	All			Things			Stuff		
	PQ	SQ	RQ	PQ	SQ	RQ	PQ	SQ	RQ
Flow	25.6	70.1	34.0	12.4	66.3	18.1	35.3	72.9	45.5
Hybrid [59] (bg) and [43] (fg)	29.4	69.8	38.5	18.0	67.2	25.7	37.6	71.6	47.8
Ours	**35.7**	**72.0**	**46.5**	**24.0**	**69.0**	**33.7**	**44.2**	**74.2**	**55.8**

Table 5. **Panoptic segmentation forecasting evaluated on the Cityscapes test set, mid-term.** Higher is better for all metrics.

	Short term: $\Delta t = 3$		Mid term: $\Delta t = 9$	
Accuracy (mIoU)	All	MO	All	MO
F2MF [53]*	70.2	68.7	59.1	56.3
Ours	67.3	58.8	57.7	48.8

Table 6. **Semantic segmentation forecasting results on the Cityscapes test dataset.** Baseline numbers, are from [53]; the * indicates training on both train and validation data. Higher is better for all metrics.

	Short term: $\Delta t = 3$		Mid term: $\Delta t = 9$	
	AP	AP50	AP	AP50
F2F [43]	-	-	6.7	17.5
Ours	14.9	31.3	**8.4**	**19.8**

Table 7. **Instance segmentation forecasting on the Cityscapes Test dataset.** Higher is better for all metrics.

$\widehat{o}_{T+1}, \ldots, \widehat{o}_{T+F}$. Unlike the other components of our approach, the egomotion estimation model does not subsample inputs – hence, it takes 9 steps of input and predicts the odometry for the next 9 steps. GRU_{cam} is a 1-layer GRU with a hidden size of 128. f_{cam} is a linear layer.

We trained the egomotion estimation model for 80000 steps using a batch size of 32, a learning rate of 0.0005, and the Adam optimizer. Gradients were clipped to a norm of 5. Inputs were normalized using means and standard deviations computed on the training set.

F.4. Training and Inference time

During inference, our unoptimized code makes a prediction in about 700ms using one 32GB NVIDIA V100. Additional engineering efforts can reduce this time further. Training the foreground and background models takes approximately 12 and 18 hours, respectively, on the same GPU.

F.5. ORB-SLAM details

We ran ORB-SLAM3 [6][5] with stereo images to obtain ego-motion in our ablation study. Each sequence (30 frames) is treated as its own SLAM session providing 6-dof poses for all the frames in the sequence. We then converted the poses into speed and yaw rate for each frame according to their timestamps.

F.6. Monocular depth details

Our experiments in Tab. 2 show we can still achieve good performance even if we don't have stereo pairs at test time, through the use of a monocular depth estimation framework. For this, we used the codebase of [19] to finetune a model for depth estimation on Cityscapes.

We initialised this model with the authors' weights trained on the KITTI dataset [18]. We used their high-resolution Resnet-18 model, which was trained with a KITTI-image input size of 1024×320 pixels[6]. We then finetuned this model on the training split of the Cityscapes dataset. For Cityscapes finetuning we trained on the full Cityscapes input image without any cropping, and we trained using self-supervised reprojection losses using the stereo pairs as supervision (we did not use the monocular sequences, though we expect including these at training could further improve scores). We resized our Cityscapes input images to 640×320 pixels at both training and test time. We used a smaller image input size than the model was trained for on KITTI to account for the change in image aspect ratio between KITTI and Cityscapes. We trained our model with Adam [31] for 15 epochs with a learning rate of $1e-4$ and then 15 epochs with $1e-5$.

F.7. Flow Baseline

To implement this model, we warp the panoptic segmentation computed by the oracle model on the most recently seen input frame using optical flow. Specifically, we compute the forward optical flow between the most recent input frames, i.e., between frames for $t \in \{2, 3\}$, using the CSS configuration of the model introduced by Ilg *et al*. [27][7]. We then iteratively warp the inputs via the following procedure: first, we warp the optical flow using itself (to align it with the current frame), and then we warp the current panoptic segmentation using the resulting flow. For short-term, this is done once, and for mid-term, this is done three times. This was the best performing procedure we tried. Negating reverse optical flow, extrapolating the flow linearly for the mid-term setting, or a combination of the two resulted in worse performance.

F.8. Hybrid Semantic/Instance Forecasting Model

The semantic segmentation forecasting component of the hybrid baseline consists of the model developed by Terwilliger *et al*. [59], which anticipates the optical flow be-

[5] https://github.com/UZ-SLAMLab/ORB_SLAM3

[6] Available from https://github.com/nianticlabs/monodepth2

[7] https://github.com/NVIDIA/flownet2-pytorch

	$\Delta t = 3$			$\Delta t = 9$		
	PQ	SQ	RQ	PQ	SQ	RQ
Ours	**49.0**	74.9	**63.3**	**36.3**	71.3	**47.8**
a) fg w/ independent RNNs	**49.2**	**75.2**	63.3	35.7	71.2	46.9
b) fg w/o velocity features	**49.2**	75.0	**63.4**	35.7	71.2	47.0

Table 8. **Additional ablations** on Panoptic segmentation forecasting using Cityscapes. Higher is better for all metrics. All approaches use predicted future odometry.

tween the most recently seen input frame and the target frame and uses it to warp the predicted semantic segmentation for that input frame. The authors originally evaluated on four settings: $\Delta t = 1$, $\Delta t = 3$, $\Delta t = 9$, and $\Delta t = 10$. Their official implementation[8] provides pre-trained models for $\Delta t = 3$ and $\Delta t = 10$; these are what we use for our short- and mid-term settings, respectively. Note, that Terwilliger *et al*. [59] don't provide a model for $\Delta t = 9$ or the code they used to train their models. Despite the fact that the $\Delta t = 10$ setting is marginally more challenging than the $\Delta t = 9$ setting, Terwilliger *et al*. report better performance on the $\Delta t = 10$ setting [59], which is why we felt comfortable using this approach for our hybrid model. The best performing semantic segmentation forecasting approach, developed by Šarić *et al*. [53], did not have code or models available at time of submission.

The instance segmentation forecasting component of the hybrid baseline consists of the model developed by Luc *et al*. [43], which anticipates the image features for the target frame autoregressively. We use their official implementation[9], which provides pre-trained models for both the short- and mid-term settings.

Predictions are fused by 'pasting' all 'thing' class predictions made by the instance segmentation forecasting model on top of the 'stuff' class predictions made by the semantic segmentation forecasting model. Note that this leaves some pixels without predictions; this occurs for pixels where the semantic forecasting model predicted a 'thing' class but the instance forecasting model did not predict a 'thing' class. We implemented a version of this model which filled in these gaps with the closest 'stuff' class predicted by the semantic forecasting model, but this performed worse.

G. More Experimental Results

Tab. 5 provides the panoptic segmentation forecasting metrics computed on the Cityscapes test dataset for our method as well as the flow and hybrid baselines for the mid-term setting. We outperform both other approaches on this data as well for all metrics.

Tab. 6 shows the semantic segmentation forecasting metrics computed on the Cityscapes test dataset for our approach as well as F2MF [53]. F2MF outperforms our approach on the test data; however, note that the F2MF model used for test evaluation was trained on both the training and validation dataset, while we only train our model on the training dataset. Moreover, F2MF does not predict instance IDs for 'thing' classes, meaning that this model cannot be used for panoptic segmentation.

Tab. 7 provides the instance segmentation forecasting metrics computed on the Cityscapes test dataset for our approach as well as F2F [43]. Our approach outperforms F2F on the mid-term setting. Luc *et al*. do not provide an evaluation on the short-term setting, so we cannot directly compare against those results here.

Tab. 8 presents a few additional ablations analyzing the impact of our modeling choices: a) *fg w/ independent RNNs* 'disconnects' the bounding box and appearance mask RNNs in the encoder and decoder so they do not use the outputs of f_{mfeat} and f_{bfeat} as input, and b) *fg w/o velocity features* only uses location features as input, i.e., $\mathbf{x}_t^i := [cx, cy, w, h, d]$. a) shows that joint modeling of instance motion and appearance mask leads to better performance at longer timescales. b) shows that guiding the network with temporal information is also important to improve longer-term forecasting.

Tables 9-14 show the per-class breakdown of all panoptic segmentation metrics presented in Tab. 1. The results shown in Tab. 1 consist of the average metric values computed over the set of classes in Cityscapes. For most classes, we outperform all other approaches on panoptic quality and recognition quality for both short- and mid-term. We additionally outperform all other approaches for many classes on segmentation quality, and outperform all other approaches on average.

H. Visualization

Fig. 7 presents visualizations for the short term setting for the sequences present in Fig. 4. Additional sequences are visualized for the mid-term setting in Fig. 8, and the corresponding short term sequences are presented in Fig. 9.

Included with the supplementary material are some videos visualizing predictions. Each displays the most recently seen input frame in the top half and the prediction 9 frames in the future from the last seen frame (i.e., for the mid-term setting). The folder '`short_videos`' contains visualizations for some of the sequences present within the Cityscapes validation dataset. The folder '`long_videos`' contains visualizations for a longer sequence taken from the unfiltered Frankfurt sequence provided with Cityscapes.

[8] https://github.com/adamtwig/segpred
[9] https://github.com/facebookresearch/instpred

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	97.9	78.2	88.5	29.4	38.9	60.0	55.6	74.5	89.5	36.1	87.9	50.8	46.4	67.3	51.5	66.6	37.8	44.2	44.1	60.3
Deeplab (Last seen frame)	94.3	52.4	71.1	11.3	19.4	6.1	12.9	15.0	72.1	16.9	72.7	10.3	8.0	29.6	35.1	51.7	24.2	9.8	7.9	32.7
Flow	95.6	61.5	79.8	17.3	**28.6**	8.7	26.2	36.8	80.7	**26.9**	79.7	21.0	14.0	43.4	40.6	56.8	26.7	23.2	18.7	41.4
Hybrid [59] (bg) and [43] (fg)	**96.2**	63.4	81.4	23.1	23.7	7.1	19.1	36.9	82.3	20.3	79.8	26.8	21.8	46.4	**42.2**	60.0	41.4	25.6	22.5	43.2
Ours	**96.2**	**66.1**	**83.5**	**26.1**	27.4	**31.7**	**37.0**	**49.9**	**84.8**	26.1	**82.0**	**31.8**	**31.5**	**48.8**	**42.2**	**61.2**	**47.0**	**31.4**	**27.3**	**49.0**

Table 9. **Per-class results for Panoptic Quality on Cityscapes validation dataset (short-term).**

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	97.9	78.2	88.5	29.4	38.9	60.0	55.6	74.5	89.5	36.1	87.9	50.8	46.4	67.3	51.5	66.6	37.8	44.2	44.1	60.3
Deeplab (Last seen frame)	90.4	32.5	57.6	7.6	10.6	4.6	8.9	7.4	55.1	8.8	57.3	5.3	2.5	13.2	19.2	27.3	10.1	4.7	3.0	22.4
Flow	90.5	35.8	66.2	7.7	15.0	4.6	11.9	11.1	65.6	11.6	64.4	5.9	2.5	19.0	21.5	27.7	13.5	11.8	5.3	25.9
Hybrid [59] (bg) and [43] (fg)	93.2	44.9	70.5	12.4	14.8	1.2	8.0	10.8	69.7	13.9	67.2	8.0	4.5	27.3	33.5	41.7	27.9	8.3	6.1	29.7
Ours	**93.9**	**50.8**	**76.4**	**18.2**	**19.9**	**8.7**	**18.7**	**28.5**	**77.0**	**18.6**	**72.7**	**16.2**	**12.0**	**33.3**	**36.1**	**53.0**	**29.8**	**14.1**	**12.6**	**36.3**

Table 10. **Per-class results for Panoptic Quality on Cityscapes validation dataset (mid-term).**

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	98.0	85.6	90.5	74.3	74.8	69.7	73.5	80.1	90.9	75.7	92.6	76.0	70.8	84.2	88.4	90.8	87.6	73.8	72.1	81.5
Deeplab (Last seen frame)	94.4	71.5	78.8	65.4	65.6	**67.0**	**68.3**	67.4	77.8	67.7	83.0	64.4	60.1	69.2	74.7	76.7	75.7	62.7	63.4	71.3
Flow	95.6	76.0	83.2	68.5	68.3	65.0	65.9	67.3	83.4	69.1	86.6	65.6	61.4	75.8	77.5	80.0	74.1	66.1	64.4	73.4
Hybrid [59] (bg) and [43] (fg)	**96.3**	**77.2**	84.9	70.0	69.0	59.5	63.6	65.9	84.6	70.8	86.5	66.8	61.9	77.2	**80.3**	**83.1**	**80.5**	65.6	63.8	74.1
Ours	**96.3**	77.0	**86.3**	**71.1**	**69.4**	61.4	65.4	**70.6**	**86.6**	**71.3**	**88.3**	**67.7**	**63.8**	**77.7**	81.4	81.4	74.8	**67.6**	**65.8**	**74.9**

Table 11. **Per-class results for Segmentation Quality on Cityscapes validation dataset (short-term).**

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	98.0	85.6	90.5	74.3	74.8	69.7	73.5	80.1	90.9	75.7	92.6	76.0	70.8	84.2	88.4	90.8	87.6	73.8	72.1	81.5
Deeplab (Last seen frame)	90.7	68.2	72.6	63.4	62.4	**66.1**	**72.7**	**73.0**	71.2	64.0	77.3	63.7	61.3	66.8	62.9	70.8	74.3	56.4	**64.4**	68.5
Flow	90.8	68.6	76.0	66.1	64.1	64.1	69.0	67.2	75.0	64.5	78.5	63.5	60.4	69.1	70.2	74.3	**75.8**	60.2	63.0	69.5
Hybrid [59] (bg) and [43] (fg)	93.3	69.7	77.9	66.6	65.3	59.9	62.9	61.9	76.9	65.1	79.6	63.7	58.4	71.5	72.6	72.2	73.7	62.1	60.6	69.1
Ours	**94.1**	**71.3**	**81.5**	**68.4**	**66.8**	59.0	64.1	65.1	**80.9**	**68.1**	**83.0**	**64.3**	**61.4**	**73.4**	**76.9**	**76.1**	74.4	**62.5**	62.9	**71.3**

Table 12. **Per-class results for Segmentation Quality on Cityscapes validation dataset (mid-term).**

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	99.9	91.3	97.8	39.5	52.1	86.1	75.6	93.0	98.5	47.7	94.9	66.9	65.5	80.0	58.2	73.4	43.1	59.9	61.2	72.9
Deeplab (Last seen frame)	**99.9**	73.4	90.2	17.3	29.7	9.1	18.9	22.2	92.7	25.0	87.6	16.0	13.2	42.8	47.0	67.4	32.0	15.6	12.4	42.7
Flow	**99.9**	81.0	95.9	25.2	**41.9**	13.4	39.7	54.7	96.8	**39.0**	92.0	32.1	22.8	57.3	52.4	71.0	36.0	35.1	29.0	53.4
Hybrid [59] (bg) and [43] (fg)	**99.9**	82.1	95.8	33.0	34.4	11.9	30.0	56.0	97.3	28.8	92.2	40.1	35.3	60.2	**52.6**	72.2	51.4	39.1	35.3	55.1
Ours	**99.9**	**85.8**	**96.7**	**36.7**	39.4	**51.6**	**56.6**	**70.7**	**97.9**	36.6	**92.9**	**47.0**	**49.3**	**62.9**	51.9	**75.2**	**62.9**	**46.3**	**41.5**	**63.3**

Table 13. **Per-class results for Recognition Quality on Cityscapes validation dataset (short-term).**

	road	sidewalk	building	wall	fence	pole	traffic light	traffic sign	vegetation	terrain	sky	person	rider	car	truck	bus	train	motorcycle	bicycle	mean
Deeplab (Oracle)†	99.9	91.3	97.8	39.5	52.1	86.1	75.6	93.0	98.5	47.7	94.9	66.9	65.5	80.0	58.2	73.4	43.1	59.9	61.2	72.9
Deeplab (Last seen frame)	99.7	47.6	79.3	12.1	17.0	7.0	12.2	10.1	77.4	13.7	74.1	8.3	4.2	19.8	30.6	38.5	13.6	8.3	4.7	30.4
Flow	99.7	52.2	87.1	11.7	23.4	7.2	17.3	16.5	87.4	18.0	82.1	9.2	4.2	27.5	30.6	37.3	17.8	19.6	8.4	34.6
Hybrid [59] (bg) and [43] (fg)	**99.9**	64.5	90.4	18.7	22.7	2.1	12.7	17.4	90.5	21.4	84.4	12.5	7.7	38.2	46.2	57.9	37.8	13.4	10.1	39.4
Ours	99.7	**71.2**	**93.7**	**26.6**	**29.8**	**14.7**	**29.2**	**43.8**	**95.1**	**27.3**	**87.6**	**25.2**	**19.5**	**45.4**	**47.0**	**69.6**	**40.0**	**22.6**	**20.0**	**47.8**

Table 14. **Per-class results for Recognition Quality on Cityscapes validation dataset (mid-term)**.

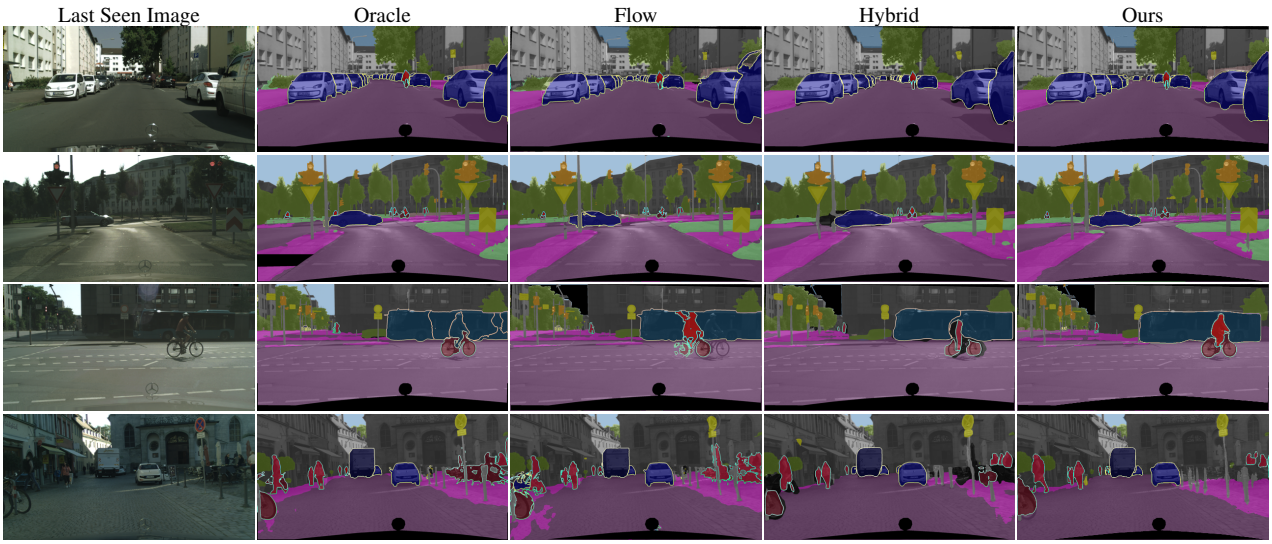

Figure 7. **Short-term panoptic segmentation forecasts on Cityscapes.**

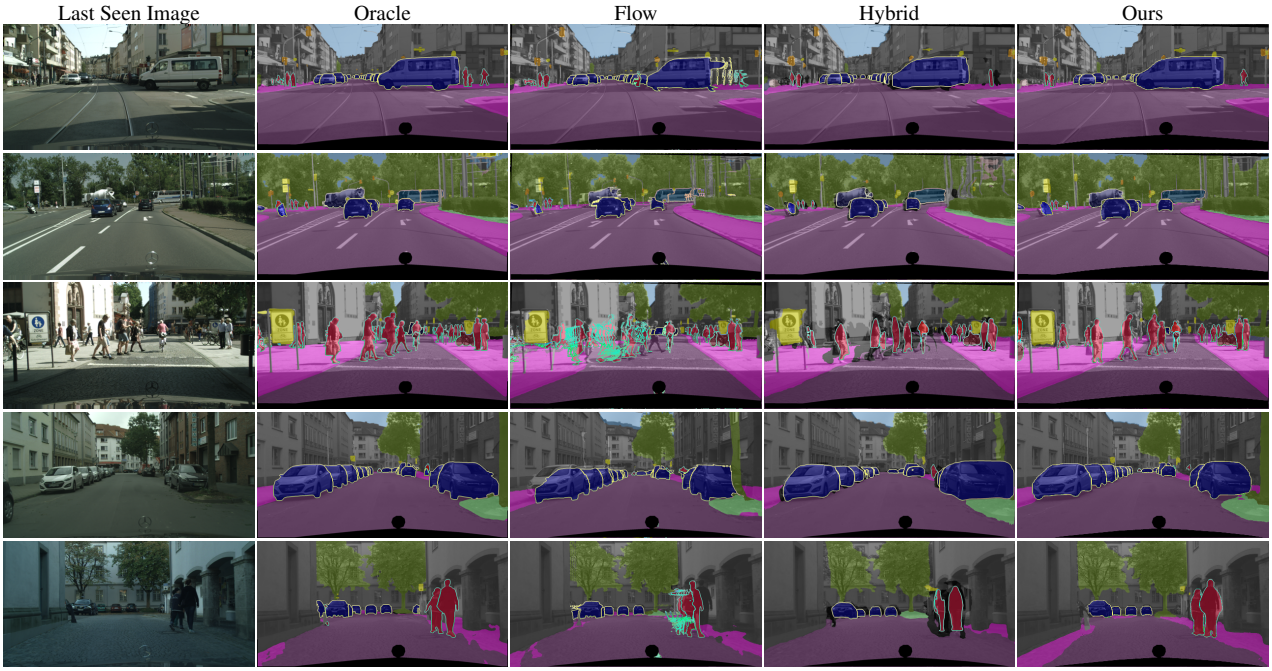

Figure 8. **Additional mid-term panoptic segmentation forecasts on Cityscapes.**

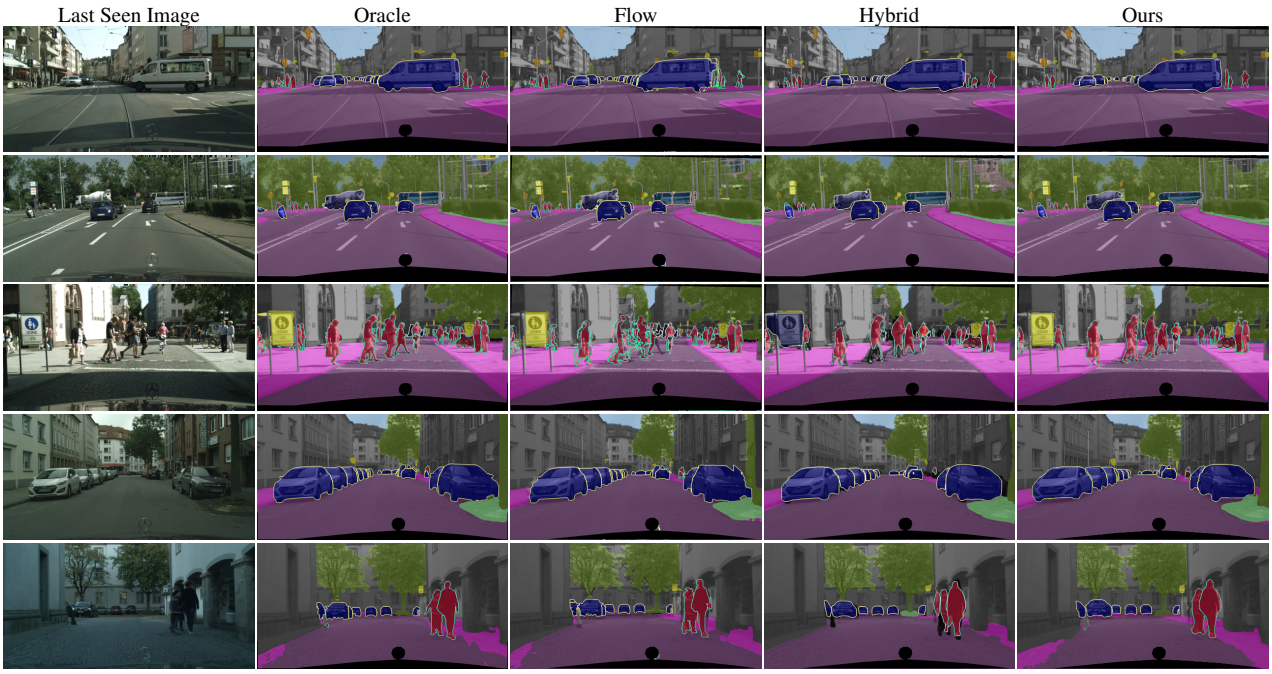

Figure 9. **Additional short-term panoptic segmentation forecasts on Cityscapes.**