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

Abstract

A self-driving vehicle (SDV) must be able to perceive its surroundings and predict the future behavior of other traffic participants. Existing works either perform object detection followed by trajectory forecasting of the detected objects, or predict dense occupancy and flow grids for the whole scene. The former poses a safety concern as the number of detections needs to be kept low for efficiency reasons, sacrificing object recall. The latter is computationally expensive due to the high-dimensionality of the output grid, and suffers from the limited receptive field inherent to fully convolutional networks. Furthermore, both approaches employ many computational resources predicting areas or objects that might never be queried by the motion planner. This motivates our unified approach to perception and future prediction that implicitly represents occupancy and flow over time with a single neural network. Our method avoids unnecessary computation, as it can be directly queried by the motion planner at continuous spatio-temporal locations. Moreover, we design an architecture that overcomes the limited receptive field of previous explicit occupancy prediction methods by adding an efficient yet effective global attention mechanism. Through extensive experiments in both urban and highway settings, we demonstrate that our implicit model outperforms the current state-of-the-art. For more information, visit the project website: https://waabi.ai/research/implicito.

1. Introduction

The goal of a self-driving vehicle is to take sensor observations of the environment and offline evidence such as high-definition (HD) maps and execute a safe and comfortable plan towards its destination. Meanwhile, it is important to produce interpretable representations that explain why the vehicle performed a certain maneuver, particularly if a dangerous event were to occur. To satisfy this, traditional autonomy stacks [2, 6, 9, 14, 15, 20, 32, 38, 39] break down the problem into 3 tasks: perception, motion forecasting and motion planning. Perception leverages sensor data to localize the traffic participants in the scene. Motion forecasting outputs the distribution of their future motion, which is typically multimodal. Finally, motion planning is tasked with deciding which maneuver the SDV should execute.

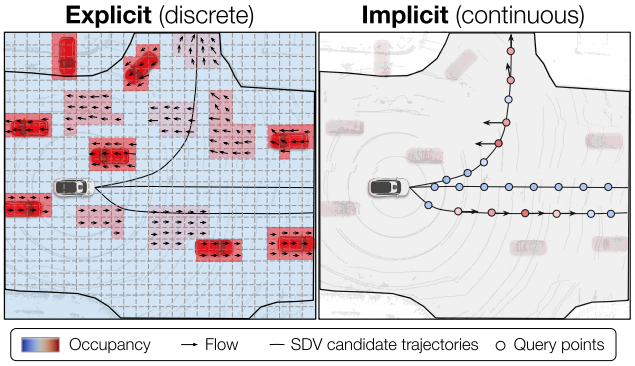

Figure 1. Left: Explicit approaches predict whole-scene occupancy and flow on a spatio-temporal grid. Right: Our implicit approach only predicts occupancy and flow at queried continuous points, focusing on what matters for downstream planning.

Most autonomy systems are *object-based*, which involves detecting the objects of interest in the scene. To do so, object detectors threshold predicted confidence scores to determine which objects in the scene, a trade off between precision and recall. Furthermore, object-based motion forecasting methods are limited to predict only a handful of sample trajectories or parametric distributions with closed-form likelihood for tractability, as they scale linearly with the number of objects and must run online in the vehicle. This causes information loss that could result in unsafe situations [30], e.g., if a solid object is below the detection threshold, or the future behavior of the object is not captured by the simplistic future trajectory estimates.

In recent years, *object-free* approaches [3, 12, 29, 30] that model the presence, location and future behavior of all agents in the scene via a non-parametric distribution have emerged to address the shortcomings of *object-based* models. Object-free approaches predict occupancy probability and motion for each cell in a spatio-temporal grid, directly from sensor data. More concretely, the spatio-temporal grid is a 3-dimensional dense grid with two spatial dimensions representing the bird's-eye view, and a temporal dimension from the current observation time to a future horizon of choice. All dimensions are quantized at regular intervals.

*Denotes equal contribution

In this paradigm, no detection confidence thresholding is required and the distribution over future motion is much more expressive, enabling the downstream motion planner to plan with consideration of low-probability objects and futures. Unfortunately, object-free approaches are computationally expensive as the grid must be very high-dimensional to mitigate quantization errors. However, most of the computation and memory employed in object-free methods is unnecessary, as motion planners only need to cost a set of spatio-temporal points around candidate trajectories, and not a dense region of interest (RoI). We refer the reader to Fig. 1 for an illustration.

This motivates our approach, IMPLICITO, which utilizes an implicit representation to predict both occupancy probability and flow over time directly from raw sensor data and HD maps. This enables downstream tasks such as motion planning to efficiently evaluate a large collection of *spatio-temporal query points* in parallel, focusing on areas of interest where there are potential interactions with the self-driving vehicle. We design an architecture that overcomes the limited receptive field of fully convolutional explicit architectures [12, 24, 29, 30] by adding an efficient yet effective global attention mechanism. In particular, we leverage deformable convolutions [8] and cross attention [37] to focus on a compact set of distant regions per query, giving the predictions a global context. This is useful as dynamic objects can move at very high speeds, particularly on the highway. For instance, when predicting in-lane occupancy 3 seconds into the future on a road where the speed limit is 30 m/s, the attention can look approximately 90 meters back along the lane to find the corresponding sensor evidence. Extensive experiments in both urban and highway scenarios show that our object-free implicit approach outperforms the two prevalent paradigms in the literature on the task of occupancy-flow prediction: (i) object-based methods that first perform object detection to localize a finite set of objects in the scene, and then predict their future trajectory distribution (ii) object-free explicit methods that predict dense spatio-temporal grids of occupancy and motion.

2. Related Work

In this section we discuss traditional *object-based* perception and prediction as well as *object-free* perception and prediction. We also outline literature in *implicit geometric reconstruction*, which inspired our approach.

Object-based Perception and Motion Forecasting: The majority of previous works have adopted object-based reasoning with a 2-stage pipeline, where first object detection [17, 42] and tracking [33, 38] are performed, followed by trajectory prediction from past tracks [4, 28, 35, 44]. As there are multiple possible futures, these methods either generate a fixed number of modes with probabilities and/or draw samples to characterize the trajectory distribution. This paradigm has three main issues [30, 36]: (1) uncertainty is not propagated from perception to downstream prediction, (2) the predicted future distributions must be overly compact in practice, as their size grows linearly with the number of objects, (3) thresholded decisions in perception make the planner blind to undetected objects. Several works [1, 2, 20, 23] tackle (1) by optimizing jointly through the multiple stages. However, (2) and (3) are fundamentally hard to address in this object-based paradigm as it implies a finite set of objects that can be large in crowded scenes. In contrast, our model is agnostic to the number of objects in the scene since it predicts the occupancy probability and flow vectors at desired spatio-temporal points.

Object-Free Perception and Prediction: These methods forecast future occupancy and motion from sensor data such as LiDAR [3, 30] and camera [12, 13, 29], *without considering individual actors*. P3 [30] first introduced temporal semantic occupancy grids as an interpretable intermediate representation for motion planning. MP3 [3] enhanced this representation by predicting an initial occupancy grid and warping it into the future with a spatio-temporal grid of multi-modal flow predictions. Compared to fully convolutional architectures, this flow-warping increases the effective receptive field for occupancy prediction, and imposes prior on how occupancy can evolve over time. However, forward flow struggles with dispersing occupancy over time when uncertainty increases, as pointed out in [24]. FIERY [12] added instance reasoning to the object-free paradigm as a postprocessing, improving interpretability. OCCFLOW [24] introduced *backwards flow* as a representation that can capture multi-modal forward motions with just one flow vector prediction per grid cell. However, OCCFLOW isolates the occupancy and flow *forecasting* problem by assuming input features (e.g., position, velocity, extent etc.) from a detection and tracking module, instead of raw sensor data.

While our work belongs to the category of object-free methods, our model only predicts occupancy-flow at select query points instead of outputting spatio-temporal occupancy-flow grids with fully convolutional networks. We achieve this with an efficient and effective global attention mechanism. This makes the model more expressive while improving efficiency by reducing the computation to only that which matters to downstream tasks.

Implicit Geometric Reconstruction: Geometric reconstruction refers to the task of predicting the 3D shape of an object given some incomplete representation of it, e.g., images, LiDAR, voxels. Implicit neural geometric reconstruction methods [5, 25, 26] have been shown to outperform explicit counterparts, which represent the 3D shape as a grid, set of points, voxels or a mesh. In contrast, implicit methods train a neural network to predict a continuous field that assigns a value to each point in 3D space, so that a shape can be extracted as an iso-surface. More concretely, this net-

work can predict non-linear binary occupancy [5, 25] over 3D space $f(x) : \mathbb{R}^3 \rightarrow [0, 1]$, or a signed distance function to the surface [26]. Our work is motivated by these ideas, and we explore their application to the task of occupancy and flow prediction for self-driving. Particularly, the architecture of our implicit prediction decoder is inspired by Convolutional Occupancy Networks [27], which proposed a translation equivariant approach to accurately reconstruct large scale scenes.

3. Implicit Perception and Future Prediction

Understanding the temporal occupancy and motion of traffic participants in the scene is critical for motion planning, allowing the self-driving vehicle (SDV) to avoid collisions, maintain safety buffers and keep a safe headway [3]. Previous methods [3, 12, 16, 24, 30] represent occupancy and motion in bird's-eye view (BEV) *explicitly* with a discrete spatio-temporal grid. This approach is resource inefficient, because it uses computational resources to predict in regions that are irrelevant to the SDV. In this section, we introduce IMPLICITO, an *implicit* neural network that can be queried for both scene occupancy and motion at any 3-dimensional continuous point (x, y, t). Here, x and y are spatial coordinates in BEV and $t = \bar{t} + \Delta t$ is the time into the future, where $\bar{t}$ refers to the current timestep at which we are making the predictions, and $\Delta t \geq 0$ is an offset from the current timestep into the future. This enables the motion planner to request the computation only at points around the candidate trajectories that are being considered. In the remainder of this section, we first describe the task parametrization, then the network architecture, and finally how to train our approach.

3.1. Task Parameterization

We discuss the task by defining its inputs and outputs.

Input parameterization: Our model takes as input a voxelized LiDAR representation (L) as well as a raster of the HD map (M). For the LiDAR, let $\mathcal{S}_{\bar{t}} = \{\mathbf{s}_{\bar{t}-T_{history}+1}, \ldots, \mathbf{s}_{\bar{t}}\}$ be the sequence of the most recent $T_{history} = 5$ sweeps. More precisely, $\mathbf{s}_{t'} \in \mathbb{R}^{P_{t'} \times 3}$ is the LiDAR sweep ending at timestep t' containing a set of $P_{t'}$ points, each of which described by three features: (p_x, p_y, p_h). p_x and p_y are the location of the point relative to the SDV's reference frame at the current timestep $\bar{t}$ — centered at the SDV's current position and with the x-axis pointing along the direction of the its heading. p_h corresponds to the height of the point above the ground. Finally, $L = Voxelize(\mathcal{S}_t) \in \mathbb{R}^{T_{history} D \times H \times W}$, where we follow the multi-sweep BEV voxelization proposed in [41] with a discretization of D depth channels normal to the BEV plane, H height pixels and W width pixels. For the raster map, we take the lane centerlines $\mathcal{C}$ represented as polylines from the high-definition map and rasterize them on a single channel $M = Raster(\mathcal{C}) \in \mathbb{R}^{1 \times H \times W}$ with the same spatial dimensions.

Output parameterization: Let $\mathbf{q} = (x, y, t) \in \mathbb{R}^3$ be a spatio-temporal point in BEV, at a future time t. The task is to predict the probability of occupancy $o : \mathbb{R}^3 \rightarrow [0, 1]$, and the flow vector $\mathbf{f} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ specifying the BEV motion of any agent that occupies that location. We model the *backwards flow* [24] for the flow vector $\mathbf{f}$, as it can capture multi-modal forward motions with a single reverse flow vector per grid cell. More concretely, backwards flow describes the motion at time t and location (x, y) as the translation vector at that location from $t - 1$ to t, should there be an object occupying it:

$$\mathbf{f}(x, y, t) = (x', y')_{t-1} - (x, y)_t, \tag{1}$$

where (x', y') denotes the BEV location at time $t - 1$ of the point occupying (x, y) at time t.

3.2. Network Architecture

We parameterize the predicted occupancy $\hat{o}$ and flow $\hat{\mathbf{f}}$ with a multi-head neural network ψ. This network takes as input the voxelized LiDAR L, raster map M, and a mini-batch $\mathcal{Q}$ containing $|\mathcal{Q}|$ spatio-temporal query points $\mathbf{q}$, and estimates the occupancy $\hat{\mathcal{O}} = \{\hat{o}(\mathbf{q})\}_{\mathbf{q} \in \mathcal{Q}}$ and flow $\hat{\mathcal{F}} = \{\hat{\mathbf{f}}(\mathbf{q})\}_{\mathbf{q} \in \mathcal{Q}}$ for the mini-batch in parallel:

$$\hat{\mathcal{O}}, \hat{\mathcal{F}} = \psi(L, M, \mathcal{Q}) \tag{2}$$

The network ψ is divided into a convolutional encoder that computes scene features, and an implicit decoder that outputs the occupancy-flow estimates, as shown in Fig. 2.

Inspired by [42], our encoder consists of two convolutional stems that process the BEV LiDAR and map raster, a ResNet [11] that takes the concatenation of the LiDAR and map raster features and outputs multi-resolution feature planes, and a lightweight Feature Pyramid Network (FPN) [21] that processes these feature planes. This results in a BEV feature map at half the resolution of the inputs, i.e., $\mathbf{Z} \in \mathbb{R}^{C \times \frac{H}{2} \times \frac{W}{2}}$, that contains contextual features capturing the geometry, semantics, and motion of the scene. It is worth noting that every spatial location (feature vector) in the feature map $\mathbf{Z}$ contains spatial information about its neighborhood (the size of the receptive field of the encoder), as well as temporal information over the past $T_{history}$ seconds. In other words, each feature vector in $\mathbf{Z}$ may contain important cues regarding the motion, the local road geometry, and neighboring agents.

We design an *implicit occupancy and flow decoder* motivated by the intuition that the occupancy at query point $\mathbf{q} = (x, y, t) \in \mathcal{Q}$ might be caused by a distant object moving at a fast speed prior to time t. Thus, we would like to use the local features around the spatio-temporal query location

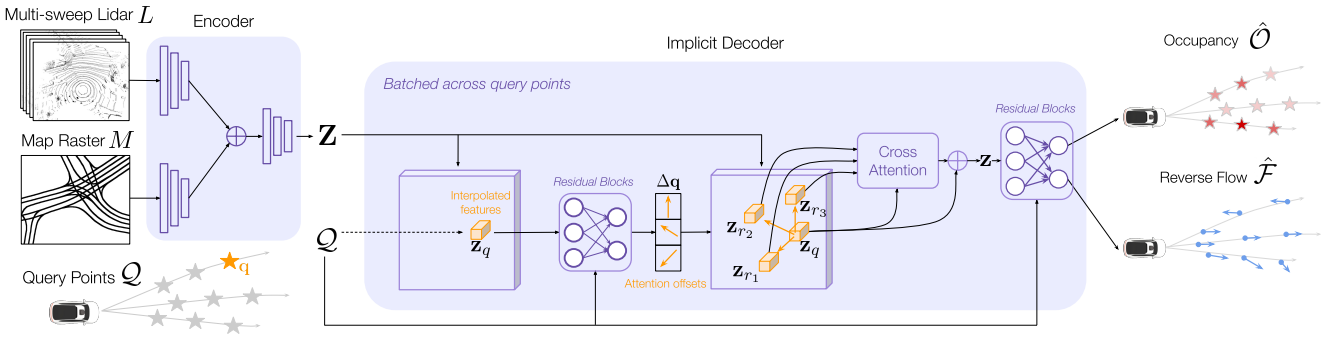

Figure 2. An overview of our model, IMPLICITO. Voxelized LiDAR and an HD map raster are encoded by a two-stream CNN. The resulting feature map $\mathbf{Z}$ is used by the decoder to obtain relevant features for the query points $\mathcal{Q}$ and eventually predict occupancy $\hat{\mathcal{O}}$ and reverse flow $\hat{\mathcal{F}}$. We illustrate the attention for a single query point $\mathbf{q}$, but the inference is fully parallel across query points $\mathcal{Q}$.

to suggest where to look next. For instance, there might be more expressive features about an object around its original position (at times $\{(\bar{t}-T_{history}+1),\ldots,\bar{t}\}$) since that is where the LiDAR evidence is. Neighboring traffic participants that might interact with the object occupying the query point at time t are also relevant to look for (e.g., lead vehicle, another vehicle arriving at a merging point at a similar time).

To implement these intuitions, we first bi-linearly interpolate the feature map $\mathbf{Z}$ at the query BEV location $\mathbf{q}_{x,y} = (x, y)$ to obtain the feature vector $\mathbf{z}_q = Interp(\mathbf{Z}, x, y) \in \mathbb{R}^C$ that contains local information around the query. We then predict K reference points $\{\mathbf{r}_1,\ldots,\mathbf{r}_K\}$ by offseting the initial query point $\mathbf{r}_k = \mathbf{q} + \Delta\mathbf{q}_k$, where the offsets $\Delta\mathbf{q}$ are computed by employing the fully connected ResNet-based architecture proposed by Convolutional Occupancy Networks [27]. For all offsets we then obtain the corresponding features $\mathbf{z}_{r_k} = Interp(\mathbf{Z}, \mathbf{r}_k)$. This can be seen as a form of deformable convolution [8]; a layer that predicts and adds 2D offsets to the regular grid sampling locations of a convolution, and bi-linearly interpolates for feature vectors at those offset locations. To aggregate the information from the deformed sample locations, we use cross attention between learned linear projections of $\mathbf{z}_q \in \mathbb{R}^{1\times C}$ and $\mathbf{Z}_r = \{\mathbf{z}_{r_1},\ldots,\mathbf{z}_{r_k}\} \in \mathbb{R}^{K\times C}$. The result is the aggregated feature vector $\mathbf{z}$. See Fig. 2 for a visualization of this feature aggregation procedure. Finally, $\mathbf{z}$ and $\mathbf{z}_q$ are concatenated, which, along with $\mathbf{q}$, is processed by another fully connected ResNet-based architecture with two linear layer heads to predict occupancy logits and flow. Please see additional implementation details and a full computational graph of our model in the supplementary.

3.3. Training

We train our implicit network by minimizing a linear combination of an occupancy loss and a flow loss

$$\mathcal{L} = \mathcal{L}_o + \lambda_{\mathbf{f}}\mathcal{L}_{\mathbf{f}}. \tag{3}$$

Occupancy is supervised with binary cross entropy loss $\mathcal{H}$ between the predicted and the ground truth occupancy at each query point $\mathbf{q} \in \mathcal{Q}$,

$$\mathcal{L}_o = \frac{1}{|\mathcal{Q}|}\sum_{\mathbf{q}\in\mathcal{Q}} \mathcal{H}(o(\mathbf{q}), \hat{o}(\mathbf{q})), \tag{4}$$

where $o(\mathbf{q})$ and $\hat{o}(\mathbf{q})$ are ground truth and predicted occupancy and query point $\mathbf{q}$, respectively. The ground truth labels are generated by directly calculating whether or not the query point lies within one of the bounding boxes in the scene. We supervised the flow only for query points that belong to foreground, i.e., points that are occupied. By doing so, the model learns to predict the motion of a query location should it be occupied. We use the ℓ_2 error, where the labels are backwards flow targets from t to $t-1$ computed as rigid transformations between consecutive object box annotations:

$$\mathcal{L}_f = \frac{1}{|\mathcal{Q}|}\sum_{\mathbf{q}\in\mathcal{Q}} o(\mathbf{q})||\mathbf{f}(\mathbf{q}) - \hat{\mathbf{f}}(\mathbf{q})||_2. \tag{5}$$

We train with a batch of continuous query points $\mathcal{Q}$, as opposed to points on a regular grid as previously proposed. More concretely, for each example we sample $|\mathcal{Q}|$ query points uniformly across the spatio-temporal volume $[0, H]\times[0, W]\times[0, T]$, where $H \in \mathbb{R}$ and $W \in \mathbb{R}$ are the height and width of a rectangular region of interest (RoI) in BEV surrounding the SDV, and $T \in \mathbb{R}$ is the future horizon we are forecasting.

4. Experiments

In this section, we introduce the datasets and metrics used to benchmark occupancy-flow perception and prediction, and show that IMPLICITO outperforms the state-of-the-art in both urban and highway settings. Further, we conduct two ablations studies to understand the effect of our contributions to the decoder architecture, and an analy-

sis of the inference time of our implicit decoder compared to explicit alternatives.

Datasets: We conduct our experiments using two datasets: Argoverse 2 Sensor [40] (urban), and HighwaySim (highway). The Argoverse 2 Sensor (AV2) dataset is collected in U.S. cities and consists of 850 fifteen-second sequences with sensor data from two 32-beam LiDARs at a frequency of 10 Hz, high-definition maps with lane-graph and ground-height data, and bounding box annotations. We split the set into 700 sequences for training and 150 for validation, and break the sequences into examples that include 5 frames of LiDAR history and a prediction time horizon of 5 seconds. In our experiments, we only consider the occupancy and flow of vehicles, which we define as the union of the following AV2 annotation classes: regular vehicle, large vehicle, wheeled device, box truck, truck, vehicular trailer, truck cab, school bus, articulated bus, message-board trailer and railed vehicle. Query points are labeled with occupancy by checking if they intersect with the annotated bounding boxes. Occupied query points are labeled with flow vectors using finite differences between the current query point and where that point was in the previous frame. Incomplete tracks caused by missing annotations were filled-in using the constant turn rate and acceleration (CTRA) motion model [18], so the models learn the prior that occupancy is persistent. HighwaySim (HwySim) is a dataset generated with a state-of-the-art simulator, containing realistic highway traffic scenarios including on-ramps, off-ramps, and curved roads. A Pandar64 LiDAR is realistically simulated, and maps with lane-graph and ground-height are provided. 700 sequences of around 15 seconds each are split 80/20 into training/validation. Sequences are cut into examples, each with a history of 5 past LiDAR frames and a 5 s future horizon.

Metrics: To be fair with the baselines, we evaluate all models with query points on a regular spatio-temporal grid. Temporally, we evaluate a prediction horizon of 5 seconds with a resolution of 0.5 seconds for both datasets. In AV2, we employ a rectangular RoI of 80 by 80 meters centered around the SDV position at time $\bar{t}$ with a spatial grid resolution of 0.2 m. In HwySim, we use an asymmetric ROI with 200 meters ahead and 40 meters back and to the sides of the SDV at time $\bar{t}$ with a grid resolution of 0.4 m. This is to evaluate on highway vehicles moving fast (up to 30 m/s) in the direction of the SDV over the full prediction horizon. For simplicity, we refer to the grid cell centroids as "query-points". We evaluate the ability of the models to recover the ground-truth occupancy-flow. In particular, we utilize metrics to measure the precision, recall, accuracy and calibration of the occupancy, the flow errors, and the consistency between the occupancy and flow predictions.

Mean average precision (mAP): mAP captures if the model correctly predicts a higher occupancy probability in occupied regions relative to unoccupied regions, i.e., an accurate ranking of occupancy probability. mAP is computed as the area under the precision recall curve averaged across all timesteps in the prediction horizon.

Soft-IoU: We follow prior works [16, 24, 34] in the use of *soft intersection over union* for assessing occupancy predictions:

$$\text{Soft-IoU} = \frac{\sum_{\mathbf{q}\in\mathcal{Q}} o(\mathbf{q})\hat{o}(\mathbf{q})}{\sum_{\mathbf{q}\in\mathcal{Q}}(o(\mathbf{q}) + \hat{o}(\mathbf{q}) - o(\mathbf{q})\hat{o}(\mathbf{q}))}. \quad (6)$$

Unlike mAP, Soft-IoU also considers the magnitude of predicted occupancy probability instead of just the predicted probability ranking.

Expected Calibration Error (ECE): ECE measures the expected difference between model confidence and accuracy. This is desirable because the occupancy outputs may be used by downstream planners in a probabilistic way — e.g., to compute the expected collision cost [3]. Thus, we need to understand if the predicted probabilities are poorly calibrated, i.e., suffering from over-confidence or under-confidence [10, 22].

Foreground mean end-point-error (EPE): This metric measures the average L2 flow error at each occupied query point:

$$\text{EPE} = \frac{1}{\sum_{\mathbf{q}\in\mathcal{Q}} o(\mathbf{q})} \sum_{\mathbf{q}\in\mathcal{Q}} o(\mathbf{q})||\mathbf{f}(\mathbf{q}) - \hat{\mathbf{f}}(\mathbf{q})||_2. \quad (7)$$

Flow Grounded Metrics: Let O_t, $\hat{O}_t$, and $\hat{F}_t$ denote the occupancy labels, predicted occupancy, and predicted flow on a spatio-temporal grid at time t, respectively. The flow grounded occupancy grid at timestep $t > \bar{t}$, is obtained by warping the ground truth occupancy grid at the previous timestep O_{t-1} with the predicted flow field $\hat{F}_t$, and multiplying it element-wise with the predicted occupancy at the current timestep $\hat{O}_t$ [34]. We report flow-grounded Soft-IoU and mAP by comparing the flow-grounded occupancy to the occupancy ground truth. The flow grounded metrics are useful for evaluating the consistency between the occupancy and flow predictions, as you can only achieve a high score if (1) the flow is accurate and (2) the warped ground-truth occupancy aligns well with the predicted occupancy for the next time step.

Inference Time: When measuring inference time, all methods were implemented with vanilla PyTorch code (no custom kernels) and run on a single Nvidia GeForce GTX 1080 Ti. This metric is sensitive to implementation, hardware, and optimizations, and thus should not be compared across different works.

Baselines: We compare against five baselines that cover the different perception and prediction paradigms outlined in the Sec. 1. MULTIPATH [4], LANEGCN [19], and GORELA [7] are object-based trajectory prediction models.

	AV2						HwySim					
	mAP ↑	Soft-IoU ↑	ECE ↓	EPE ↓	Flow Grounded		mAP ↑	Soft-IoU ↑	ECE ↓	EPE ↓	Flow Grounded	
					mAP ↑	Soft-IoU ↓					mAP ↑	Soft-IoU ↓
MULTIPATH [4]	0.625	0.398	0.916	0.982	0.803	0.321	0.299	0.231	0.433	4.227	0.463	0.154
LANEGCN [19]	0.620	0.449	1.138	0.709	0.778	0.350	0.472	0.283	0.337	2.951	0.636	0.194
GORELA [7]	0.609	0.453	1.161	0.671	0.813	0.355	0.548	0.259	0.288	2.206	0.722	0.166
OCCFLOW [24]	0.675	0.356	0.348	0.390	0.886	0.493	0.597	0.370	0.260	0.842	0.841	0.330
MP3 [3]	0.774	0.422	0.201	0.472	0.902	0.466	0.637	0.246	0.208	1.172	0.833	0.193
IMPLICITO	**0.799**	**0.480**	**0.193**	**0.267**	**0.936**	**0.597**	**0.716**	**0.415**	**0.076**	**0.510**	**0.886**	**0.492**

Table 1. Comparing our proposed model IMPLICITO to state-of-the-art perception and prediction models on AV2 and HwySim. The first three rows are object-based models, while the others are object-free.

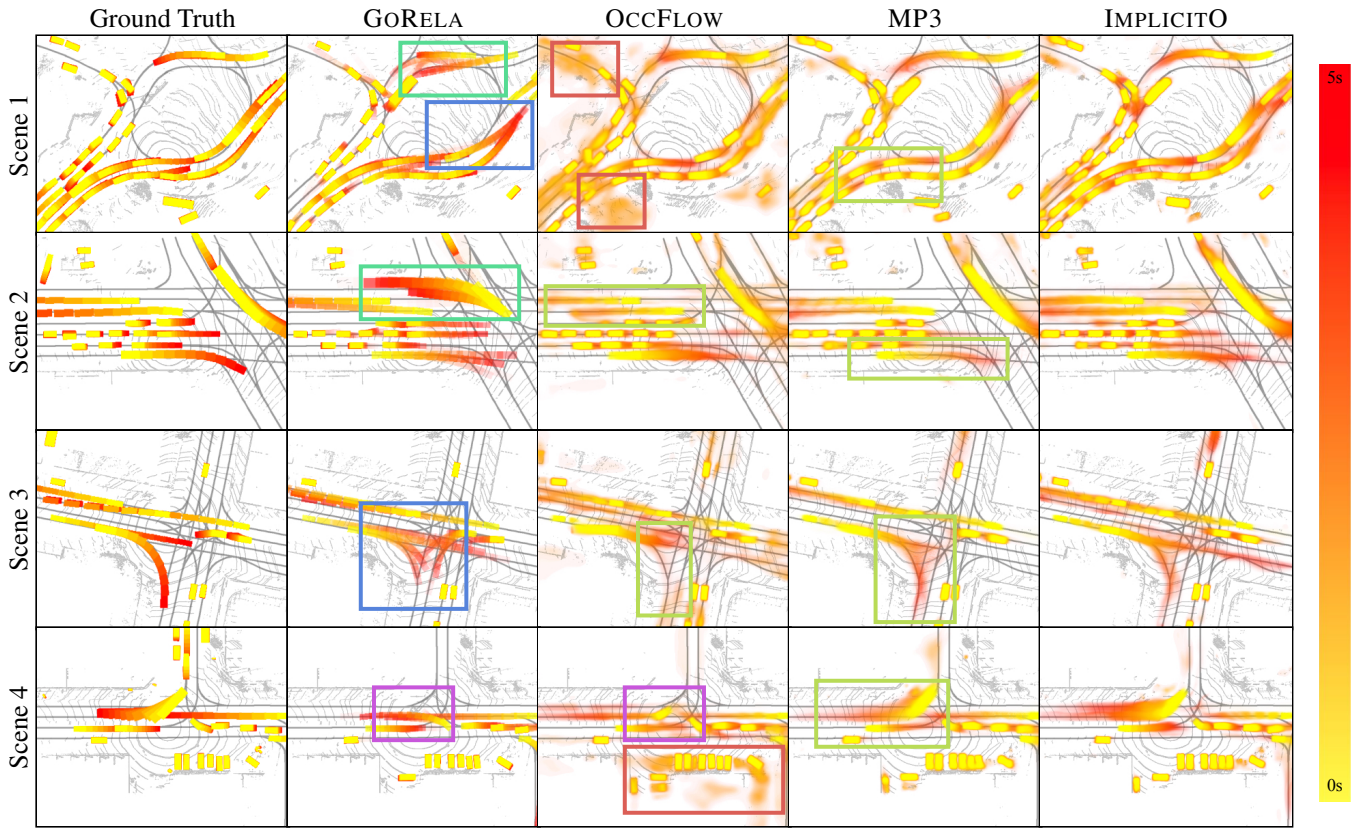

Figure 3. Occupancy predictions of various models (columns) across four scenes (rows) in AV2. Opacity denotes occupancy probability, and the colormap indicates prediction time Δt (from current to future horizon, as shown on the right). Failure modes are highlighted with colored boxes: **occupancy hallucination**, **fading/missing occupancy**, **inconsistent with map**, **inconsistent with actors**, **miss-detection**.

Following [16, 22], to evaluate these object-based models on the task of occupancy-flow prediction, we rasterize the trajectory predictions to generate occupancy and flow fields. For occupancy, we rasterize the multi-modal trajectory predictions weighted by the mode probabilities. For flow, we generate a multi-modal spatio-temporal flow field, where for each mode, a grid cell predicted to be occupied by an object contains the forward-flow rigid-transformations defined by the trajectory of that object. OCCFLOW uses the occupancy-flow prediction architecture and *flow-traced loss* from Mahjourian et al. [24], using input features from a pre-trained detection and tracking module. More information on the detector can be found in the supplementary. MP3 [3] is an end-to-end trained perception and prediction method that predicts multi-modal forward-flow vectors and associated probabilities, and uses these to warp a predicted initial occupancy grid forward in time. We compute EPE on the expected motion vector (the probability-weighted sum of modes) when evaluating MP3.

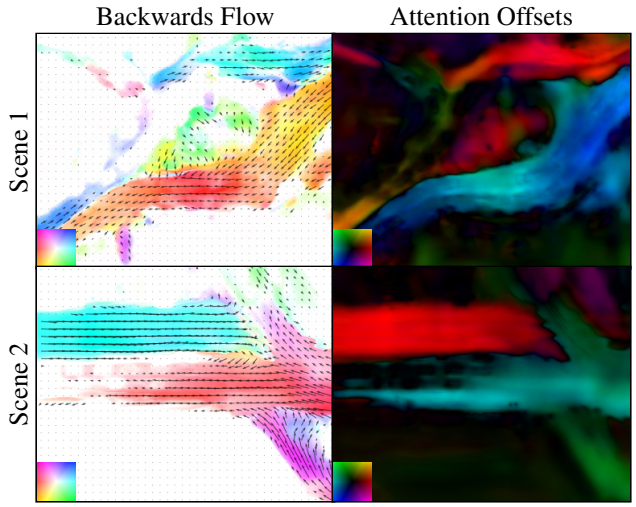

Figure 4. Visualizations of the backwards flow field predictions and attention offset predictions of IMPLICITO at the last timestep of the prediction horizon on Scene 1 and Scene 2 from Fig. 3.

Benchmark against state-of-the-art: Tab. 1 presents our results on AV2 and HwySim against the state-of-the-art baselines described above. For this experiment, our model IMPLICITO predicts $K = 1$ attention offset. Our method outperforms all others across all metrics and both datasets, showing the suitability of IMPLICITO in both urban and highway settings. Fig. 3 displays qualitative results of these models on AV2. Notice that all the object-based models generally under-perform relative to the object-free approaches. This is likely because these models are not optimized for occupancy-flow directly, rather they are trained to predict accurate trajectories. The qualitative results of GORELA in Fig. 3 show that thresholding to produce instances can result in missed detections (Scene 4). Further, the trajectory parameterization results in rasterized occupancy that is more often inconsistent with the map (Scenes 1 and 2), or inconsistent due to apparent collisions with other actors (Scenes 1 and 3). This agrees with the results from [16], and reaffirms the utility of the object-free parameterization. Interestingly, on AV2, the object-based approaches have a high Soft-IoU despite their inaccurate occupancy ranking. We find this is because these models are overconfident (reflected in their high ECE), which is heavily rewarded by Soft-IoU on the many "easy" examples in AV2 with stationary vehicles (in the evaluation set, 64.4% of actors are static within the prediction horizon). This is supported by the worse relative Soft-IoU of these object-based models on HwySim, which has a much higher proportion of dynamic actors. Interestingly MP3 outperforms OCCFLOW in the joint perception and prediction setting, contrary to the results under perfect perception assumption reported by [24]. We hypothesize this is because MP3 is trained end-to-end and does not have the intermediate object-based detection representation. We can see in Fig. 3 that OCCFLOW hallucinates occupancy at the initial timestep (Scenes 1 and 4), and misses the detection of a vehicle in Scene 4, both of which are artifacts of training with input from a pre-trained detection model.

		mAP ↑	Soft-IoU ↑	ECE ↓	EPE ↓	Flow Grounded	
						mAP ↑	Soft-IoU ↑
AV2	MP3 [3]	0.774	0.422	0.201	0.472	0.902	0.466
	CONVNET	0.796	0.466	**0.135**	0.312	0.929	0.581
	CONVNETFT [24]	0.796	0.475	0.198	0.302	0.929	0.582
	IMPLICITO	**0.799**	**0.480**	0.193	**0.267**	**0.936**	**0.597**
HwySim	MP3 [3]	0.637	0.246	0.208	1.172	0.833	0.193
	CONVNET	0.648	0.344	**0.024**	0.657	0.859	0.408
	CONVNETFT [24]	0.654	0.351	**0.024**	0.657	0.860	0.416
	IMPLICITO	**0.716**	**0.415**	0.076	**0.510**	**0.886**	**0.492**

Table 2. Comparing the performance of occupancy-flow decoders, trained end-to-end, with the same encoder.

Flow and Attention visualization: Fig. 4 plots the reverse flow vectors as well as the attention offsets on two of the scenes from Fig. 3 (the middle two rows). The first observation is that the flow vectors and attention offsets rely very heavily on the map raster, as expected. The second observation is that the direction of the backward flow vectors and the attention offsets are very heavily correlated. This shows that the model has learned to "look backwards" along the lanes to gather relevant features despite the offsets being unsupervised. We hypothesize that IMPLICITO outperforms the others because of its larger effective receptive field. Fig. 3 shows that IMPLICITO maintains occupancy into the future more accurately than MP3. We attribute this to the attention offsets being a more general and expressive mechanism than MP3's forward flow warping. To illustrate this further, in the supplementary we plot occupancy metrics as a function of prediction time Δt.

Influence of the decoder architecture: In this section, we compare various occupancy-flow decoders, all trained end-to-end from LiDAR input with the same encoder architecture as IMPLICITO (described in Sec. 3.2). This allows us to isolate the effect of our implicit decoder architecture design. CONVNET implements the decoder from Mahjourian et al. [24], but it takes as input a feature map from the encoder, instead of hand-crafted detection features. CONVNETFT denotes this same decoder architecture trained with the auxiliary supervision of flow trace loss [24]. Note that MP3 and IMPLICITO from Tab. 1 already use this encoder and are trained end-to-end, so the same results are presented for this ablation study. As shown in Tab. 2 our implicit decoder IMPLICITO outperforms all the other decoders across all metrics except for ECE, on both HwySim and AV2. Notice that CONVNET and CONVNETFT outperform their detection-based counterpart OCCFLOW in Tab. 1 by a significant margin. This highlights the utility of end-to-

	Num. offsets	mAP ↑	Soft-IoU ↑	ECE ↓	EPE ↓	Flow Grounded	
						mAP ↑	Soft-IoU ↑
AV2	$K = 0$	0.790	0.456	**0.128**	0.300	0.930	0.583
	$K = 1$	**0.799**	**0.480**	0.193	0.267	**0.936**	**0.597**
	$K = 4$	0.797	0.478	0.257	**0.252**	**0.936**	0.570
HwySim	$K = 0$	0.649	0.359	0.052	0.686	0.857	0.421
	$K = 1$	**0.716**	**0.415**	0.076	0.510	0.886	**0.492**
	$K = 4$	0.714	0.404	**0.051**	**0.509**	**0.890**	0.487

Table 3. Ablation study on the effect of the number of predicted attention offsets on the performance of IMPLICITO.

end training in the object-free paradigm for occupancy-flow prediction. Evidently, thresholding to produce detections and hand-crafted features limits the information available for occupancy-flow perception and prediction. Again, we hypothesize that IMPLICITO outperforms the others due to its offset mechanism increasing the effective receptive field. Even with the powerful encoder and flow warping mechanism, CONVNET and MP3 fail to match this. This is supported by the relatively close performance of CONVNET to IMPLICITO on AV2, but not HwySim. On HwySim most vehicles travel larger fraction of the ROI, so a larger effective receptive field is necessary. On AV2 more vehicles are stationary or move slowly, so a large receptive field is less important for occupancy-flow prediction.

Influence of the number of offsets (K): Based on the attention offset visualizations in Fig. 4, we have conjectured that the predicted attention offsets of our implicit decoder are responsible for its state-of-the-art performance. In this section, we ablate the number of predicted offsets of IMPLICITO to investigate this further. Tab. 3 reports results for implicit decoders with a different number of attention offsets. $K = 0$ denotes no attention offset, predicting occupancy-flow from $\mathbf{z_q}$ alone without cross-attention (see Fig. 2). We first note that $K = 1$ clearly outperforms $K = 0$, particularly on HwySim. This aligns with our intuition that the main function of the attention offsets is to expand the receptive field of a query point. Since vehicles travel at much lower speeds in urban than highway, AV2 has a lower effective receptive field requirement than HwySim and thus the improvements are not as pronounced. We observe fairly close and mixed results between one and four attention offsets. $K = 1$ has the best occupancy prediction metrics, while $K = 4$ is the best in some flow metrics. Under the assumption that the predicted offsets look back to where occupancy could have come from in the past, $K > 1$ would only improve performance over $K = 1$ when occupancy could come from more than one past location (e.g., complex intersections, Scene 2 of Fig. 3). These examples are rare in the training and evaluation datasets, and having redundant offsets in the simple cases where one offset suffices could introduce noise, explaining why $K = 4$ does not outperform $K = 1$. See the supplementary for visualizations of the attention offsets when $K = 4$.

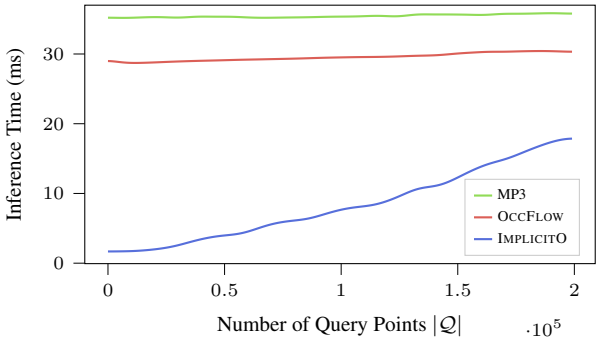

Figure 5. Decoder inference time as a function of the number of query points for the object-free decoders presented in Tab. 1 on HwySim. IMPLICITO uses $K = 1$.

Inference Time Comparison: In this section we compare the decoder inference time of explicit object-free methods in the literature (from Tab. 1) against the decoder of IMPLICITO with $K = 1$. Fig. 5 presents the inference time as a function of the number of query points. For the explicit models (MP3, OCCFLOW), this includes the time to bi-linearly interpolate occupancy probability at the continuous query points. The plot evaluates on query points in a range $(1, 2{\cdot}10^5)$, that most planners will operate within. For instance, $2,000$ candidate trajectories $\times$ 10 timesteps per trajectory is well aligned with the literature [3, 29–31, 43]. With $200,000$ trajectories, this allows for 10 queries per timestep to integrate occupancy over the volume of the SDV, which should provide a good estimation of collision probability. We notice that for $\lesssim 20,000$ query points, IMPLICITO has a constant inference time because it is batched over query points. Once the GPU is saturated, the operations are run sequentially so inference time increases approximately linearly. The explicit decoders have approximately constant inference times (the only increase is due to bilinear interpolation), but are significantly slower than IMPLICITO in this "planner-relevant" range.

5. Conclusion

In this paper, we have proposed a unified approach to joint perception and prediction for self-driving that implicitly represents occupancy and flow over time with a neural network. This queryable implicit representation can provide information to a downstream motion planner more effectively and efficiently. We showcased that our implicit architecture predicts occupancy and flow more accurately than contemporary explicit approaches in both urban and highway settings. Further, this approach outperforms more traditional object-based perception and prediction paradigms. In the future, we plan to assess the impact of our improvements on the downstream task of motion planning.

References

[1] Sergio Casas, Cole Gulino, Simon Suo, Katie Luo, Renjie Liao, and Raquel Urtasun. Implicit latent variable model for scene-consistent motion forecasting. In *ECCV*, 2020. 2

[2] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *CoRL*, 2018. 1, 2

[3] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7, 8

[4] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019. 2, 5, 6

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *CVPR*, 2019. 2, 3

[6] Alexander Cui, Sergio Casas, Abbas Sadat, Renjie Liao, and Raquel Urtasun. Lookout: Diverse multi-future prediction and planning for self-driving. In *ICCV*, 2021. 1

[7] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. *arXiv preprint arXiv:2211.02545*, 2022. 5, 6

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 2, 4

[9] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *ICCV*, 2021. 1

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[12] 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. 1, 2, 3

[13] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. St-p3: End-to-end vision-based autonomous driving via spatial-temporal feature learning. In *ECCV*, 2022. 2

[14] Boris Ivanovic, Karen Leung, Edward Schmerling, and Marco Pavone. Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach. *RA-L*, 2020. 1

[15] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *ICCV*, 2019. 1

[16] Jinkyu Kim, Reza Mahjourian, Scott Ettinger, Mayank Bansal, Brandyn White, Ben Sapp, and Dragomir Anguelov. Stopnet: Scalable trajectory and occupancy prediction for urban autonomous driving. In *ICRA*, 2022. 3, 5, 6, 7

[17] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 2

[18] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH*, 2014. 5

[19] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *ECCV*, 2020. 5, 6

[20] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *CVPR*, 2020. 1, 2

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3

[22] Katie Luo, Sergio Casas, Renjie Liao, Xinchen Yan, Yuwen Xiong, Wenyuan Zeng, and Raquel Urtasun. Safety-oriented pedestrian motion and scene occupancy forecasting. 2021. 5, 6

[23] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *CVPR*, 2018. 2

[24] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *RA-L*, 2022. 2, 3, 5, 6, 7

[25] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2, 3

[26] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2, 3

[27] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 3, 4

[28] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *CVPR*, 2020. 2

[29] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, 2020. 1, 2, 8

[30] Abbas Sadat, Sergio Casas, Mengye Ren, Xinyu Wu, Pranaab Dhawan, and Raquel Urtasun. Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. In *ECCV*, 2020. 1, 2, 3, 8

[31] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *IROS*, 2018. 8

[32] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *ECCV*, 2020. 1

[33] Sarthak Sharma, Junaid Ahmed Ansari, J Krishna Murthy, and K Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *ICRA*, 2018. 2

[34] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020. 5

[35] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. In *Advances in Neural Information Processing Systems*, 2019. 2

[36] Ameni Trabelsi, Ross J Beveridge, and Nathaniel Blanchard. Drowned out by the noise: Evidence for tracking-free motion prediction. *arXiv preprint arXiv:2104.08368*, 2021. 2

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

[38] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IROS*, 2020. 1, 2

[39] Xinshuo Weng, Ye Yuan, and Kris Kitani. Ptp: Parallelized tracking and prediction with graph neural networks and diversity sampling. *RA-L*, 2021. 1

[40] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS*, 2021. 5

[41] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *CoRL*, 2018. 3

[42] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 2018. 2, 3

[43] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *CVPR*, 2019. 8

[44] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *CVPR*, 2019. 2