

OccFlowNet: Supplementary Material

A. Qualitative Results

A.1. Predicted Occupancy

In Fig. A.1 and Fig. A.2, we show examples of predicted occupancy of our OccFlowNet model compared to ground truth occupancy. Figure A.1 shows a comparison between estimated and ground-truth occupancy from a top-down perspective, while Fig. A.2 shows the voxel grids from a third-person view. The figures depict results on the Occ3D-nuScenes validation set, and the model used is OccFlowNet with occupancy flow, but without voxel labels. As observable, the model can accurately estimate dynamic and static objects around the vehicle, in day scenes as well as in night scenes. In Fig. A.1 one can also recognize that the predicted occupancy always appears stretched out from the ego-vehicle into the depth direction, which results from the rendering supervision approach. On the other hand, the model is often more dense than the ground truth, which is based on LiDAR scans. Our model can accurately fill out areas that are visible in the images but were not scanned by the LiDAR. Additionally, we provide a set of videos with the supplementary material that show inference results of our model on some Occ3D-nuScenes validation scenes. Videos *scene-0558.mp4* and *scene-0916.mp4* show predicted occupancy from the perspective of the input cameras, from third-person and from a Birds-Eye-View perspective. The videos *scene-0038_gt.mp4* and *scene-0107_gt.mp4* compare model predictions against the ground truth occupancy, both from a third-person perspective.

A.2. Rendered Depth and Semantics

Additional qualitative results are provided in Fig. A.3. The figure shows what the model "sees" when the 3D voxel predictions are rendered into the 2D space using differentiable volume rendering. The LiDAR ground truth represent the labels that the model receives during training, while the rendered depth and semantics are the rendered occupancy estimations of a trained OccFlowNet model. As clearly visible, the model has learned to estimate depth and semantics correctly, and that volume rendering can be effectively used to create 2D predictions, while being fully differentiable.

B. Details: Dynamic Ray Filter

Figure B.4 demonstrates how the dynamic ray filtering is applied. During training, while loading the data, we filter out dynamic objects in adjacent time steps. LiDAR points associated with dynamic object classes are removed, such as the *car* within the red box or the *construction vehicle* within the green box depicted in the figure. In the current time step, dynamic LiDAR points are kept, as the dynamic objects are at the correct position. Then, a ray is generated for each remaining LiDAR point on each input image. It is important to note that this technique alone is insufficient, as moving objects may expose background points that are not filtered (disocclusions), introducing misleading supervision.

C. Details: Occupancy Flow

Figure C.5 shows a flowchart of the occupancy flow mechanism proposed in the paper. As a precomputation step, for each time step in the dataset, we compute the movement of each voxel to each time step in the horizon (e.g., to the previous 3 and next 3 time steps). We do this by calculating the transformation between the corresponding boxes in the current and the temporal frames, and assign this transformation to all voxels inside the bounding box. During training, after the forward pass (and before employing the rendering), we can now load the transformations for all voxels of the current frame. We then clone the estimated occupancy once for each temporal time step and apply the corresponding flow. We now have one estimated occupancy field for each temporal time step, but compensated for object motion. Volume rendering is now applied as described in the paper, but each camera renders the occupancy field of its time step. Figure C.6 illustrates how the occupancy flow is applied using the example of a single instance of the *car* class. Given the current time step t and a target time step $t + 1$, we utilize the precomputed transformation from the current bounding box to the target bounding box to move the estimated occupancy to the target time step. This procedure is repeated for all time steps within the specified horizon and for each dynamic object with a bounding box. Note that all voxels that have been estimated to belong to a static category are not moved. Using this technique, the dynamic ray filter becomes unnecessary, as dynamic objects are now correctly positioned even in temporal time steps.

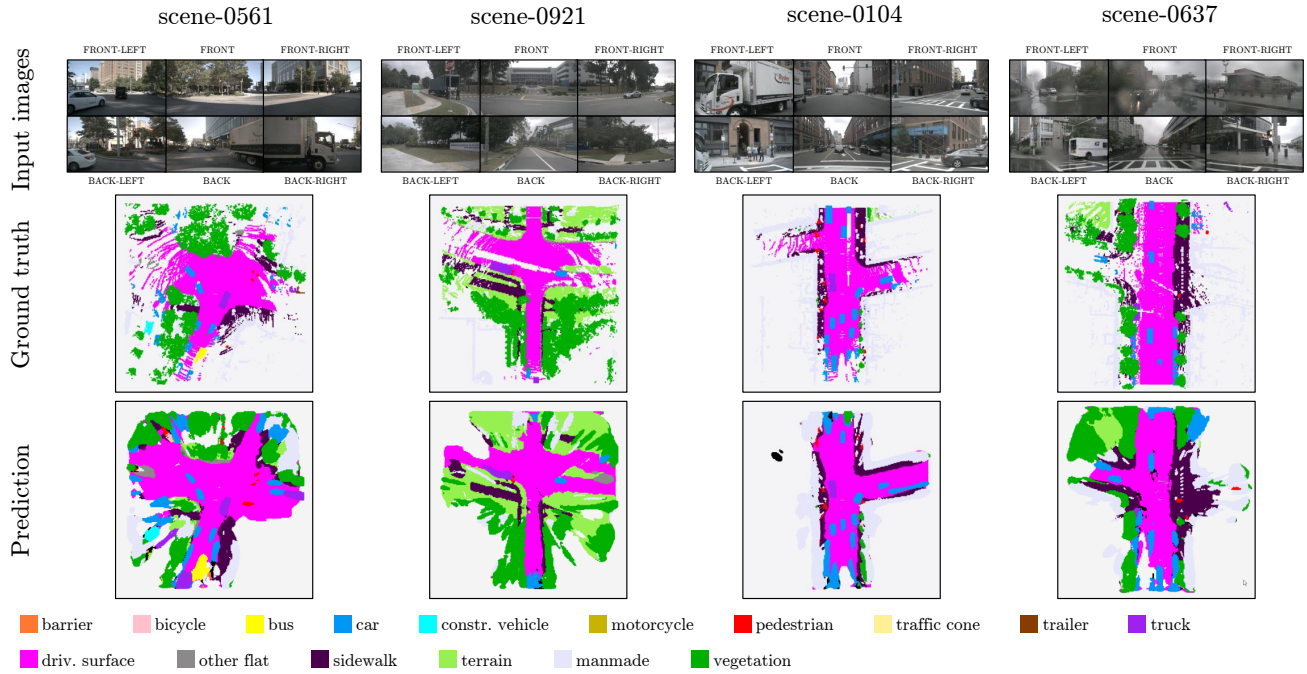


Figure A.1. **Qualitative results on the Occ3D-nuScenes dataset, viewed from the top.** The proposed model (Ours Flow 2D) can estimate the environment in 3D correctly and generalizes well to unseen areas.

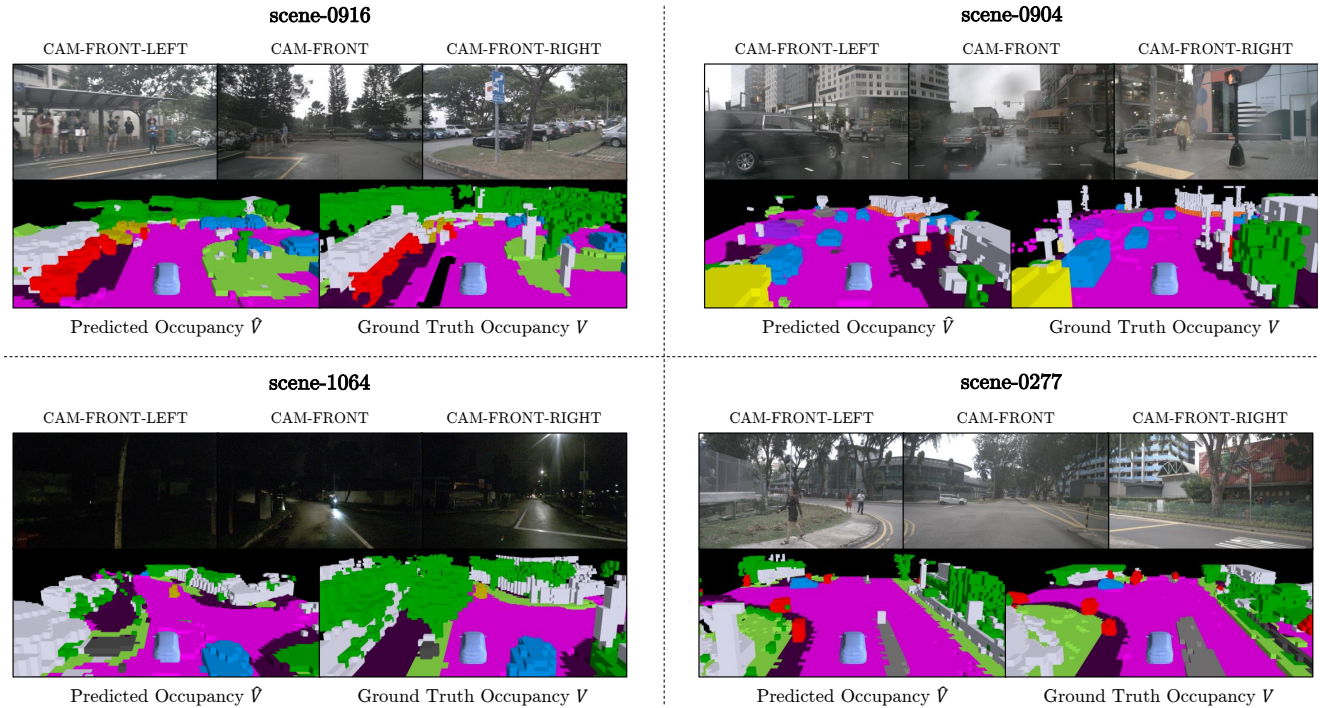


Figure A.2. **Qualitative results on the Occ3D-nuScenes validation set.** Images of the occupancy space are taken from above and behind the ego vehicle ("third-person" view). The model can estimate the semantic occupancy well compared to the ground truth.

Significantly more rays can be generated for dynamic ob-

jects, greatly increasing their detection performance.

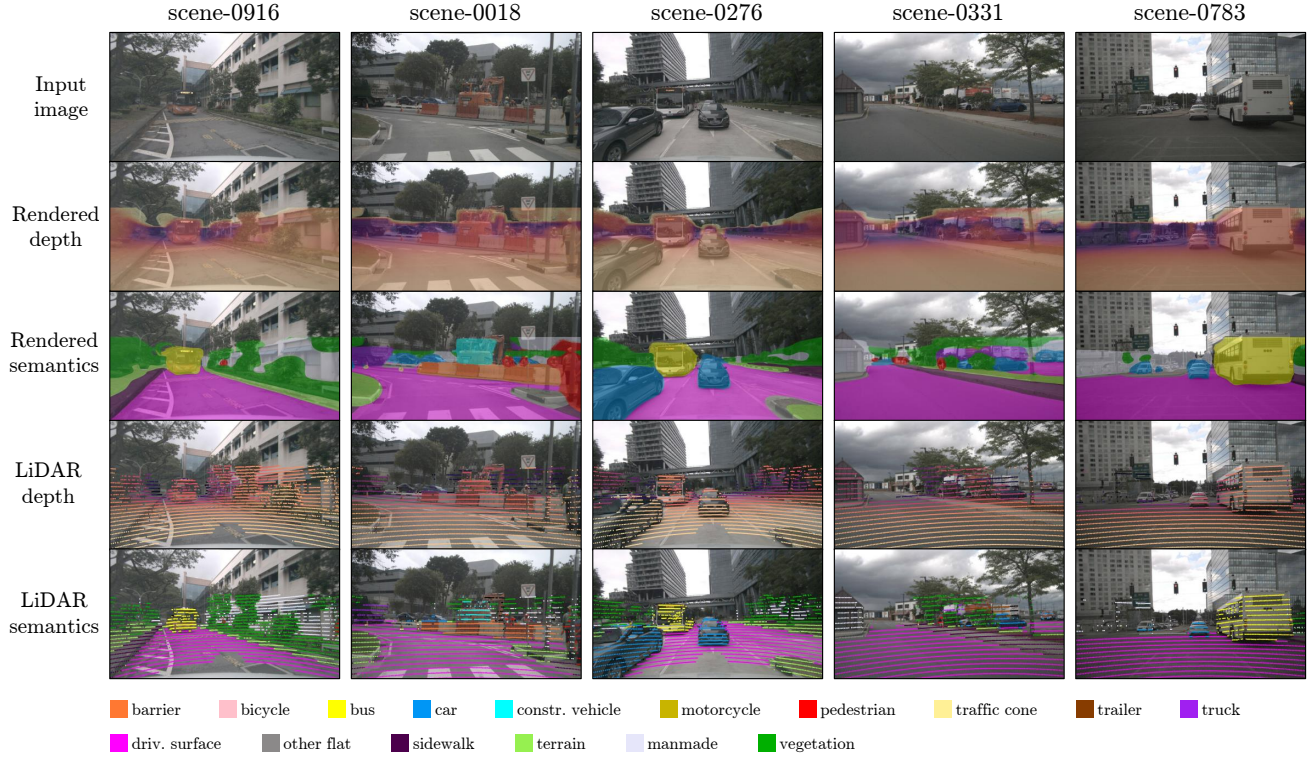


Figure A.3. **Qualitative results on the Occ3D-nuScenes validation set.** Each column shows an *input image*, and the corresponding *rendered depth* \hat{D} and *rendered semantics* \hat{S} when rendering the occupancy predictions using volume rendering, simulating the training process. Below, the 2D training labels are shown, generated by projecting annotated LiDAR point scans onto the input images.

D. Source Code

We provide source code to reproduce the results presented in the paper at <https://github.com/boschresearch/OccFlowNet>. Please follow the instructions in the *README.md* file to install the repository and run the code. We provide instructions to train and evaluate models in all of our configurations on the Occ3D-nuScenes dataset.

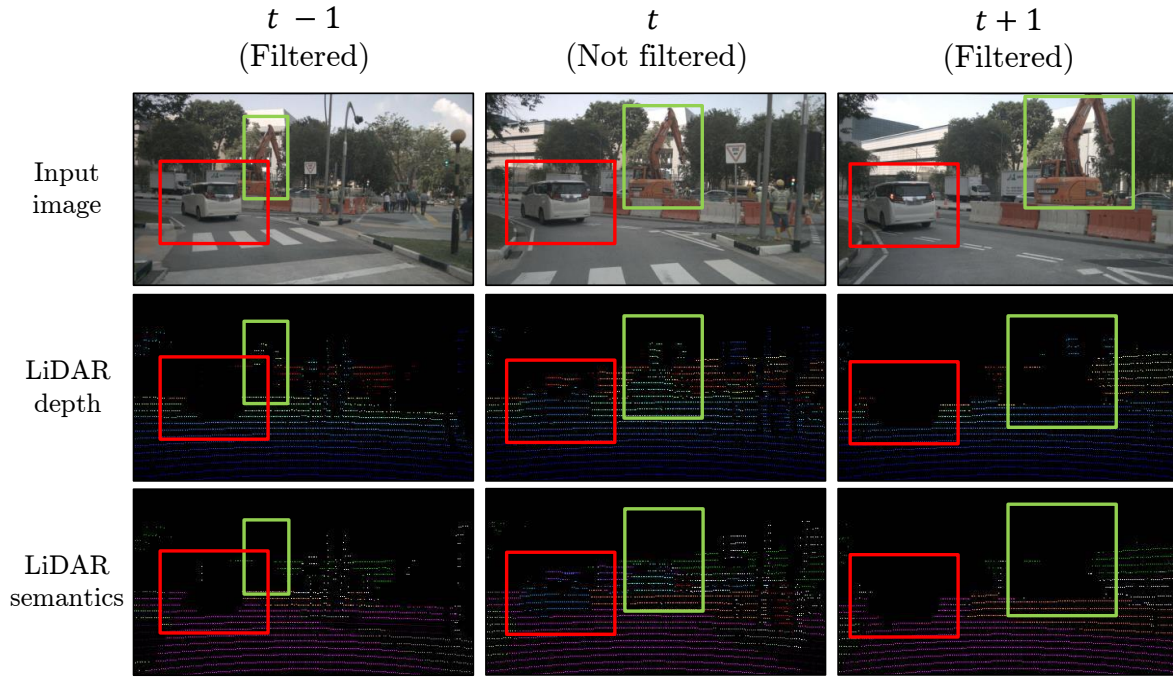


Figure B.4. **Illustration of the dynamic ray filter.** Shown are the *input image*, *LiDAR depth* and *LiDAR semantics* for a single camera across three consecutive time steps when using the dynamic ray filtering. LiDAR points corresponding to dynamic objects are removed in adjacent time steps $t - 1$ and $t + 1$. In the current time step t , points of dynamic objects are preserved.

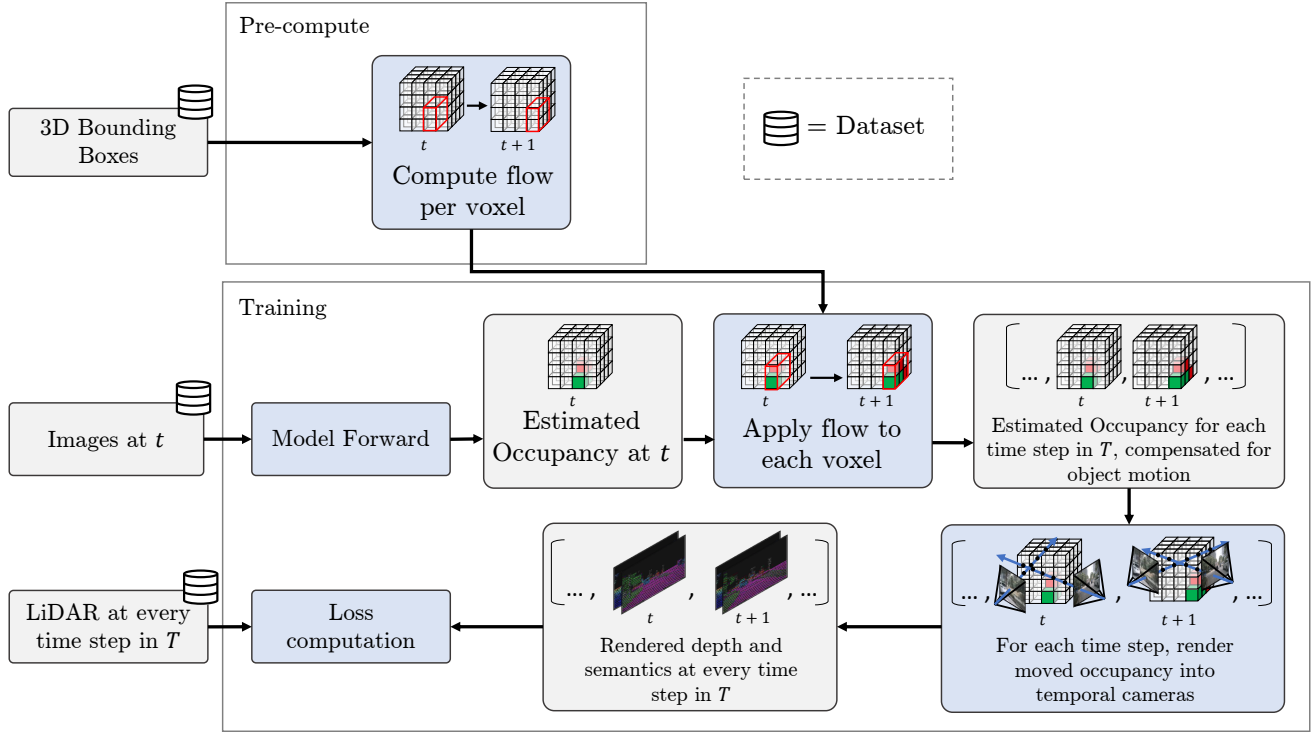


Figure C.5. **Flowchart illustrating how occupancy flow is applied.** Prior to training, the flow of each voxel to all possible time steps is computed. During training, the occupancy estimated by the model is duplicated once for each temporal time step, and the corresponding flow is applied to each voxel. Then, each temporal camera uses the motion-compensated version of the occupancy to render the estimated depth and semantic maps. Finally, LiDAR scans are projected on images to create 2D ground truth labels for loss computation.

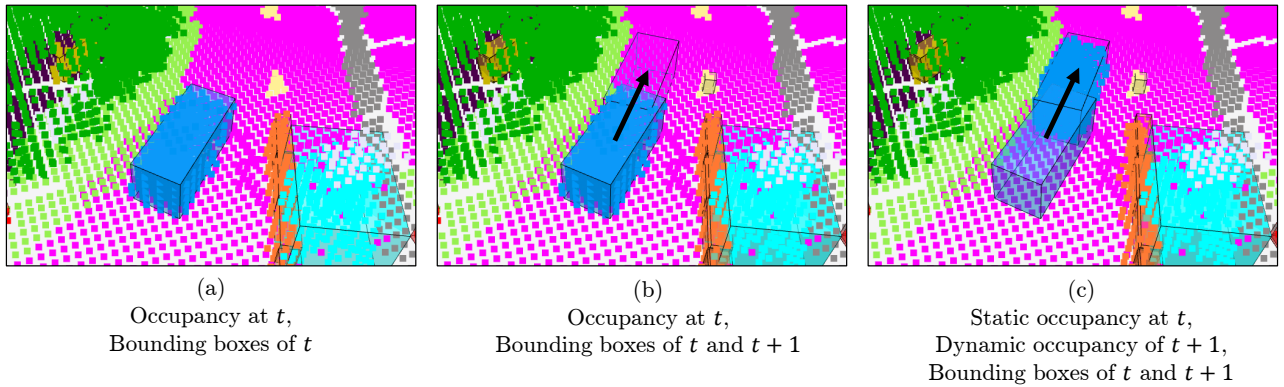


Figure C.6. **Example of the occupancy flow.** We precompute the transformations between corresponding boxes in adjacent time steps, and use them to relocate the estimated occupancy \hat{V} to the target time steps.