UniOcc: A Unified Benchmark for Occupancy Forecasting and Prediction in Autonomous Driving

Supplementary Material

6. Broader Impacts

Our unified occupancy framework, UniOcc, holds significant promise for improving safety and efficiency in autonomous driving systems. By standardizing occupancy labels and providing voxel-level flow annotations across real and simulated domains, UniOcc supports a broad spectrum of perception and forecasting tasks. This paves the way for more robust multi-modal reasoning and trajectory planning.

Our work aligns with a broader research agenda that spans generative and discriminative modeling for autonomous driving. For instance, the incorporation of occlusion-free LiDAR from aerial-ground collaborative platforms like AirV2X [7] and novel view synthesis [10] can further enhance scene understanding in the occupancy space. The ability to reason about partial observations, as explored in point-based reasoning frameworks [4, 13, 17–19, 52, 54], complements UniOcc's handling of voxel-level sparsity. Moreover, the integration of large vision-language models (VLMs) for driving scene understanding [14, 25, 27, 45, 47, 48, 57] opens new directions for using language and retrieval-augmented feedback in occupancy prediction. Relatedly, our efforts in benchmarking trustworthiness in VLMs for driving [8, 44, 46] can guide future work on safety-critical deployment of such models in open-world environments.

Finally, the unified occupancy modeling in UniOcc provides a strong foundation for generative world models [22, 39] and equivariant scene reasoning [37, 38], which are essential for long-horizon forecasting and generalization across cities and sensor setups. We believe that UniOcc will accelerate the development and evaluation of both classical and foundation-model-driven autonomous driving systems.

7. Algorithm Details

7.1. Occupancy Flow Computation

We discuss the details of our occupancy flow computation introduced in Section 3.1.

1. **Dynamic Foreground.** At time t, we gather all voxels coordinates $V^t_{n,d} \in \mathbb{R}^{n\times 3}$ that belong to a labeled agent a. We also have the agent-to-ego transformations at times t and $t+1$, namely $T^{e,t}_{a,t}$ and $T^{e,t+1}_{a,t+1}$, as well as the ego-to-ego transformation $T^{e,t+1}_{e,t} = (T^w_{e,t+1})^{-1} T^w_{e,t}$. We compute the agent's frame transformation as:

$$T^{a,t+1}_{a,t} = \left(T^{e,t+1}_{a,t+1}\right)^{-1} T^{e,t+1}_{e,t}\, T^{e,t}_{a,t}, \tag{13}$$

whose inverse gives the per-voxel motion in the a,t frame:

$$M^{a,t+1}_{a,t} = \left(T^{a,t+1}_{a,t}\right)^{-1}. \tag{14}$$

We then transform each voxel $V^t_{n,d}$ to predict its position $\widetilde{V^{t+1}_{n,d}}$ at $t+1$:

$$\widetilde{V^{t+1}_{n,d}} = T^{e,t+1}_{a,t+1}\, T^{a,t+1}_{a,t}\, M^{a,t+1}_{a,t}\, T^{a,t}_{e,t}\, V^t_{n,d}. \tag{15}$$

The flow field is then given by:

$$F^t_{n,d} = \widetilde{V^{t+1}_{n,d}} - V^t_{n,d}, \tag{16}$$

expressed in the ego coordinate frame by default, though we optionally provide an agent-centric flow variant for rotation-invariant models [37, 56].

2. **Static Background.** For each static voxel $V^t_{n,s}$, we compute its coordinates at $t+1$ using:

$$\widetilde{V^{t+1}_{n,s}} = \left(T^w_{e,t+1}\right)^{-1} T^w_{e,t}\, V^t_{n,s}, \tag{17}$$

and obtain $F^t_{n,s}$ as in Eq. (16).

For 2D flows, we begin with 2D grids and apply the same transformation steps outlined above.

7.2. GMM for Shape Comparison

We discuss the details of the GMM model fitting process in this section.

Let $D_c \in \mathbb{R}^{n\times 3}$ be the n dimensions for a category c (for example *Car*). We wish to fit a GMM with K components. Each GMM is parameterized by:

$$\text{GMM}_c : \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K,$$

where

- π_k are the mixing coefficients, with $\sum_{k=1}^K \pi_k = 1$.
- $\boldsymbol{\mu}_k \in \mathbb{R}^3$ are the mean vectors.
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3\times 3}$ are the covariance matrices (usually symmetric and positive-definite).

We fit the GMM parameters via the standard expectation-maximization(EM) algorithm so that each GMM component is "pulled" toward the points with high responsibility for the component. Rather than manually setting the number of components K and the covariance type, we perform a hyperparameter search using the Bayesian Information Criterion (BIC). Given a dataset of bounding box dimensions for a category, we fit multiple GMMs with

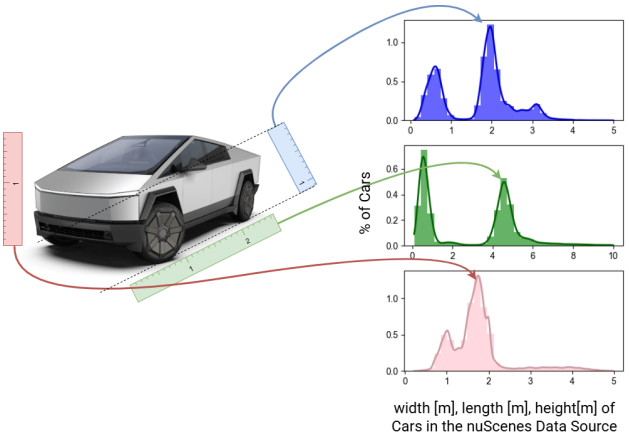

Figure 7. We compute the probability of a car by comparing its dimensions to the dimensional distribution of all cars in the data source.

varying numbers of components K and covariance types to find the optimal configuration.

Let $\mathcal{B}$ be the set of extracted bounding box dimensions:

$$\mathcal{B} = \{< l_i, w_i, h_i > | \; i = 1, \ldots, n\},$$

where l, w, h denote length, width, and height. To introduce robustness to the dimension data from occupancy predictions, we apply slight random perturbations corresponding to the voxel resolution:

$$l \leftarrow l + \epsilon_l, \quad w \leftarrow w + \epsilon_w, \quad h \leftarrow h + \epsilon_h, \quad \epsilon \sim \mathcal{U}(-\frac{\epsilon}{2}, \frac{\epsilon}{2}).$$

We then perform a grid search over:

- $K \in \{1, 2, \ldots, 20\}$ (number of components)
- Covariance types: *spherical*, *tied*, *diag*, *full*

For each candidate model, we compute the BIC score and select the best configuration:

$$\hat{K}, \hat{\Sigma} = \arg\min_{K,\Sigma} \text{BIC}(K, \Sigma).$$

This ensures that the selected GMM balances model complexity and data fit. With the learned GMM, we can evaluate how well each detected object fits the learned shape distribution, enabling object classification and outlier identification.

Once we acquire the GMM distribution, we measure the dimension of the predicted/forecasted objects and compare it with the distribution to estimate its probability of being a true object corresponding to that category. Figure 7 illustrates this process.

7.3. Ego Localization in the Occupancy Space

By design, the results of 3D occupancy forecasting do not inherently include the future poses of the ego vehicle. However, such information is critical for downstream tasks such as object tracking, behavior prediction, and motion planning. To address this gap, we propose a method to estimate the ego motion directly from a sequence of occupancy grids and associated flow fields. Specifically, we leverage the 3D flow vectors observed on static voxels (*e.g.*, road), under the assumption that their apparent motion arises solely from the ego vehicle's movement between frames.

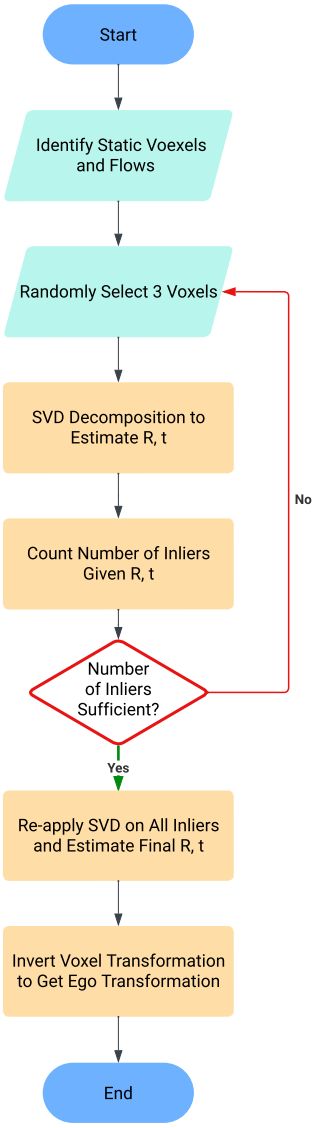

Figure 8. Occupancy space localization steps.

Given a temporal sequence of dense 3D occupancy grid $G \in \{0, \ldots, C\}^{T \times L \times W \times H}$ and a corresponding flow field $F \in \mathbb{R}^{T \times L \times W \times H \times 3}$, we isolate static scene elements and estimate the ego vehicle's $SE(3)$ transformation via robust point cloud registration. This procedure is performed independently at each frame to recover the frame-to-frame ego motion.

7.3.1. Estimating Rigid Transform from 3D Motion

Given that each voxel in the forecasted 3D occupancy grid is associated with a semantic class, we can isolate voxels corresponding to static scene elements (e.g., roads, buildings). Let V_s^t denote the set of static voxels at time t, and let F_s^t represent the corresponding 3D flow vectors.

According to our flow definition in Eq. (16), the estimated position of each static voxel in the next frame can be computed as:

$$\widetilde{V_s^{t+1}} = V_s^t + F_s^t \tag{18}$$

We aim to estimate a rigid-body transformation $(\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$ that aligns the original static voxel set with its predicted next-frame positions:

$$\widetilde{V_s^{t+1}} \approx \mathbf{R}V_s^t + \mathbf{t} \tag{19}$$

Similar to the centroid extraction in Eq. (4), we compute the centroids of the voxels. We then re-center them by subtracting the centroids, and we denote them as $\overline{\widetilde{V_s^{t+1}}}$, $\overline{V_s^t}$.

To estimate the optimal rigid transformation, we apply the Kabsch algorithm:

1. Compute cross-covariance matrix:

$$\mathbf{H} = \sum_{i=1}^{N} \overline{\widetilde{V_s^{t+1}}}\,\overline{V_s^t}^{\top}$$

2. Perform SVD: $\mathbf{H} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$
3. Recover rotation:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^{\top}$$

4. If $\det(\mathbf{R}) < 0$, negate the third column of $\mathbf{V}$ to ensure a proper rotation.
5. Compute translation:

$$\mathbf{t} = \widetilde{V_s^{t+1}} - \mathbf{R}V_s^t$$

7.3.2. Robust Estimation with RANSAC

To improve robustness against noise and outliers in the flow field, we adopt a RANSAC-based procedure for estimating the ego-motion transform. At each iteration, we perform the following steps:

1. Randomly sample 3 point-motion pairs $V_{s,3}^t$, $F_{s,3}^t$.
2. Estimate a rigid transform $(\mathbf{R}, \mathbf{t})$ like above.
3. Compute residuals:

$$\varepsilon_i = \left\| \mathbf{R}V_{s,3}^t + \mathbf{t} - \widetilde{V_{s,3}^{t+1}} \right\|_2$$

4. Count inliers with $\varepsilon_i < \tau$, where τ is a fixed threshold.
5. Retain the transform with the largest inlier count.

Once RANSAC converges, we re-estimate the final rigid transformation using all inliers to improve accuracy. In our experiments, we set the inlier threshold $\tau = 0.01$ and run for a maximum of 100 iterations, which provides a good trade-off between accuracy and efficiency.

7.3.3. Ego Motion via Inversion

The estimated transformation $(\mathbf{R}, \mathbf{t})$ maps voxel points from time t to $t+1$. Since the scene is assumed static, this transform is the inverse of the ego motion, which is therefore:

$$\mathbf{R}_{\text{ego}} = \mathbf{R}^{\top}, \quad \mathbf{t}_{\text{ego}} = -\mathbf{R}^{\top}\mathbf{t}$$

This yields the relative ego pose between frames t and $t+1$, expressed in the ego's initial frame. The steps above are illustrated in Figure 8.

7.3.4. Accumulating Camera Poses

To obtain the ego's trajectory, we recursively compose relative ego motion transforms:

$$\hat{\mathbf{T}}_0 = \mathbf{I}_{4\times4}, \quad \hat{\mathbf{T}}_{t+1} = \hat{\mathbf{T}}_t \cdot \mathbf{T}_{t+1}$$

This yields the ego pose $\hat{\mathbf{T}}_t \in \mathrm{SE}(3)$ in the coordinate frame of the initial timestep.

7.4. Occupancy Space Tracking, Alignment and Comparison Process

In Section 3.4.2, we discuss the process for occupancy tracking, alignment, and shape comparison. In Figure 9, we further illustrate it with an example. We begin with two consecutive model-forecasted occupancy grids (e.g., from OccWorld [55]) at $t = 0\,\text{s}$ and $t = 1\,\text{s}$, each containing voxel-wise semantic labels and flow vectors. In this example, we choose *car* as an example, so we isolate the voxels with semantic label *car*. Using the occupancy grid at $t = 0\,\text{s}$, we perform a single-step forward estimation by adding the forward flow vectors to the occupied voxels to obtain their predicted coordinates in the next frame.

Next, we compute the centroids of all segmented objects in both the estimated and forecasted grids. We apply bipartite Hungarian matching to associate corresponding object instances across frames, completing the tracking step. To evaluate shape consistency, we apply the alignment procedure described in Algorithm 1, which aligns the voxel sets by rotating the estimated objects to match the forecasted ones using principal axis alignment. Once aligned, we measure their voxel-wise similarity via the intersection-over-union (**IoU**) metric.

Importantly, this method of computing temporal consistency is modality-agnostic. For instance, in Figure 10, we illustrate the case of multi-modal forecasting with three possible futures. Temporal consistency can be independently computed for each future branch by aligning the prior

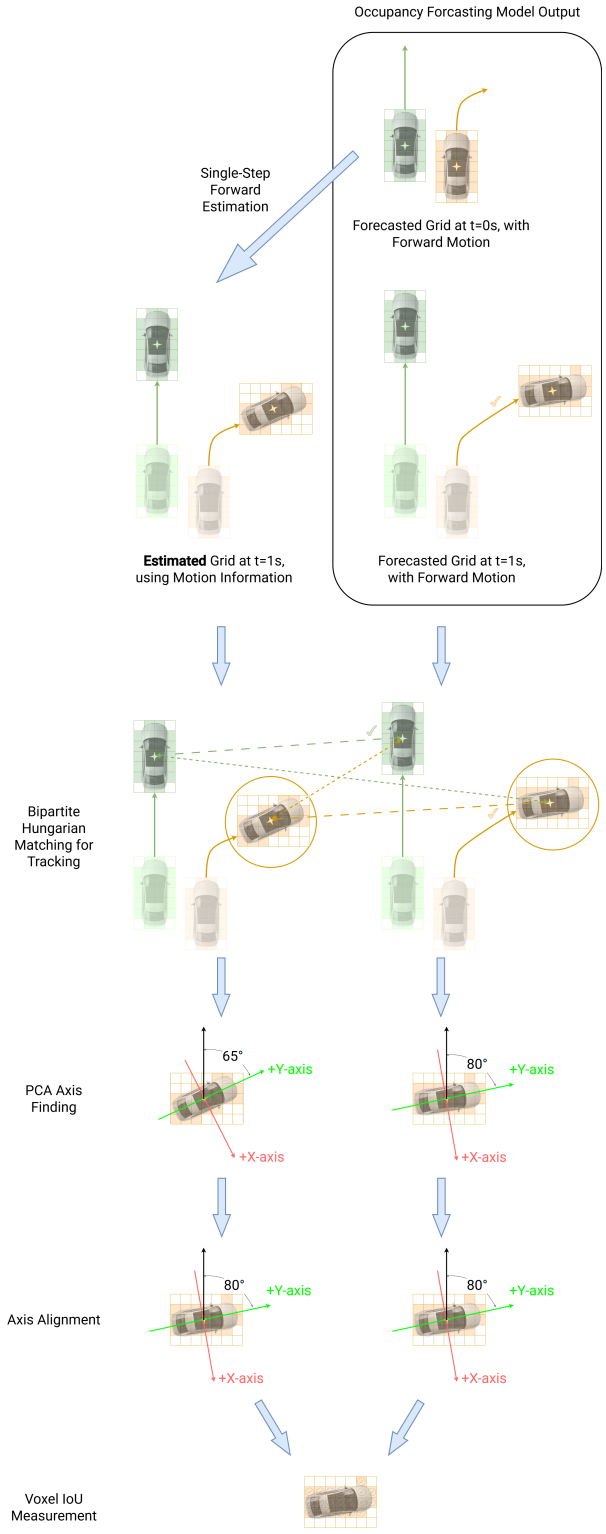

Figure 9. Flow Chart for Occupancy Space Tracking, Alignment and Comparison.

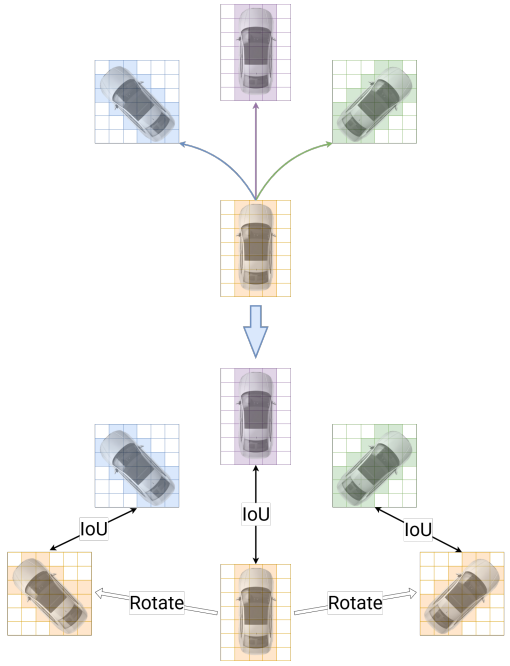

Figure 10. Example of how temporal consistency is measured in a multi-modal of three futures. *I.e.*, the ego is rotated to match each of the modalities.

frame's occupied voxels to the corresponding forecasted voxels before evaluating their overlap.

It is also important to note that the tracking procedure applies not just to the ego, but also to the other agents, allowing us to turn the occupancy forecasting problem into a motion prediction problem.

7.5. Voxel Alignment with PCA

In Algorithm 1, we include the pseudo code to align voxels through the temporal dimension by standardizing their rotations, which we introduced in Section 2.

8. Sim-Real Compatibility

The simulation data generated from CARLA differs from real-world driving data in several important aspects, including ego and agent speeds, traffic density, and vehicle geometry. Among these factors, we found that differences in agent speed between simulation and real-world datasets have the most significant impact on the temporal consistency of the trained models. Specifically, scenes in our CARLA dataset tend to feature narrower streets and slower-moving agents compared to those in nuScenes or Waymo. A comparative analysis of speed distributions is shown in Figure 7.

To quantify the impact of this discrepancy, we conducted additional experiments using a subset of CARLA scenes focused on highway driving, generated with the *Town06* map, which supports speeds up to $55\,\mathrm{mph}$ ($24.5\,\mathrm{m/s}$). As

Table 6. Occupancy forecasting performance of OccWorld [55] after mixing into the training data with either full-speed-range (Unbiased) CARLA scenes or highway-only CARLA scenes.

Train Sources	Test Source	$\text{mIoU}_{\text{geo}}\uparrow$				$\text{IoU}_{\text{geo}}\uparrow$				$\text{IoU}_{\text{bg}}\uparrow$	$\text{IoU}_{\text{car}}\uparrow$	$\text{P}_{\text{car}}\uparrow$
		0 s	1 s	2 s	3 s	0 s	1 s	2 s	3 s	Average over 0 s to 3 s		
nuScenes + Unbiased CARLA	nuScenes	71.47	31.70	22.69	18.11	62.94	35.83	28.29	21.03	55.07	77.87	82.97
nuScenes + Highway CARLA	nuScenes	72.63	31.89	22.21	21.85	63.70	36.24	28.68	21.15	61.40	85.59	82.89
Waymo + Highway CARLA	Highway CARLA	84.02	54.12	53.07	52.40	74.32	28.11	24.15	23.05	87.43	92.53	81.62
nuScenes + Waymo + Highway CARLA	Highway CARLA	87.50	67.15	61.86	59.75	74.85	28.57	25.20	23.61	88.14	98.41	82.54

Algorithm 1 PCA-Based Coordinate Transformation with Consistent Handedness

Input: V^t: A set of 3D points at timestep t, pca^{t-1} (if available)

Output: $\hat{V}^t$: Transformed coordinates with consistent orientation

Step 1: Fit the PCA to the current timestep.

$\text{pca}^t.\text{fit}(V^t)$

$\text{pca}_{\text{ref}} = \text{pca}^t$

Step 2: Transform coordinates into the principal component space.

$\hat{V}^t \leftarrow \text{pca}^t.\text{transform}(\bar{V}^t)$

for $t \in \{t_2, \ldots, t_n\}$ **do**

Step 3: Ensure consistent handedness using PCA from previous timestep.

(a) Compute diagonal sign vector.

$k^t = \text{pca}^t.\text{components}$

$k_{ref}^{t-1} = \text{pca}_{\text{ref}}^{t-1}.\text{components}^T$

$\boldsymbol{\sigma} \leftarrow \text{sign}\Big(\text{diag}\big(k^t \cdot k_{ref}^{t-1}\big)\Big)$

(b) Apply sign correction to maintain consistency.

$\hat{V}^t \leftarrow \hat{V}^t \odot \boldsymbol{\sigma}$; `// Hadamard product ensures consistent PCA orientation`

end

return $\hat{V}^t$

shown in Table 6, incorporating these high-speed scenes into training yields notable improvements: +11% in background consistency (IoU_{bg}) and +9% in object-level consistency (IoU_{car}). These results suggest that higher-speed simulated environments lead to more realistic motion dynamics and improved temporal coherence when used for joint training.

In our future work, we plan to further explore the role of sim-to-real compatibility and investigate principled ways to adapt or calibrate simulation data for improved generalization to real-world domains.

9. Unified Quality Score

Even though our proposed metrics do not depend on existing pseudo-labels, we retain the current widely used labels/metrics to keep comparability with prior works, while our novel metrics provide further insights into shape realism and consistency of voxels.

To facilitate direct comparison across methods and evaluation settings, we introduce a composite metric, **UniOcc Score**, which aggregates multiple specialized metrics via a weighted average:

$$\begin{aligned}\text{UniOccScore} = {} & \lambda_{\text{Recon}} \cdot \text{IoU}_{\text{geo}}^{t=0\text{s}} \\ & + \lambda_{\text{Forecast\_1s}} \cdot \text{IoU}_{\text{geo}}^{t=1\text{s}} \\ & + \lambda_{\text{Forecast\_2s}} \cdot \text{IoU}_{\text{geo}}^{t=2\text{s}} \\ & + \lambda_{\text{Forecast\_3s}} \cdot \text{IoU}_{\text{geo}}^{t=3\text{s}} \\ & + \lambda_{\text{Temp\_bg}} \cdot \text{IoU}_{\text{bg}} \\ & + \lambda_{\text{Temp\_car}} \cdot \text{IoU}_{\text{car}} \\ & + \lambda_{\text{Probs\_car}} \cdot \text{P}_{\text{car}}\end{aligned}$$

.

Here, each λ weight controls the contribution of a particular aspect of performance: reconstruction quality, forecasting accuracy at various horizons, temporal consistency for background and dynamic objects, and the realism probability for predicted cars.

While the weights may be adjusted depending on the downstream application (*e.g.*, emphasizing forecasting metrics in long-horizon prediction tasks), we provide a reference configuration that balances reconstruction and forecasting:

Weight Component	Value
λ_{Recon}	0.20
$\lambda_{\text{Forecast\_1s}}$	0.15
$\lambda_{\text{Forecast\_2s}}$	0.10
$\lambda_{\text{Forecast\_3s}}$	0.05
$\lambda_{\text{Temp\_bg}}$	0.30
$\lambda_{\text{Temp\_car}}$	0.20
$\lambda_{\text{Probs\_car}}$	0.10

The resulting UniOcc scores from our experiments are reported in Table 9.

10. Additional Experiments

In Table 8, we show the results of using voxel-level flow on OccWorld [55] on the Waymo dataset. The performance gain is consistent with the nuScenes dataset in Table 2.

Table 7. Distribution of ego vehicle speeds in CARLA and nuScenes datasets. The standard CARLA scenes exhibit predominantly low-speed driving behavior, whereas nuScenes features a broader speed distribution. Our highway-focused CARLA subset (generated using the *Town06* map) better matches the real-world speed range observed in nuScenes, improving sim-real compatibility for training.

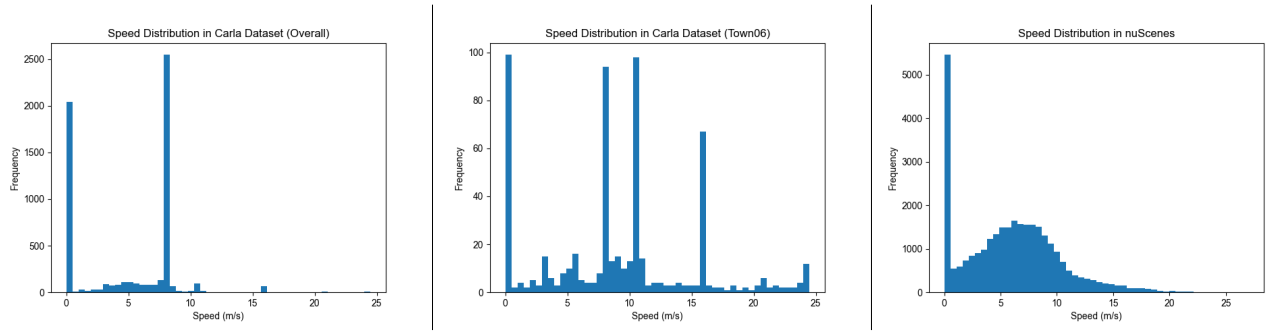

Table 8. Occupancy forecasting performance of OccWorld [55] on Waymo datasets with different types of flow.

Train and Test Source	**Flow Type**	**mIoU$_{geo}$↑**				**IoU$_{geo}$↑**				**IoU$_{bg}$↑**	**IoU$_{car}$↑**	**P$_{car}$↑**
		0 s	1 s	2 s	3 s	0 s	1 s	2 s	3 s	Average over 0 s to 3 s		
Waymo	None	68.24	30.40	24.03	21.79	70.89	34.41	28.85	26.33	56.06	88.10	83.55
Waymo	Object	67.66	30.65	24.41	21.93	71.26	34.03	29.18	26.60	55.56	88.11	84.30
Waymo	Voxel	**71.35**	**32.04**	**25.77**	**23.76**	**72.69**	**36.04**	**30.48**	**27.96**	**58.26**	**89.30**	**86.68**

Table 9. Reference UniOcc Score.

Train Sources	**Test Source**	**UniOcc Score**
nuScenes	nuScenes	63.99
Waymo	nuScenes	57.26
CARLA	nuScenes	40.10
nuScenes	Waymo	65.36
Waymo	Waymo	68.40
CARLA	Waymo	41.70
nuScenes	CARLA	71.48
Waymo	CARLA	71.05
CARLA	CARLA	46.39
nuScenes + Waymo	nuScenes	65.53
nuScenes + Waymo	Waymo	68.86
nuScenes + CARLA	nuScenes	62.24
Waymo + CARLA	nuScenes	58.61
nuScenes + CARLA	Waymo	66.13
Waymo + CARLA	Waymo	68.45
nuScenes + CARLA	CARLA	73.53
Waymo + CARLA	CARLA	71.00
nuScenes + Waymo + CARLA	nuScenes	63.64
nuScenes + Waymo + CARLA	Waymo	70.11
nuScenes + Waymo + CARLA	CARLA	70.58

11. Additional Dataset Details

In Table 10, we include a list of the different labels used in our different data sources.

Table 10. Label correspondence across four different sources (nuScenes, Waymo, CARLA, and UniOcc (Ours)). Empty cells indicate no direct counterpart.

ID	**nuScenes**	**Waymo**	**Carla**	**UniOcc (Ours)**
0	general_object	general_object	free	general_object
1	barrier	vehicle	buildings	vehicle
2	bicycle	pedestrian	fences	bicycle
3	bus	sign	other	motorcycle
4	car	cyclist	pedestrians	pedestrian
5	construction_vehicle	traffic_light	poles	traffic_cone
6	motorcycle	pole	roadlines	vegetation
7	pedestrian	construction_cone	roads	road
8	traffic_cone	bicycle	sidewalks	walkable/terrain
9	trailer	motorcycle	vegetation	building
10	truck	building	vehicles	free
11	drivable_surface	vegetation	walls	-
12	other_flat	tree_trunk	trafficSigns	-
13	sidewalk	road	sky	-
14	terrain	walkable	ground	-
15	manmade	-	-	-
16	vegetation	-	-	-
17	free	-	-	-
23	-	free	-	-