# Revealing Occlusions with 4D Neural Fields

## Supplementary Material

## A. Implementation Details

### A.1. Architecture and Hyperparameters

We illustrate the full model architecture in Figure 10. The input point cloud video $\mathcal{X}$ is subsampled using a mixture of random point sampling and farthest point sampling to contain precisely 14,336 points with 7 to 8 features each. After the first MLP, the embedding size per point is 36, which becomes double at every down transition step, such that $\mathcal{Z}$ has 288 features per embedding. The encoder $\phi$ also averages all features into a 128-dimensional global embedding, which is also passed to $f$ alongside the featurized point cloud $\mathcal{Z}$. The latent size of embeddings within $f$ is 416 across all its components.

The self-attention block causes every point to attend to its 16 nearest neighbors [69]. The down transition block selects one third of all incoming points by means of farthest point sampling, and subsequently performs channel-wise max-pooling from its 12 nearest neighbors. The residual MLP block in $f$ initially distills a weighted linear interpolation (based on Euclidean distance) of the 8 nearest neighbors in $\mathcal{Z}$ around the query point as a starting point, after which cross-attention has the chance to apply a *learned* interpolation of embeddings instead. The cross-attention block causes every query point to attend to its 14 nearest neighbors in the featurized input points $\mathcal{Z}$, *i.e.* $|\mathcal{M}(i)| = 14, \forall\, i$.

Using the AdamW optimizer [31], we train two separate models (one per dataset) over 20 epochs for GREATER, and 40 epochs for CARLA. Our model takes between 18 and 55 hours to train on two RTX A6000 GPUs, and dense inference across the entire spacetime cube takes roughly one minute. The initial learning rate is $0.001$, but this drops by a factor $2.5$ at progress rates of $40\%$, $60\%$, and $80\%$.

### A.2. Point Sampling

During training, within every frame we sample 7,168 solid query points ($\sigma = 1$) and 10,752 free space (air) query points ($\sigma = 0$). The air points are uniformly randomly sampled within the output box of interest, except if they are within a distance of $2\epsilon = 0.2$ within any target point in $\mathcal{Y}$. The solid points are selected as a random subset of the target point cloud, but we add a small random spatial offset to every solid query point that itself is uniformly sampled within a spherical ball of radius $\epsilon = 0.1$. This roughly corresponds to the spacing between target points on the objects and the floor in the dataset, encouraging the model to learn to "fill in the gap" between those points.

In the case of CARLA, to ensure that we maintain an effective learning signal, we describe several tricks that help guide supervision toward areas where it is deemed more important. In addition to random sampling from the target point cloud, 14% of solid points are explicitly sampled in dynamic regions of the scene, *i.e.* moving points that did not exist in a randomly selected other frame. The converse is also done for air points (*i.e.* regions that are now missing, but were present in another frame). At least 7% of sampled solid points focus exclusively on vehicles and pedestrians. We also oversample occluded vehicles and pedestrians (for up to 7% of all sampled solid points), by counting and comparing the number of points of every object seen by every view. Moreover, we ensure that objects that were never seen in the first place (*e.g.* pedestrians who remain behind a building or wall throughout the entire video) are not oversampled, to avoid confusing the model. Lastly, for class balancing, 14% of sampled solid points treat all semantic categories equally, *i.e.* we sample the same number of points from every class that is present in the scene.

For a fair and correct evaluation, there are no sampling tricks at test time, *i.e.* we apply uniform sampling within the cuboid of interest in a way that is agnostic of the ground truth. For GREATER, we sample $N = 2^{19}$ points per frame per scene, while for CARLA, $N = 2^{21}$.

### A.3. Evaluation Metrics

The model is evaluated by the Chamfer Distance (CD) between the sampled prediction and the ground truth point cloud. Sometimes, the model fails to predict an object when it is fully occluded (*i.e.* a false negative), which may cause the output point cloud (filtered by the desired category and occlusion rate) to consist of zero points. Rare or "tiny" classes with a low number of target points per scene, *e.g.* traffic sign, may face similar issues. In that case, the CD would normally be undefined, but in order to ensure that the average metric accounts for this and is still affected, we substitute the prediction with a single point in the center of the scene: $(0, 0, 0)$ for GREATER, or $(20, 0, 0)$ for CARLA.

## B. Dataset Description

Both GREATER and CARLA are posed multiview RGB-D video datasets, with added instance segmentation and semantic segmentation annotations respectively. The camera views are illustrated in Figures 11 and 12.

### B.1. GREATER

All videos are recorded with a virtual RGB-D camera with known intrinsics and extrinsics. The dataset is gen-

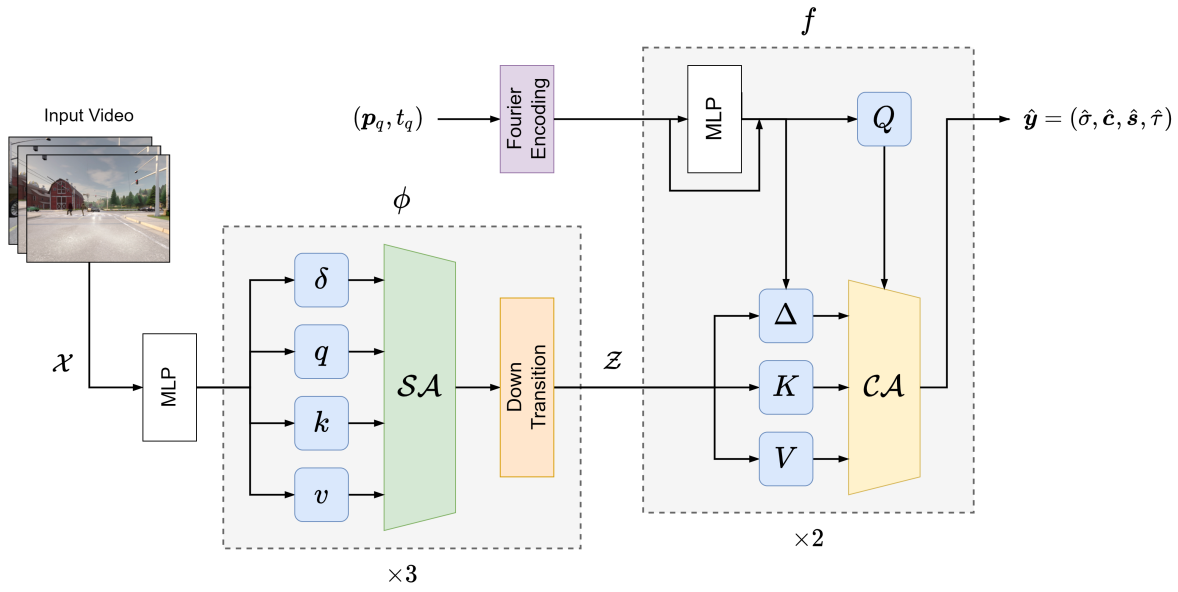


Figure 10. **Detailed Architecture –** We show the feature encoder $\phi$ and implicit representation $f$, with self-attention $\mathcal{SA}$ and cross-attention $\mathcal{CA}$ operations respectively. $\phi$ is a point transformer with four self-attention layers in total (the one after the last down transition is not shown), although the outputs of the last two down self-attention blocks are combined to form multi-scale features for $\mathcal{Z}$. The exact operation of the down transition modules is described in [69]. For the query points $(\boldsymbol{p}_q, t_q)$, we adopt the same Fourier encoding as [34]. All MLP blocks consist of two linear layers with a ReLU non-linearity in-between. The MLP blocks in the implicit representation $f$ are residual, similar to [67].

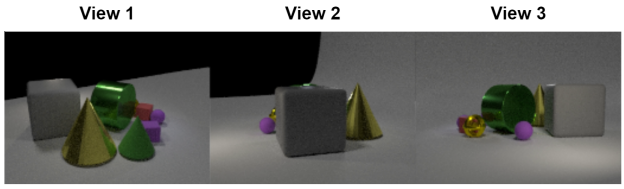


Figure 11. **GREATER Dataset Views –** Every GREATER scene has three non-moving RGB-D cameras at uniformly random azimuth angles, with the condition that all three are spaced by at least 45° away from each other. A random view is always selected to serve as input view, such that the other two views (along with the input view itself) serve as supervision during training.

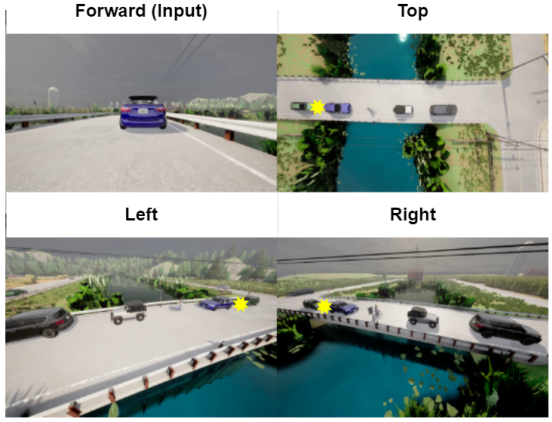


Figure 12. **CARLA Dataset Views –** Every CARLA scene has one RGB and LiDAR sensor attached to the front of the ego vehicle, which always records the input video. Three other supervisory views (in which the forward sensor position is marked with a yellow asterisk) operate only at training time.

erated at 24 frames per second (FPS), and objects move and/or rotate in synchronized cycles that repeat every 32 to 42 frames. The model's data loader subsamples temporally and uses 8 FPS, such that along with $T = 12$, roughly one full cycle is covered per clip. All objects are precisely twice as large as compared to CATER [18]. For every scene, the number of objects is selected uniformly at random between 8 and 12 (inclusive). The cameras are also chosen randomly per scene, but henceforth remain static (*i.e.* never move over time) over the duration of a single scene. With the spatially vertical axis denoted $z$, the 3D bounding box within which both the input and predictions happen is $x \in [-5, 5]$, $y \in [-5, 5]$, $z \in [-1, 5]$.

### B.2. CARLA

All videos are recorded with a pair of sensors with known intrinsics and extrinsics: one RGB camera, and one LiDAR sensor. The point cloud data generated by the latter sensor does not contain color information, so we use the RGB images to colorize the points. While the dataset is generated at 10 FPS, the model's data loader subsamples temporally and uses 5 FPS. Both the LiDAR and camera horizontal fields of view are 120 degrees. However, the LiDAR's spherical geometry is different from a camera's projective geometry. implying that the LiDAR data is not directly aligned with the camera intrinsics. Therefore, in order to obtain a colorized point cloud per frame, we first project all LiDAR points onto the image and then map the pixel's RGB values it was assigned to back to the 3D point. If a LiDAR point falls outside of the camera's field of view, we mark it with a generic "color unavailable" constant, *i.e.* $(-1, -1, -1)$.

For every input clip passed to the model, since the vehicle pose over time is known, we correct all point clouds to a common reference frame. This reference frame is chosen to be the last input (and output) frame, such that the pair of sensors mounted to the ego vehicle is always at $(0, 0, 1)$ at time $t = T - 1$. With the forward axis denoted $x$, the sideways axis denoted $y$, and the vertical axis $z$, the input bounding box (*i.e.* containing all observed points) is $x \in [-14, 50]$, $y \in [-20, 20]$, $z \in [-1, 10]$, and the output bounding box (*i.e.* containing all predicted and ground truth points) is $x \in [0, 40]$, $y \in [-16, 16]$, $z \in [-1, 6.4]$ (in meters).

### B.3. Clip Sampling

During training, for GREATER, we sample clips uniformly randomly. For CARLA however, most clips are relatively uninteresting, and we encourage learning about occlusions by performing biased clip sampling. Specifically, we construct a subset of starting frame indices where we know (as derived from semantic information over time in the LiDAR point clouds provided by the simulator) that occlusions are more likely to happen, and return a clip from this pool 40% of the time. To preemptively avoid overfitting, the data loader will never return the exact same clip twice over the entire duration of training.

During testing and evaluation, we deterministically sample a single clip within every video that has the most occlusions happening at once, counted over the number of objects. This implies that the test set for CARLA is significantly more challenging than the average driving situation (*i.e.* as compared to if we were to sample clips uniformly at random).

## C. Qualitative Results

Please see our webpage at `occlusions.cs.columbia.edu` for more visualizations, as well as links to our datasets, source code, and models.