Supplementary Materials
Implicit Occupancy Flow Fields for Perception and Prediction in Self-Driving

Ben Agro*, **Quinlan Sykora***, **Sergio Casas***, **Raquel Urtasun**
Waabi, University of Toronto
{bagro, qsykora, sergio, urtasun}@waabi.ai

In this appendix, we first describe implementation details that are relevant to IMPLICITO, including the full computation graph of our implicit decoder architecture, the encoder architecture, and training details. Then, we provide implementation details for the baselines. Last but not least, we showcase several additional results:

- Introspection of IMPLICITO ($K = 4$) using visualizations of the attention offsets, cross attention weights, and qualitative results.
- Analysis of occupancy metrics over time for both object-based and object-free prior art as well as IMPLICITO.
- Analysis of how performance is affected by grid resolution for explicit object-free methods.
- Insights into the occupancy probability calibration of different methods via reliability diagrams.

1. IMPLICITO Implementation Details

1.1. Encoder

See Fig. 1 for a diagram of the encoder. The encoder network takes as input the voxelized LiDAR $L \in \mathbb{R}^{T_{history}D \times H \times W}$ and map raster $M \in \mathbb{R}^{1 \times H \times W}$ to produce the feature map $\mathbf{Z}$ used by the implicit decoder of IMPLICITO. A detailed diagram of its architecture can be found in Fig. 2.

First, the voxelized LiDAR L and map raster M are processed by two independent convolutional stems. The LiDAR stem consists of two sequential $\{$`Conv2d`, `BatchNorm2d`, `ReLU`$\}$ blocks, with an input feature size of $T_{history}D = (5)(32) = 160$, a hidden size of 32, a kernel size of 3, and a stride of 1. The map stem is similar but uses three sequential blocks with an input feature size of 1, and a hidden size of 64.

The resulting LiDAR and map features are concatenated along the feature dimension, and passed through a ResNet [5] with five blocks, where the output of each block is saved. Each of the five blocks consist of $\{2, 3, 6, 6, 6\}$ "`BottleNeck`"s (see Fig. 1). Each `BottleNeck` is a sequence of three alternating `Conv2d` and `BatchNorm2d` layers, where the first and last convolutional layers use a kernel size of 1 and the second layer uses a kernel size of 3. All stride lengths are 1, except for the second layer of the first `BottleNeck` in each block, which may use a stride length of 2 to downsample the spatial dimensions of the input. For each `BottleNeck`, the output of the convolutions is a residual, i.e., it gets added to the input with a residual connection (downsampling with a `Conv2d` layer if necessary), and passed through a `ReLU` activation function. For each of the five blocks, we use hidden sizes of $\{32, 48, 64, 96, 128\}$ for the `BottleNeck`s, and each block has an output feature dimensionality that is four times its hidden size. The output of this five-block ResNet is a multi-resolution feature map with feature dimensionalities $\{128, 192, 256, 384, 512\}$, with spatial dimensions $\{(H/2, W/2), \ldots, (H/2^5, W/2^5)\}$, inspired by PIXOR [12].

These five feature maps are then processed by a lightweight FPN [7] with 32 output channels, resulting in a feature map with spatial dimensions $(H/2, W/2)$ and feature dimensionality 32. A final pyramid pooling layer increases the feature dimensionality to $C = 64$ [4]. In Fig. 2 of the main paper, we referred to the resulting feature map as $\mathbf{Z}$.

*Denotes equal contribution

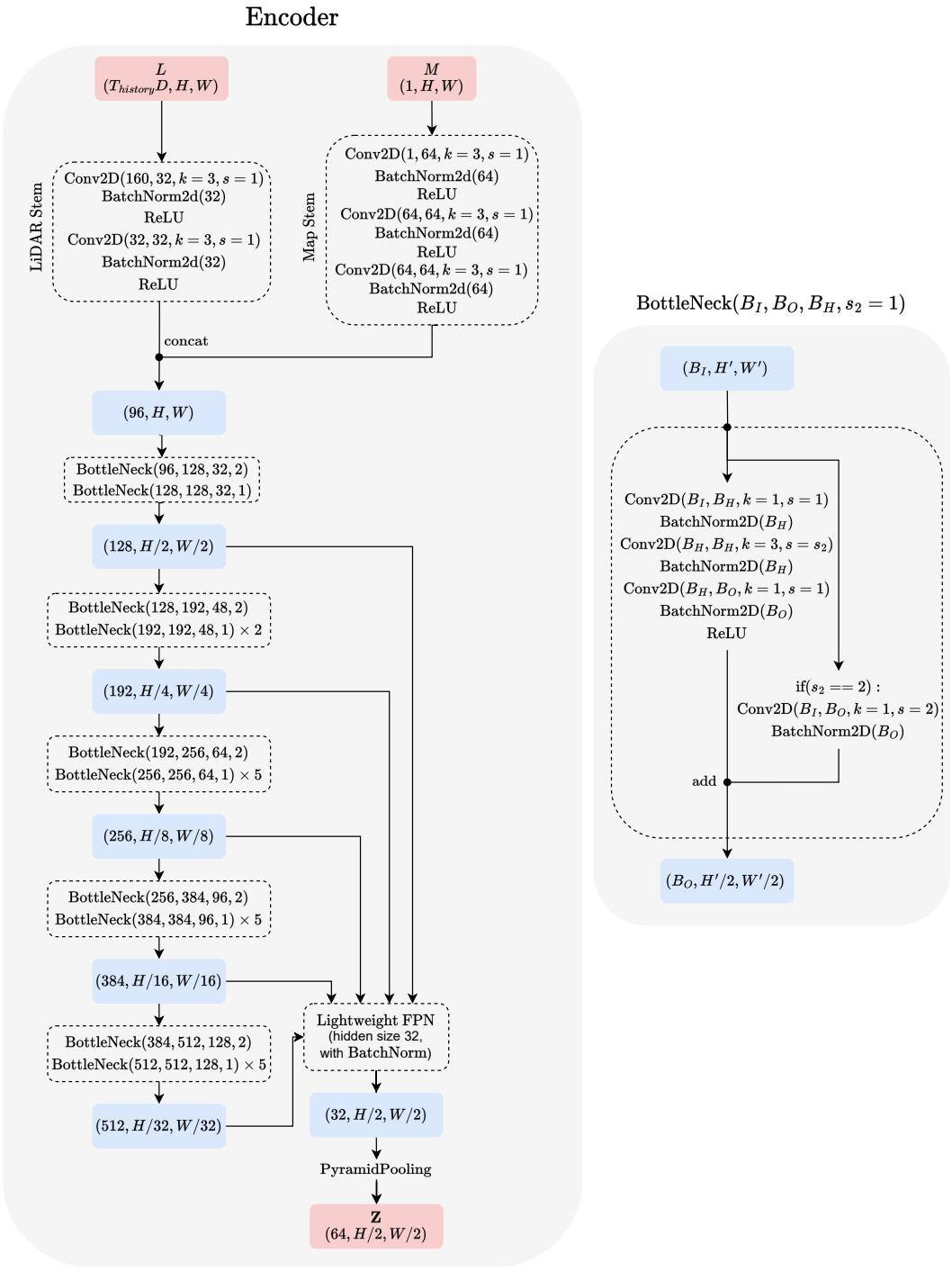

Figure 1. **Architecture of IMPLICITO's encoder.**

1.2. Implicit Decoder

The inputs to the decoder are the set of 3-dimensional query points $\mathcal{Q} \in \mathbb{R}^{|\mathcal{Q}| \times 3}$, and feature map from the encoder, $\mathbf{Z} \in \mathbb{R}^{C \times \frac{H}{2} \times \frac{W}{2}}$, where $|\mathcal{Q}|$ is the number of query points, and C is the feature dimensionality. H and W are the number of rows and columns of voxelized LiDAR and map raster input to the encoder, as described in the main paper. We use a different region of interest (ROI) and spatial discretization for AV2 and HwySim. For AV2 we use a square ROI of $80\,\mathrm{m} \times 80\,\mathrm{m}$ and a voxel spatial resolution of $0.1\,\mathrm{m}$, $H = W = 80\,\mathrm{m}/0.1\,\mathrm{m} = 800$. For HwySim we use an asymmetric ROI of $80\,\mathrm{m} \times 240\,\mathrm{m}$ and a voxel spatial resolution of $0.2\,\mathrm{m}$, $H = 80\,\mathrm{m}/0.2\,\mathrm{m} = 400, W = 240\,\mathrm{m}/0.2\,\mathrm{m} = 1200$. We use a larger voxel size on HwySim because it is too memory intensive to use anything smaller, given the larger ROI necessary for prediction at highways speeds.

As illustrated by the computational graph in Fig. 2, the decoder consists of three main parts: offset prediction, feature aggregation, and occupancy-flow prediction. We will describe each in turn.

Offset Prediction: The offset prediction module begins by interpolating $\mathbf{Z}$ at the (x, y) locations specified by the spatial components of the query points $\mathbf{q}_{x,y} \in \mathcal{Q}_{x,y}$ to produce interpolated feature vectors $\mathbf{z}_q$ that contain information about the scene around each query point. Next, linear projections of $\mathbf{z}_q$ and $\mathbf{q}$ are added and passed through a residual layer,

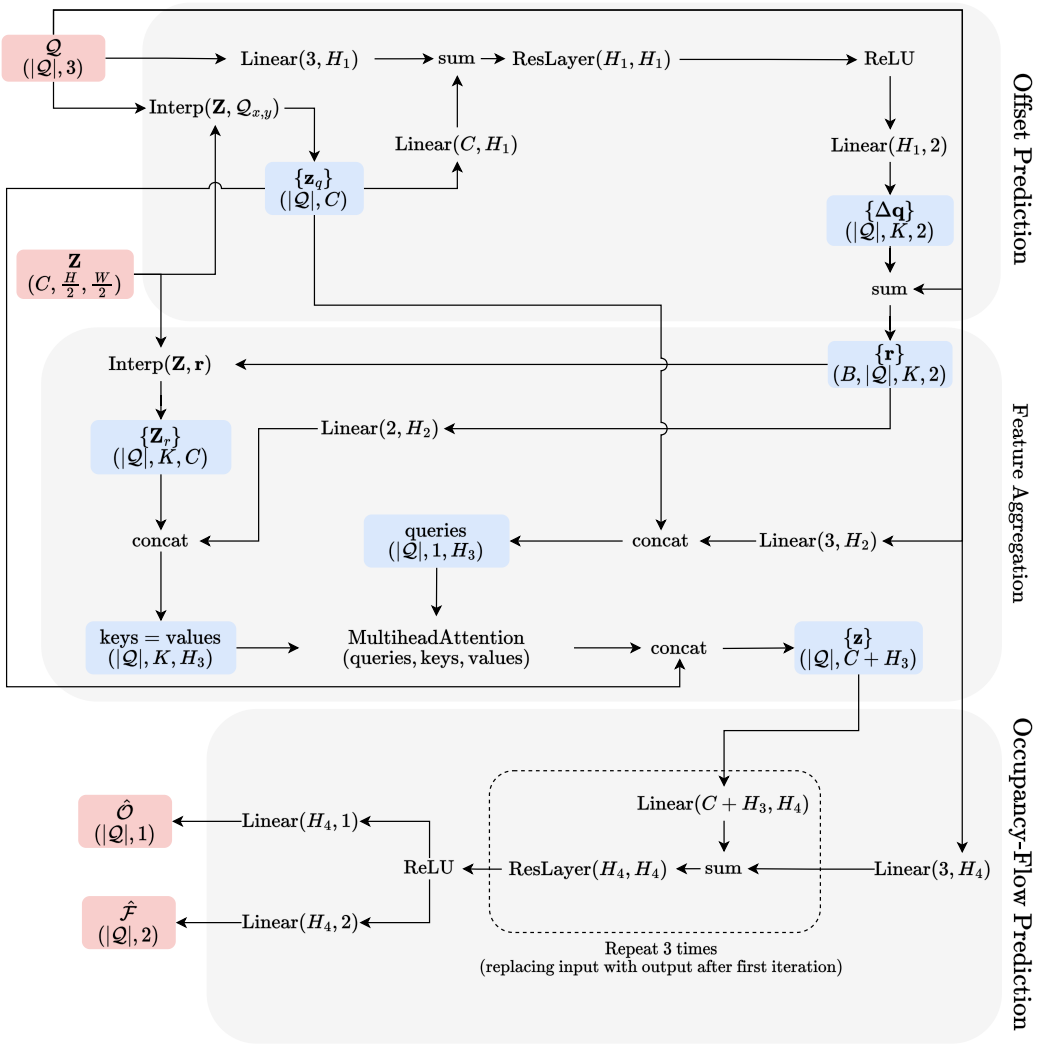

Figure 2. **Architecture of IMPLICITO's occupancy-flow decoder**. Note that we use the notation $\{\cdot\}$ to denote a set of variables so that the variable names match those of the main paper (where the decoder was explained for a single query point for simplicity).

`ResLayer`, consisting of two fully connected linear layers and a residual connection. We use $H_1 = 16$ for these linear projections and the hidden size of the linear layers in `ResLayer`, as shown in Fig. 1. Finally, a linear layer is used to produce a set of K offsets per query point $\Delta\mathbf{q}$. These offsets are added to the query points to find the offset sample locations $\mathbf{r} = \mathbf{q}_{x,y} + \Delta\mathbf{q}_{x,y}$, which is the input to the next module.

Feature Aggregation: The offset sample points $\mathbf{r}$ are meant to store the (x', y') locations in $\mathbf{Z}$ that contain important information for occupancy-flow prediction at query point (x, y, t). As such, the feature aggregation module begins by interpolating $\mathbf{Z}$ at $\mathbf{r}$ to obtain a set of K feature vectors, $\mathbf{Z}_r$, for each query point. To aggregate these feature vectors, we use cross-attention between the feature vector interpolated at the query point $\mathbf{z}_q$, and those interpolated at the offset locations $\mathbf{Z}_r$. Specifically, we concatenate $\mathbf{Z}_r$ and $\mathbf{z}_q$ with their positional embeddings — the interpolation locations passed through a linear layer (output feature dimension $H_2 = 8$) — to produce the queries (not to confuse these transformer query embeddings with our query points), keys and values, respectively. Then we use multi-head attention to aggregate the features from the sampled reference points, MultiHeadAttention(queries, keys, values), and finally concatenate it with $\mathbf{z}_q$ to obtain the summary features for the query point, denoted $\mathbf{z}$, of dimensionality $H_3 = C + H_2 = 72$. We treat the model variants where we use no offsets ($K = 0$) and a single offset ($K = 1$) as special cases:

- $K = 0$ case: In this case, there are no attention offsets, and occupancy-flow is predicted using $\mathbf{z} = \mathbf{z}_q$.
- $K = 1$ case: this eliminates the need for aggregating multiple interpolated feature vectors via cross-attention because $\mathbf{Z}_r \in \mathbb{R}^{|\mathcal{Q}| \times 1 \times C}$ contains just one feature vector per query point. In this case, that we directly concatenate $\mathbf{Z}_r$ with $\mathbf{z}_q$ along the feature dimension to produce $\mathbf{z}$. Note that the main results in the paper are made with IMPLICITO with $K = 1$

Occupancy-Flow Prediction: This module uses the aggregated feature vector $\mathbf{z}$ and the query points $\mathbf{q}$ to predict occupancy logits, $\hat{\mathcal{O}}$, and the backwards flow predictions, $\hat{\mathcal{F}}$, for all query points. $\mathbf{z}$ and $\mathbf{q}$ are passed through three residual blocks with an architecture inspired by Convolutional Occupancy Networks [10]. The residual block takes as input a linear projection of $\mathbf{q}$. Each block adds its input to a linear projection of $\mathbf{z}$, and then passes this through a residual layer `ResLayer` using a hidden size $H_4 = 16$. The input to the subsequent block is the output of the previous block.

1.3. Training Details

Initialization: We found that training was more consistent if we initialized the weights with a small standard deviation of 0.01 and a bias of 0 for the linear layer Linear$(H_1, 2)$ that predicts the offsets $\Delta\mathbf{q}$. This ensures that the initial offset predictions are small enough such that $\mathbf{r} = \mathbf{q} + \Delta\mathbf{q}$ still represents a point within the feature map $\mathbf{Z}$, but large enough such that the predicted offsets do not get stuck at $\Delta\mathbf{q} = \mathbf{0}$.

Loss hyperparameters: We use a flow regression loss weighting of $\lambda_{\mathbf{f}} = 0.1$ (downweighted by a factor of 10 relative to occupancy loss).

Optimizer: We use an initial learning rate of 1.0×10^{-3}, which is multiplied by a factor of 0.25 every 6 training epochs. We train for a total of 20 epochs, using the AdamW optimizer [8] with a weight decay of 1.0×10^{-4}.

Query points used during training: We use $|\mathcal{Q}| = 220,000$ query points per example in the batch, sampled uniformly on the spatio-temporal volume. Future work may investigate more effective sampling methods for training.

2. Baseline Implementation Details

Object detector: For a fair comparison with object-based and hybrid baselines (MULTIPATH, LANEGCN, GORELA and OCCFLOW), we use the same state-of-the-art object detector, Two-stage PIXOR, in all of their implementations. This detector is a variant of the original PIXOR [12], with the following improvements:

- We use multi-scale deformable self-attention [13] instead of the upsampling deconvolutional layers after the ResNet backbone to aggregate information from different scales and enhance the feature extraction. Following PIXOR, we output the feature maps with $1/4$ resolution.

- We use the output from the dense detector header as the first-stage results. The top 500 bounding boxes after non-maximum supression (NMS) are used as region proposal for the second stage. The 2D IoU threshold for NMS is set to 0.7.
- We use RotatedROIAlign [11] to extract 3×3 ROI features from the feature map in the second-stage header, and apply two self-attention layers: one on features within each ROI and another between features of different ROIs.
- Two MLPs are used to predict the classification score and box refinement for the region proposals.
- We use DETR-like set-based loss [1] with bipartite matching for both stages. An additional IoU loss is used for bounding box regression besides the smooth L1 loss.

Trajectory prediction baselines: For the methods that perform trajectory prediction (MULTIPATH, LANEGCN, GORELA), we train Two-stage PIXOR jointly with the trajectory prediction model in an end-to-end fashion, as proposed in SpAGNN [2].

OccFlow: This hybrid method performs object-based perception (object detection) and object-free prediction (occupancy-flow). Since it requires kinematic information in the raster passed from perception to prediction, we train the detector to additionally output longitudinal, lateral and angular velocity and acceleration. We then postprocess this information into a raster with five channels: occupancy probability based on detector confidence, velocity in x and y directions, and acceleration in x and y directions, for each pixel. Given this raster, we employ the same encoder architecture as our model (described in Sec. 1.1) for a fair comparison. Finally, a fully convolutional header decodes both occupancy and backward flow on a spatio-temporal grid.

MP3: This baseline also uses the same encoder as our model for a fair comparison (see Sec. 1.1). It then decodes initial occupancy and forward flow over time with a fully convolutional header, and the future occupancy is obtained by warping the initial occupancy with the temporal flow as described in the original paper.

3. Additional Results

3.1. Qualitative Results for IMPLICITO with $K = 4$ offsets

Fig. 3 presents the occupancy predictions over time, reverse flow predictions, attention offsets, and cross attention weights of IMPLICITO with $K = 4$ across four different scenes in AV2. The reverse flow predictions, attention offsets, and cross attention weights are all visualized at the final timestep in the prediction horizon. For clarification on the different flow visualizations used throughout main manuscript and supplementary, please see Fig. 5. The cross-attention weights are those used in the weighted sum of the K interpolated feature vectors in the feature aggregation module (see Fig. 2).

First, we notice that the offsets learn to look mainly backward and forward in the direction of the lanes. Second, we find that the decoder pays more attention to those offsets that look backwards along the lanes. This is particularly apparent in Offset 1; it is weighted heavily in the lane regions where it points against the flow of traffic. This aligns with our intuition that the offset sampling mechanism helps the model attend to the where occupancy was in the past — where the LiDAR evidence is — in order to predict the future.

We hypothesize that IMPLICITO with $K = 4$ does not outperform IMPLICITO with $K = 1$, because, assuming the offsets look backwards in time and space, $K = 1$ can already model multi-modal futures for occupancy, in the same way that the backwards-flow formulation does. Then, multiple offset predictions would only provide more information in areas where occupancy could have come from multiple directions in the past, like the intersection in the Scene 3 of Fig. 3. Indeed, focusing in on this example in Fig. 4, we see that IMPLICITO with $K = 4$ is qualitatively able to predict future occupancy from both the left turning car (**A**) and the cars entering the intersection (**B**), as highlighted in the blue box, while IMPLICITO with $K = 1$ only predicts occupancy from **B**. However, these examples are rare in the training and evaluation data, and $K > 1$ adds noise in cases where it is not needed, explaining the slightly worse metrics of $K = 4$ relative to $K = 1$.

3.2. Measuring Occupancy Predictions Over Time

This section presents an analysis of the occupancy prediction accuracy as a function of prediction time Δt. We want to investigate (1) how do the occupancy metrics of the object-free baselines from the literature and IMPLICITO evolve over time, and (2) how well do the explicit models (those that predict occupancy on a spatio-temporal grid) predict occupancy at timesteps that are not aligned with the grid.

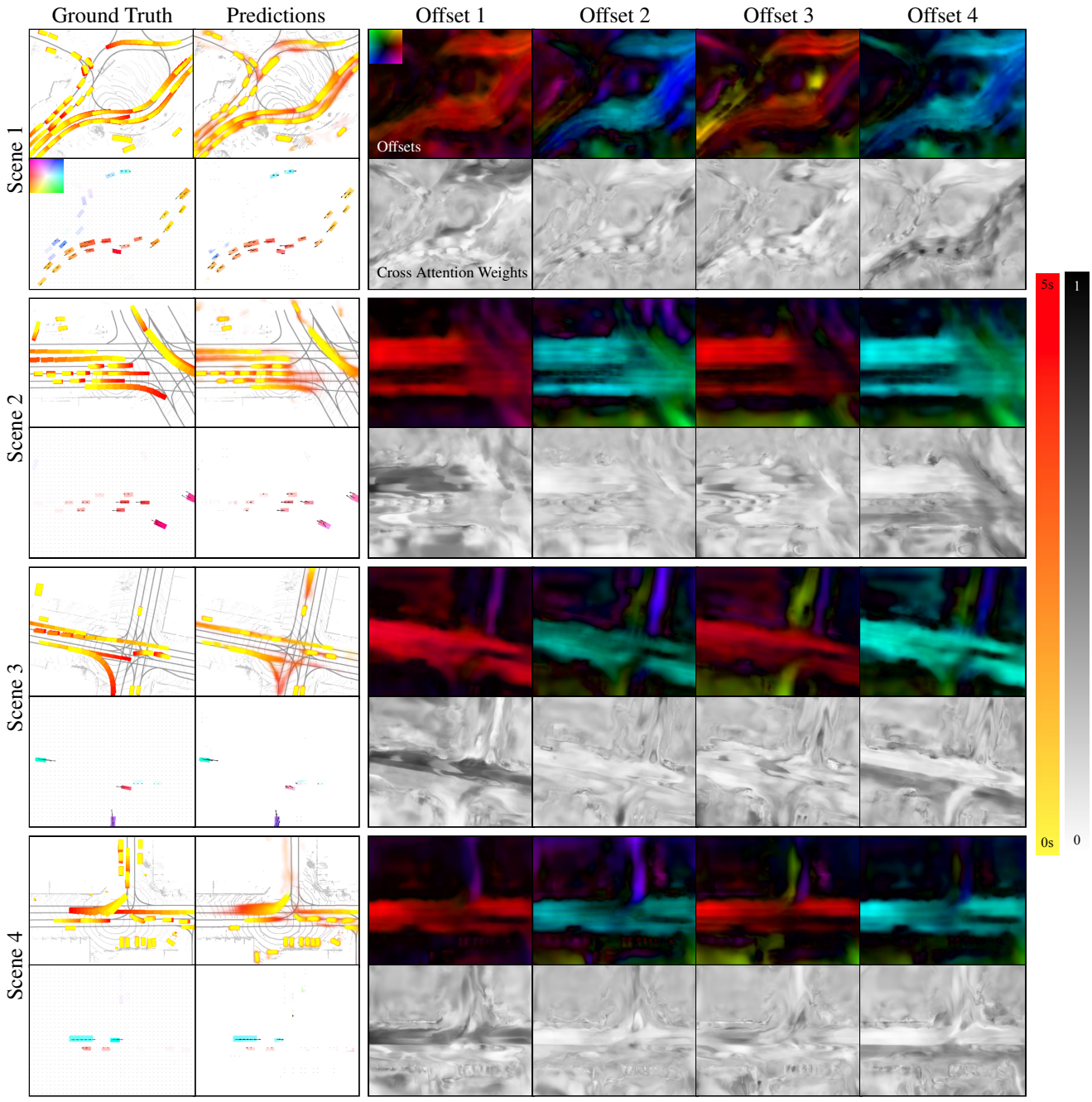

Figure 3. Qualitative results of IMPLICITO with $K = 4$ on AV2. The grey scale images display the cross attention weights, which sum to 1 across the columns. The HSV color images are the offset predictions. The predicted reverse flow field is masked by the ground truth occupancy for visual comparison (see Fig. 5). The reverse flow, attention offsets, and cross attention weights are all visualized at the final timestep.

Expanding on (2): Due to memory constraints, all explicit models (e.g., MP3 and OCCFLOW) output an occupancy map with temporal dimension $(5/0.5)+1 = 11$, where the $+1$ comes from the prediction at the initial observation time ($\Delta t = 0$). However, typically motion planners may need higher temporal resolution query points, because, for example, a highway vehicle can travel $15\,\mathrm{m}$ in $0.5\,\mathrm{s}$. IMPLICITO by-passes this problem because it is trained on continuous query points at no additional cost. Thus, in this section we evaluate all models at a time discretization of 0.1s (the period of LiDAR the sweeps). For the explicit models, this requires linear interpolation on the predicted occupancy map.

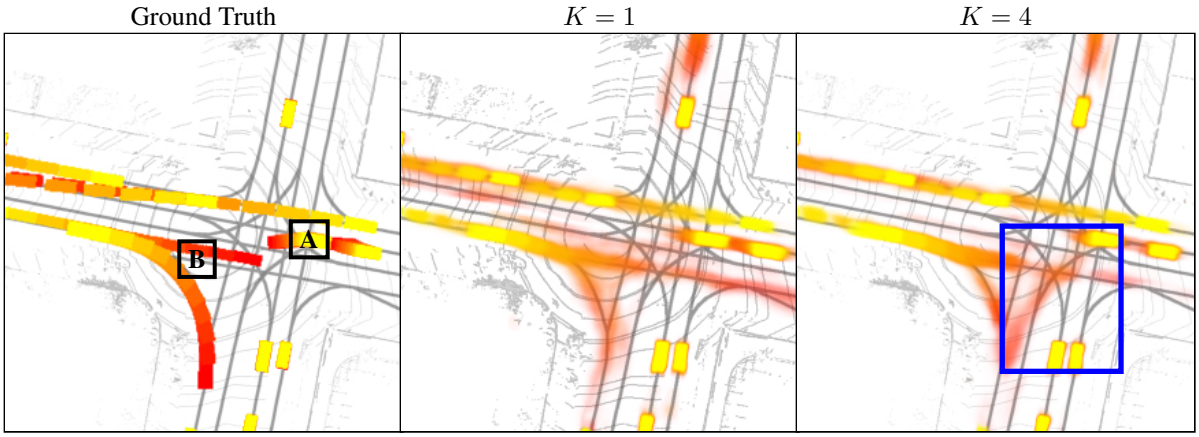

Figure 4. A side by side comparison of IMPLICITO with $K = 1$ and $K = 4$. The blue box highlights that $K = 4$ is qualitatively better able to predict occupancy in the intersection that could have come from two past locations, namely **A** and **B**.

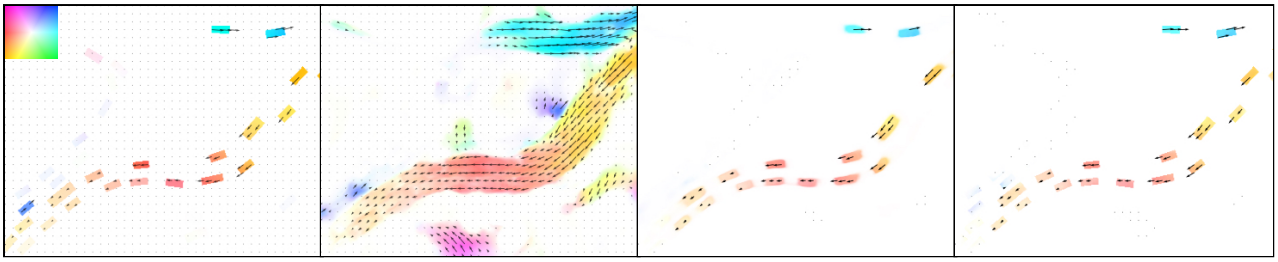

Figure 5. Visualizations of the reverse flow predictions of IMPLICITO with $K = 4$ at the initial timestep. (a) is the ground truth flow field, (b) is the predicted flow field, (c) is the predicted flow field masked by the occupancy predictions, and (d) is the predicted flow field masked by the ground truth occupancy.

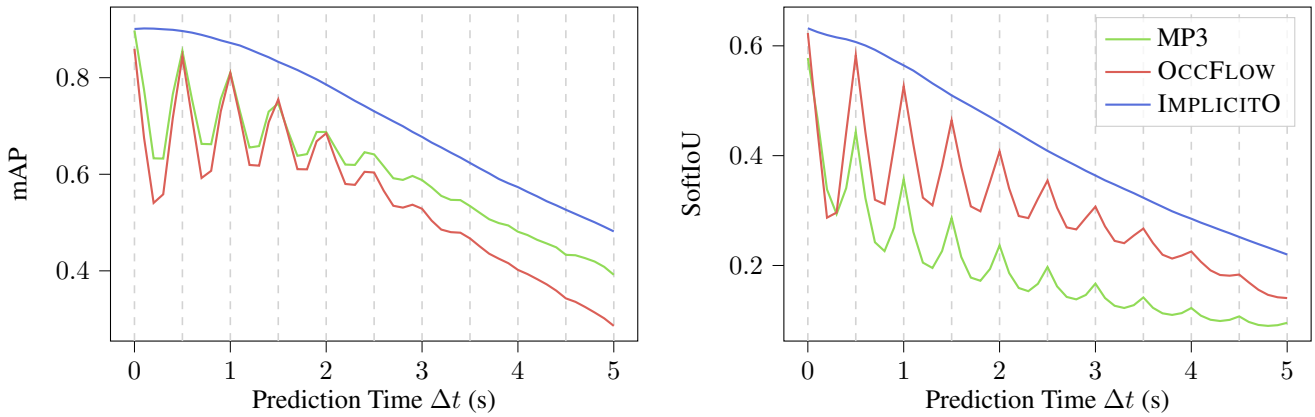

Figure 6. mAP and Soft IoU as functions of prediction time for IMPLICITO and various object-free models from the literature on HwySim.

The results for HwySim are presented in Fig. 6. The first trend to note is that while all models have a similar mAP and Soft-IoU at the initial timestep, IMPLICITO has a higher mAP and Soft-IoU at later timesteps. We attribute this to its expressive attention offset mechanism, which increases the effective receptive field, allowing accurate occupancy predictions far away from the LiDAR evidence information in $\mathbf{Z}$. Secondly, we observe that at prediction times that require interpolation (e.g., those not at $\{0, 0.5, \ldots, 5\}s$), the explicit models have worse mAP and Soft-IoU than IMPLICITO. This is because this interpolation inherently fails to model how occupancy moves; e.g., interpolation at a spatio-temporal point between grid timesteps can result in no predicted occupancy even if occupancy had passed through that point along its trajectory. IMPLICITO does not suffer from this problem because it can produce predictions at any continuous time. We observe this in

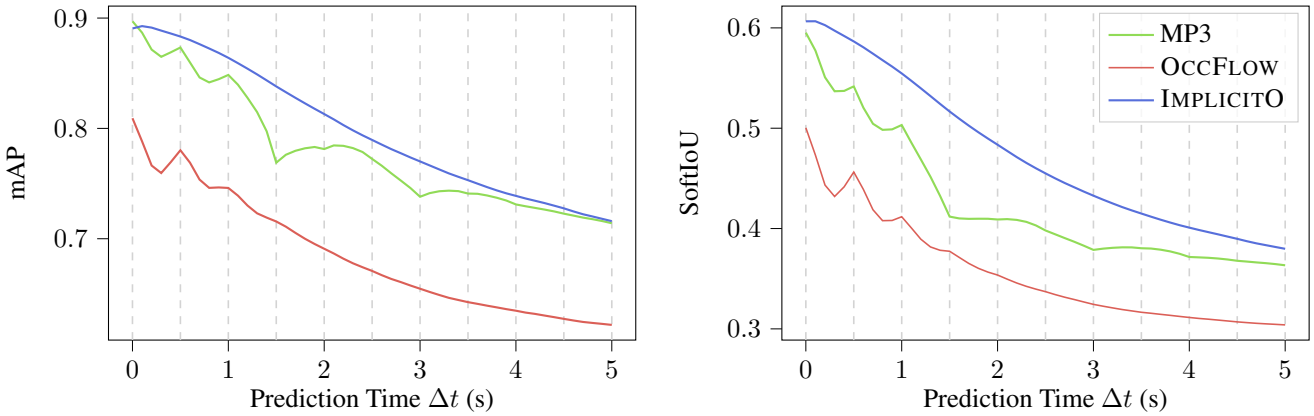

Figure 7. mAP and Soft IoU as functions of prediction time for various object-free models from the literature and ImplicitO on AV2.

the smooth mAP and Soft-IoU curves over time of ImplicitO. We notice that while MP3 has a more accurate occupancy ranking (mAP) over time than OccFlow, it has a worse Soft-IoU. The better occupancy ranking is attributed to the flow-warping mechanism of MP3, which imposes a realistic prior on the evolution of occupancy, and increases the effective receptive field. However, as described by Mahjourian et al. [9], MP3 suffers from lost occupancy over time, as reflected in its poor Soft-IoU.

We also include occupancy metrics over time on AV2, presented in Fig. 7. While the conclusions are similar here, they are less obvious, likely due to the fact that the vehicles move a smaller proportion of the ROI than in HwySim on average, making interpolation more accurate. To illustrate the advantages of ImplicitO we compare it qualitatively against baselines on a single scene from AV2 in Fig. 8. GoRela predicts many agents entering the intersection at once, which is unrealistic. OccFlow has a detection with incorrect size and orientation of a vehicle at $\Delta t = 0.0\,\mathrm{s}$, and suffers from spreading / inaccurate occupancy predictions at later timesteps, likely due to the limited receptive field of its convolutional decoder. MP3 has disjoint-pixel occupancy predictions, which is caused by its forward-flow warping mechanism and was first observed by Mahjourian et al. [9]. In contrast, ImplicitO exhibits accurate occupancy predictions and realistic multi-modality (vehicle going straight or turning left, and a lane-change).

3.3. Spatial Resolution Evaluation

In this experiment, we investigate how the performance of an explicit object-free model is affected by its output grid resolution. We train three MP3 models, each with a different output grid resolution; $0.2\,\mathrm{m}$, $1.0\,\mathrm{m}$, and $2.0\,\mathrm{m}$. The rasterized input to the encoder is kept the same, and the models are evaluated with a spatial resolution of $0.2\,\mathrm{m}$ by bi-linearly interpolating the output grid. The results are presented in Tab. 1, along with the metrics of ImplicitO for comparison. We notice that at an output resolution of $2.0\,\mathrm{m}$, the performance of MP3 is much worse than with a $0.2\,\mathrm{m}$ output resolution. However, $1.0\,\mathrm{m}$ is only slightly worse. This is likely because the ground-truth labels are rectangular bounding boxes, so as long as the discretization resolution is less than the smallest bounding box dimension, bi-linear interpolation on the output grid will be accurate. Most vehicles have a smallest side length greater than $1.0\,\mathrm{m}$, but less than $2.0\,\mathrm{m}$, explaining the results.

	mAP ↑	Soft-IoU ↑	ECE ↓	EPE ↓	Flow Grounded	
					mAP ↑	Soft-IoU ↑
MP3 (2.0 m)	0.635	0.286	0.731	0.547	0.986	0.280
MP3 (1.0 m)	0.769	0.416	0.307	0.458	0.927	0.460
MP3 (0.2 m)	0.774	0.422	0.201	0.472	0.902	0.466
ImplicitO	**0.799**	**0.480**	**0.193**	**0.267**	**0.936**	**0.597**

Table 1. Comparing MP3 [3] with different spatial output resolutions, evaluated at a spatial resolution of $0.2\,\mathrm{m}$ on AV2. The results for ImplicitO are presented for reference.

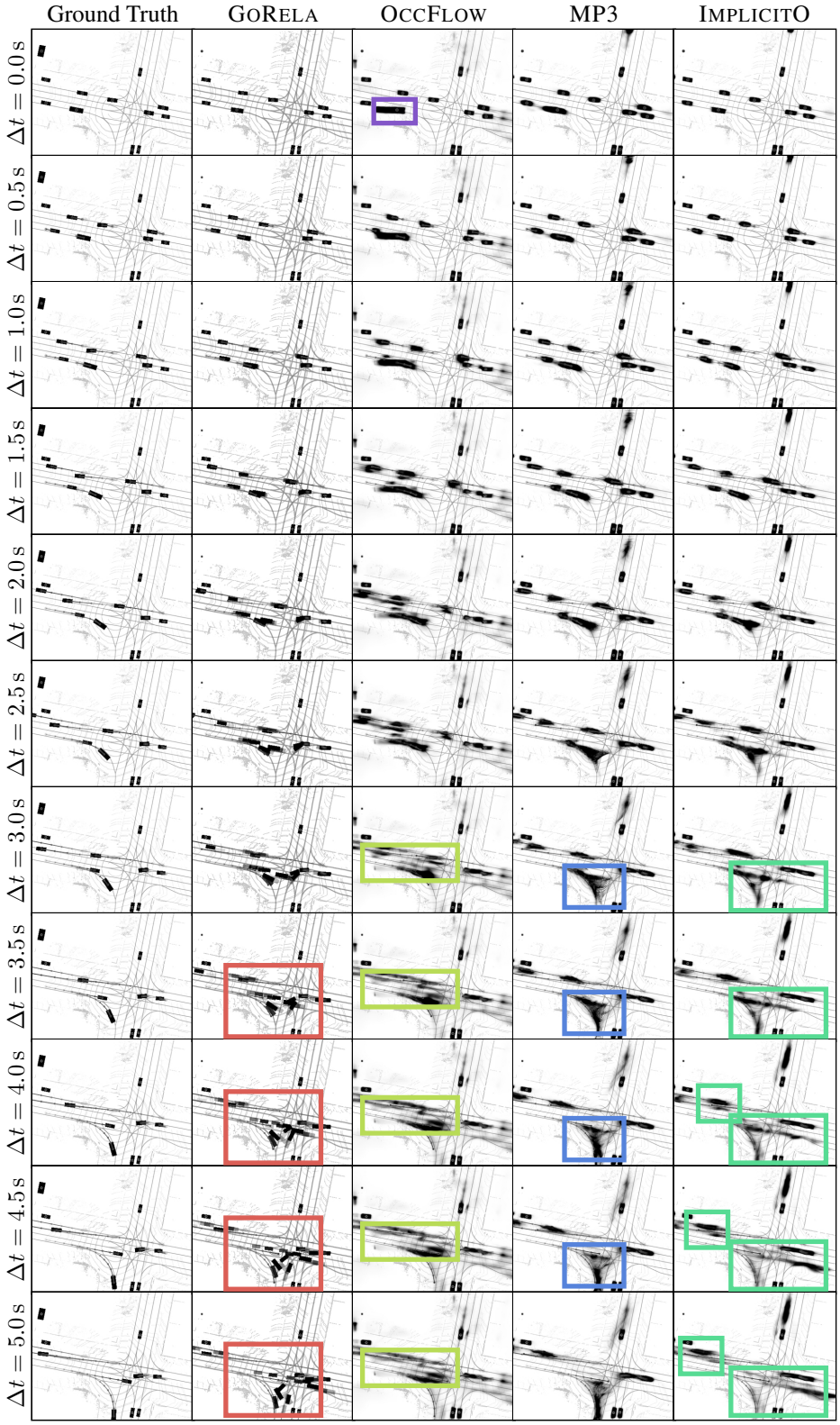

Figure 8. Occupancy predictions over time on a single scene from AV2 for various baseline models and ImplicitO. The alpha channel denotes occupancy probability, colored in black. Observations are denoted with colored boxes: **inconsistent with actors**, **spreading occupancy predictions**, **incorrect detection size**, **disjoint-pixel occupancy predictions**, **realistic multi-modal predictions**,

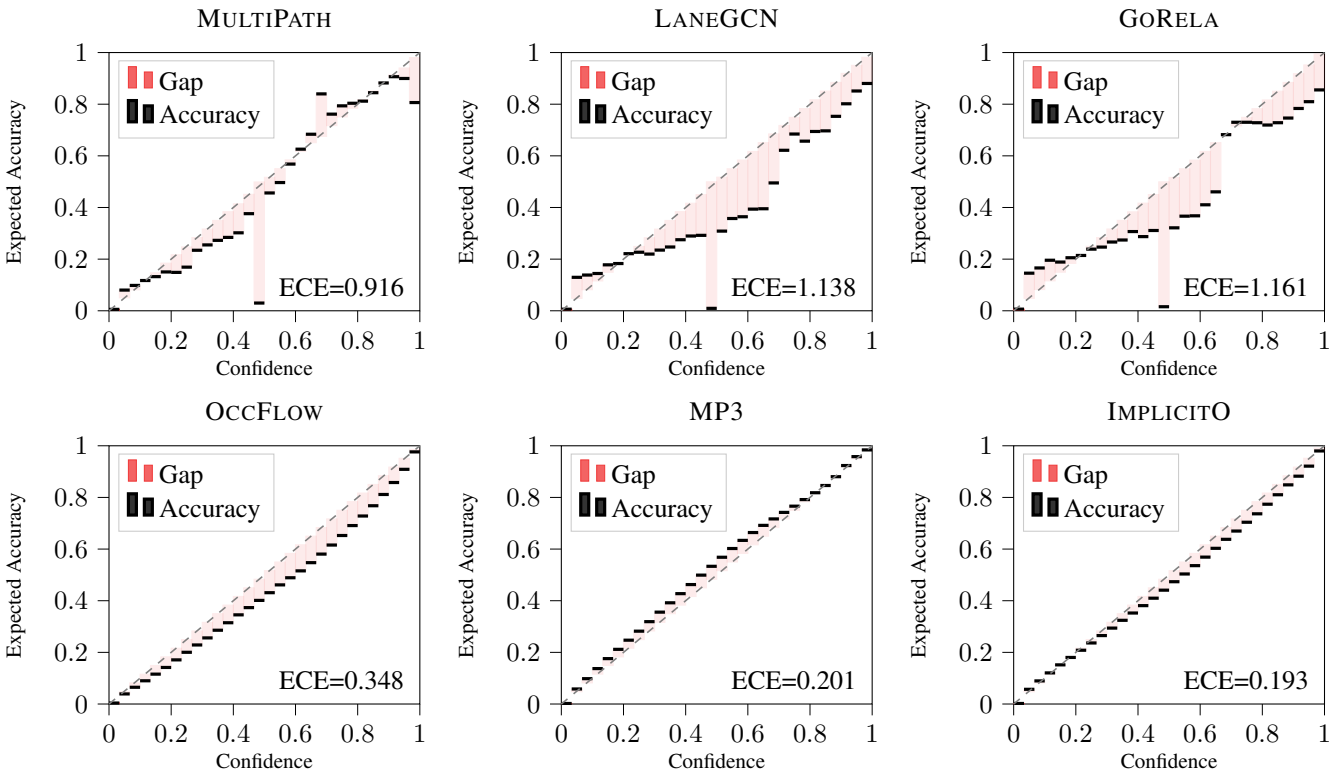

Figure 9. Reliability diagrams on AV2.

3.4. Analyzing Occupancy Calibration with Reliability Diagrams

Fig. 9 displays the reliability plots for all baselines from the literature on AV2. All the methods in the top row are object-based, while those in the bottom row are object-free. We notice that the object-based methods are poorly calibrated on the task of occupancy prediction relative to the object-free methods. Object-based methods predict discrete multi-modal trajectories which poorly capture the uncertainty in occupancy, and result in overconfidence. On the contrary, object-free methods can capture non-parametric occupancy distributions without thresholding to produce instances, resulting in better calibration.

3.5. Query Point Training Procedure

As described in Sec. 3.3, we train IMPLICITO with a batch of continuous query points $\mathcal{Q}$ sampled uniformly from the future spatio-temporal volume. This section ablates this querying procedure by training IMPLICITO on AV2 by querying points on a regular spatio-temporal grid with 0.5 s temporal resolution and varying spatial resolutions. Our evaluation procedure is the same as described in Sec. 4, and our results are presented in Tab. 2. We observe a slight disadvantage of training with points from a regular grid with respect to our proposed method.

	Spatial Resolution			
	0.2 m	0.5 m	1.25 m	2.0 m
mAP ↑	−0.19%	−0.25%	−0.83%	−0.90%
Soft-IoU ↑	−0.46%	−1.86%	−1.66%	−2.24%

Table 2. Comparing the performance of IMPLICITO when trained with query points sampled from a regular spatio-temporal grid against the proposed continuous query point training procedure. We report the relative change in evaluation metrics.

3.6. LiDAR History Ablation

In this section we measure the effect of LiDAR input frequency and history duration on the performance of IMPLICITO. First, we highlight that the LiDAR frequency and history duration as well as encoder architecture were held constant for all

models so changes in performance due to different LiDAR inputs are expected to be similar across all methods. When we train IMPLICITO with 10 past LiDAR frames at 10 Hz and see a minor relative improvement of 0.24% in mAP and 0.98% in Soft-IoU. For reference, MP3 improves similarly (0.45% mAP and 0.22% Soft-IoU).

3.7. Inferring Agent Identity

In this section, we provide a proof of concept for inferring agent identity (ID) with appropriate post-processing, similar to [6,9]. Inferring agent ID requires some form of instance prediction (like the "centerness" field from [6], or object detections in OCCFLOW [9]).

Without any additional training required, we pair IMPLICITO $K = 1$ with an object detector (the same one used by the object-based baselines). Then, for any query point $\mathbf{q}$, we add the attention offset $\mathbf{\Delta q}$ to find the reference point $\mathbf{r} = \mathbf{q} + \Delta\mathbf{q}$. Finally, we assign the ID of occupancy at $\mathbf{q}$ to that of detection with the closest centroid to $\mathbf{r}$. Recall that despite the attention offset $\mathbf{\Delta q}$ being unsupervised, we found $\mathbf{r}$ to point to the occupancy at the current time $\Delta t = 0$ (i.e., where the LiDAR evidence for that object is; see L721-737, Fig. 4 in the main paper).

Figure 10 visualizes the predicted occupancy colored by inferred instance ID (i.e., its associated detection). This simple approach illustrates that agent ID can be preserved in our implicit occupancy-flow method, and opens the door to future work on this area to improve this heuristic.

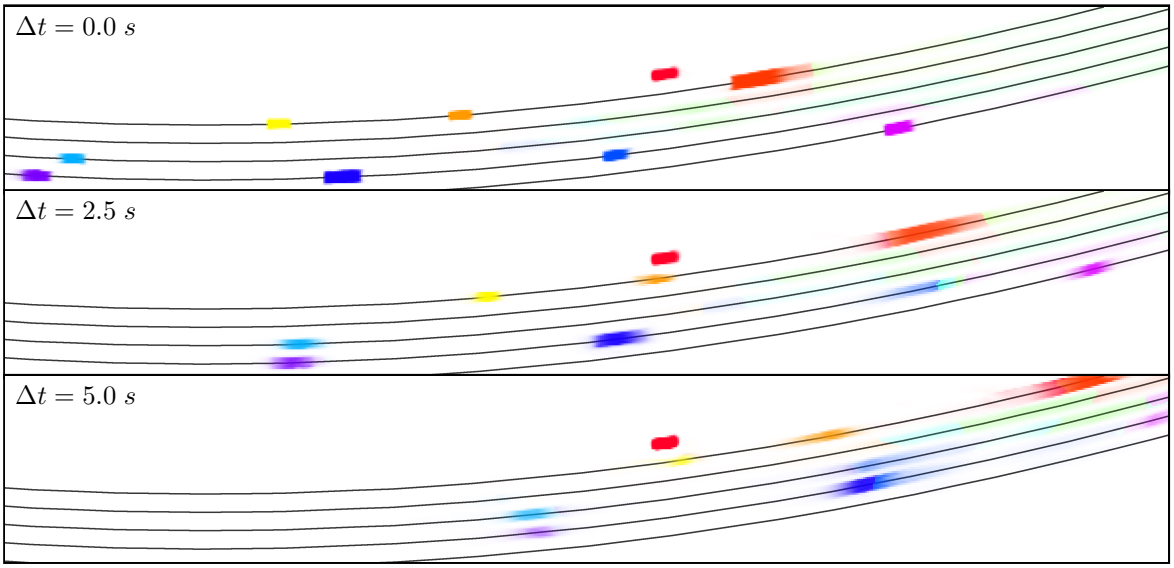

Figure 10. Qualitative examples of IMPLICITO inferring agent ID.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 5

[2] Sergio Casas, Cole Gulino, Renjie Liao, and Raquel Urtasun. Spatially-aware graph neural networks for relational behavior forecasting from sensor data. *arXiv preprint*, 2019. 5

[3] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *CVPR*, 2021. 8

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. pages 346–361, 2014. 1

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 1

[6] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In *ICCV*, 2021. 11

[7] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 4

[9] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 2022. 8, 11

[10] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 4

[11] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 5

[12] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 1, 4

[13] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv*, 2020. 4