Revealing Occlusions with 4D Neural Fields

Basile Van Hoorick
Columbia University
basile@cs.columbia.edu

Purva Tendulkar
Columbia University
purva.tendulkar@columbia.edu

Dídac Surís
Columbia University
didac.suris@columbia.edu

Dennis Park
Toyota Research Institute
dennis.park@tri.global

Simon Stent
Toyota Research Institute
simon.stent@tri.global

Carl Vondrick
Columbia University
vondrick@cs.columbia.edu

Abstract

For computer vision systems to operate in dynamic situations, they need to be able to represent and reason about object permanence. We introduce a framework for learning to estimate 4D visual representations from monocular RGB-D video, which is able to persist objects, even once they become obstructed by occlusions. Unlike traditional video representations, we encode point clouds into a continuous representation, which permits the model to attend across the spatiotemporal context to resolve occlusions. On two large video datasets that we release along with this paper, our experiments show that the representation is able to successfully reveal occlusions for several tasks, without any architectural changes. Visualizations show that the attention mechanism automatically learns to follow occluded objects. Since our approach can be trained end-to-end and is easily adaptable, we believe it will be useful for handling occlusions in many video understanding tasks. Data, code, and models are available at `occlusions.cs.columbia.edu`.

1. Introduction

When an object becomes occluded in video, its location and visual structure is often still predictable. In several studies, developmental psychologists have been able to demonstrate that shortly after birth, children learn how objects persist during occlusions [2, 5, 42, 52], and evidence suggests that animals perform similar reasoning too [32, 41].[1] For example, although the yellow orb in Figure 1 disappears behind other objects, its location, geometry, and appearance remain evident to you. Occlusions are fundamental to computer vision, and predicting the contents behind them underlies many applications in video analysis.

The field has developed a number of deep learning methods to operate on point clouds [19, 44, 66, 68, 69], which due to their attractive properties, have emerged as the representation of choice for numerous 3D tasks. Point clouds are sparse, making them particularly scalable to large scenes. However, to solve the problem in Figure 1, we need a video representation that (1) uses evidence from the previous frames in order to (2) generate the new points that are not observed in the subsequent frames. Since point clouds are possible to collect at scale [7, 9], we believe they are an excellent source of data for learning to predict behind occlusions in video. However, the representation must also have the capacity to *create* points conditioned on their context.

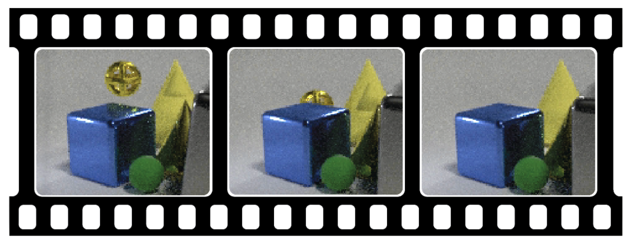

Figure 1. **Video Occlusions –** Although the yellow orb becomes fully occluded in the video, we can still perform many visual recognition tasks, such as predicting its location, reconstructing its appearance, and classifying its semantic category. This paper introduces a video representation architecture that is able to learn to perform all of these occlusion reasoning tasks.

This paper introduces an architecture for learning to predict 4D point clouds from an RGB-D video camera. The key to our approach is a continuous neural field representation of a point cloud, which uses an attention mechanism to condition the full space on the observations. Since the representation is continuous, the approach can learn to produce points anywhere in spacetime, allowing for high-fidelity reconstructions of complex scenes. Where there are occlusions and missing observations, the representation is able to use attention to find the object and/or missing scene structure when it was last visible, and subsequently put the right

[1] See "What The Fluff Challenge" on YouTube.

points in the right place.

Experiments show that our video representation learns to successfully perform many occlusion reasoning tasks, such as visual reconstruction, geometry estimation, tracking, and semantic segmentation. The same method works for these tasks without architectural changes. On two different datasets, we show the approach remains robust for both highly cluttered scenes and objects of various sizes. Though we train the representation without ground truth correspondence, visualizations show that the attention mechanism automatically learns to follow objects through occlusions.

There are three principal contributions in this paper. Firstly, we propose the new fundamental task of **4D dynamic scene completion**, which forms a basis for spatiotemporal reasoning tasks. Secondly, we present new benchmarks to evaluate scene completion and object permanence in cluttered situations. Thirdly, we introduce a new architecture for deep learning on point clouds, which is able to generate new points conditioned on their context. This architecture allows for large-scale point cloud data to be leveraged for representation learning. In the remainder of the paper, we describe these contributions in detail. We invite the community to use these benchmarks to test their model's video understanding capabilities.

2. Related Work

Learning to persist objects through occlusions has been a long-standing challenge in computer vision [20, 36, 37]. In recent years, researchers have combined modern deep learned features with a variety of approaches to track through occlusions. These include classical Kalman filtering or linear extrapolation [25], 2D recurrent neural networks [57], and more explicit reasoning mechanisms [49]. Our approach tackles the problem in a more holistic manner, drawing on improvements in point cloud modeling, neural fields, and attention mechanisms. We briefly recap relevant work from each area.

Point cloud modeling. Earlier work on representing point clouds with deep networks is based on 2D projection [11, 24, 54] or 3D voxelization [33, 51]. These methods capitalize on the success of 2D and 3D convolutions in image and video understanding by preprocessing input point clouds into 2D or 3D grids. PointNet [44] proposed to use point-wise MLPs and pooling layers to compute permutation-invariant point cloud representations. PointNet was subsequently extended to allow for hierarchical features to better model local geometric structures [45], and combined with idea of voxelization to create a highly efficient point cloud encoder [26]. More recently, researchers have begun to apply transformer attention mechanisms that were first found to be valuable in the language domain [58] to encode point clouds [27, 28, 64]. To address the quadratic complexity in attention computation applied to large-scale point clouds, the Point Transformer [69] replaced global attention with local vector attention and introduced relative position encoding. We adopt the Point Transformer as our feature encoder backbone because of its efficiency and performance for various point cloud tasks.

Point cloud tasks. Our goal is somewhat similar to that of point completion networks [21, 55, 60, 62, 68], although these works typically operate on a per-object basis and address only *self*-occlusions or amodal completion. In contrast, we aim to reconstruct entire scenes and address the fundamental challenge of occlusions more generally. Because existing 4D architectures [13, 29, 56] lack a mechanism to efficiently create new points, they have not been demonstrated to be capable of dynamic scene completion. For example, in 4D panoptic LiDAR segmentation [3], the goal is to jointly tackle semantic and instance segmentation in 3D space over time. While our work addresses related tasks, we wish to be able to model not just the visible, but also the *occluded* parts of the scene, by drawing on past observations or priors. This is especially valuable when spatial inputs are sparse, as they often are in LiDAR applications.

Neural fields. Neural implicit functions have become very popular for 3D representation in recent years [12, 48, 50, 67], building on the seminal work of Neural Radiance Fields (NeRF) [34] and neural implicit surface modeling [39]. The basic idea of NeRF is to learn to represent a scene using a fully connected deep neural network, whose inputs are a 3D point and viewing direction and whose outputs are an estimated color and volume density. This is attractive because it avoids the need to discretize space and can encode a scene more efficiently and richly than traditional representations such as meshes or voxels, which themselves can be extracted from the implicit model. Numerous efforts have been made to extend NeRF to dynamic scenes [16, 17, 40, 43, 63], but in addition to requiring per-scene retraining, occlusions are typically explicitly ignored by applying losses over the non-occluded scene only.

Transformers in vision. The attention mechanism introduced in [4, 58] has been applied with great success to computer vision [10, 14, 53, 65]. Recently, architectures that are built solely with self-attention as computational units have started to perform on par with or better than convolutional networks as generic feature extractors [38] in standard vision tasks such as object detection and segmentation [30, 46, 47] and point cloud–based detection [35]. The role of cross-attention has also been extended as a mechanism for sensor fusion. DETR3D [61] extends DETR [8] by computing the keys and values from multi-view images. Recently, Perceiver [22] showed that asymmetric attention mechanisms can distill inputs from multiple modalities (*i.e.* vision or point clouds) into robust latent representations.

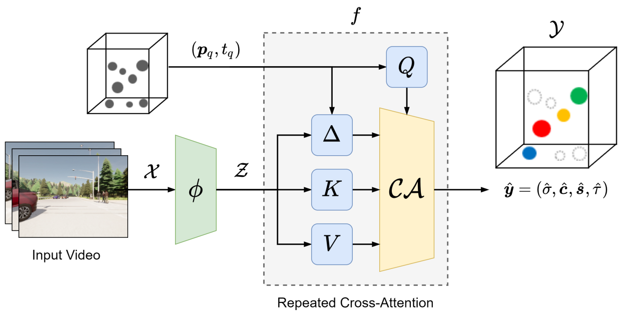

Figure 2. **Neural Architecture –** The encoder ϕ is a point transformer that featurizes the input point cloud video $\mathcal{X}$ using self-attention to produce $\mathcal{Z}$. The implicit representation f, conditioned on $\mathcal{Z}$, incorporates cross-attention blocks to contextualize the query points $(\boldsymbol{p}_q, t_q)$ and create the desired output features $\hat{\boldsymbol{y}}$ for that location and time within the scene.

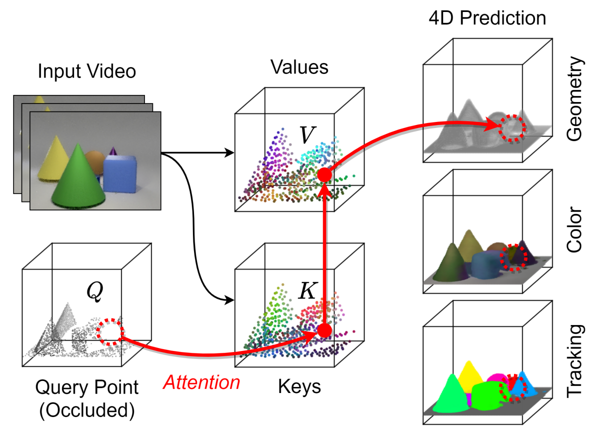

Figure 3. **Key Idea –** Given a query point in 4D, the model learns to attend to keys and values extracted from the input video. When the query point corresponds to world coordinates that are occluded, the attention mechanism will learn to find the object when it was not yet occluded. When the world coordinates correspond to empty space, the model instead learns to predict a low occupancy $\hat{y}_\sigma \approx 0$. The network is flexible, and we can train the same model to produce point clouds for many different video tasks requiring object permanence.

3. Approach

We introduce the new task of 4D dynamic scene completion from posed monocular RGB-D video input. Let $\mathcal{X} = \{ (\boldsymbol{p}_i, t_i, \boldsymbol{x}_i) \}$ be a point cloud video captured from a single camera view.[2] Each discrete point $(\boldsymbol{p}_i, t_i, \boldsymbol{x}_i)$ has a spatial position $\boldsymbol{p}_i \in \mathbb{R}^3$, a time $t_i \in \mathbb{R}$, and an RGB color $\boldsymbol{x}_i \in \mathbb{R}^3$ where the subscript i indicates the index. This information can be obtained realistically using a regular camera coupled with either a depth camera or a LiDAR sensor, aggregating data over multiple frames. Note that the input point cloud is only a partial scan, and consequently there are missing points due to occlusions, which makes this a challenging task. Our goal is to learn a mapping from $\mathcal{X}$ to a complete point cloud $\mathcal{Y} = \{ (\boldsymbol{p}_j, t_j, \boldsymbol{y}_j) \}$ that densely encodes the full spacetime volume. The output vector $\boldsymbol{y}_j \in \mathbb{R}^d$ encodes any labels that we want to predict, such as color or semantic category.

3.1. Model

Point clouds are often treated as discrete, which causes them to have an irregular structure that makes traditional deep representation learning on them difficult. In order for our model to learn to persist points after they become occluded, we need a mechanism to create new points that have not been observed.

We will model the output point cloud as continuous, which allows us to compactly parameterize all the putative points across the 4D spacetime volume. Let $(\boldsymbol{p}_q, t_q) \in \mathbb{R}^4$ be a continuous spacetime query coordinate. Our model estimates the features $\hat{\boldsymbol{y}}$ located at $(\boldsymbol{p}_q, t_q)$, which may be occluded, with the decomposition:

$$\hat{\boldsymbol{y}}(\boldsymbol{p}_q, t_q) = f(\boldsymbol{p}_q, t_q; \phi(\mathcal{X})) \tag{1}$$

[2]A *point cloud video* assumes known camera parameters to deproject the RGB + depth information into some canonical coordinate system.

where ϕ is a feature extractor and f is our continuous representation. There are many possible choices for $\phi(\mathcal{X})$ [44, 45, 68], and we use the architecture from the Point Transformer network [69], which produces contextualized features for every point in the (subsampled) input.

The model is able to continuously predict a representation $\hat{\boldsymbol{y}}$ for the entire spacetime volume, shown in Figure 2. We can train $\hat{\boldsymbol{y}}$ for many different point cloud tasks, providing us the flexibility to predict, for example, geometry, semantic information, color, or object identity.

Our model uses a continuous representation similar to methods in neural rendering and computer graphics [16, 17, 40, 43, 63], which also enjoy significant computational advantages from the compact scene representation. However, our approach operates on point clouds instead of signed distance functions or radiance fields, which allows us to train and apply our model for many tasks besides view synthesis. Furthermore, our approach is conditioned on a set of frames in a dynamic point cloud video, enabling the model to learn a rich spatiotemporal representation for occluded objects.

3.2. Point Attention

Given the query coordinate $(\boldsymbol{p}_q, t_q)$, we need to estimate the contents at that spatiotemporal location. However, in a video with occlusions, the contextual evidence for those contents might be both spatially and temporally far away.

We introduce a cross-attention layer that uses the query coordinates to attend to the input video in order to generate this prediction. We illustrate this process in Figure 3. Typi-

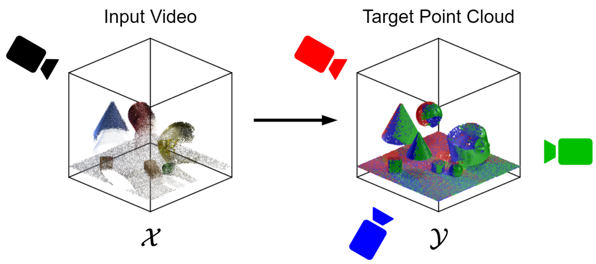

Figure 4. **Learning About Occlusions –** When an occlusion occurs in the input video, it is typically still visible from other viewpoints. For example, the smaller cone becomes fully occluded in the last input frame, but is revealed again by the ground truth. Our approach capitalizes on this natural clue to provide geometrically consistent multi-view self-supervision to the model, thus distilling the notion that objects persist across space and time.

cally, attention works by using a query to attend to keys and retrieve the relevant values. In our case, we will operate on the featurized point cloud $\mathcal{Z} = \phi(\mathcal{X}) = \{(\boldsymbol{p}_i, \boldsymbol{\alpha}_i)\}$. We form the keys $\boldsymbol{K}$ and values $\boldsymbol{V}$ from $\boldsymbol{\alpha}_i$, and the the relative positional encodings $\boldsymbol{\Delta}$ from $\boldsymbol{p}_q - \boldsymbol{p}_i$. We form the query $\boldsymbol{Q}$ from $\boldsymbol{p}_q$ and t_q.

Our layer can be recursively stacked, allowing the model to build increasingly rich representations of the scene. We implement the above vector cross-attention strategy through the following calculations, inspired by [69]:

$$\boldsymbol{Q}_q = \boldsymbol{w}_Q^\intercal \boldsymbol{\beta}_{q,n} \qquad \text{(queries)} \qquad (2)$$

$$\boldsymbol{V}_i = \boldsymbol{w}_V^\intercal \boldsymbol{\alpha}_i \qquad \text{(values)} \qquad (3)$$

$$\boldsymbol{K}_i = \boldsymbol{w}_K^\intercal \boldsymbol{\alpha}_i \qquad \text{(keys)} \qquad (4)$$

$$\boldsymbol{\Delta}_{q,i} = \boldsymbol{w}_\Delta^\intercal (\boldsymbol{p}_i - \boldsymbol{p}_q) \qquad \text{(positions)} \qquad (5)$$

where $\boldsymbol{\beta}_{q,n}$ is a feature vector that encodes the features for query point $(\boldsymbol{p}_q, t_q)$. The base case is $\boldsymbol{\beta}_{q,0} = \text{MLP}(\boldsymbol{p}_q, t_q)$, and as we stack the cross-attention block, it will be iteratively refined with:

$$\boldsymbol{\beta}_{q,n+1} = \sum_{i \in \mathcal{M}(q)} \rho(\gamma(\boldsymbol{Q}_q - \boldsymbol{K}_i + \boldsymbol{\Delta}_{q,i})) \odot (\boldsymbol{V}_i + \boldsymbol{\Delta}_{q,i}) \quad (6)$$

where $\mathcal{M}(q)$ is a set of nearest neighbors within $\phi(\mathcal{X})$ around $\boldsymbol{p}_q$, ρ is the softmax operation for normalization, γ is a mapping MLP that produces the attention weights, and $\odot$ is an element-wise product to represent per-channel feature modulation [69].

We apply the operation in Equation (6) twice, meaning we terminate the recursion at $\boldsymbol{\beta}_{q,2}$. This produces a feature vector that describes the contents at the query location, which we decode into the predicted labels. Finally, we use an MLP to map $\boldsymbol{\beta}_{q,2}$ to $\hat{\boldsymbol{y}}(\boldsymbol{p}_q, t_q)$.

3.3. Learning and Supervision

We train the model for 4D dynamic scene completion. Given several camera views of a scene, we assume known camera parameters to deproject their recordings into point clouds. We select one camera view to be the input view, which creates $\mathcal{X}$. To form the target $\mathcal{Y}$, we use the point cloud that merges all the camera views together. We train the model to predict the multi-view point cloud $\mathcal{Y}$ from the single-view point cloud $\mathcal{X}$, illustrated in Figure 4.

Due to the efficiency of our representation, we can train the model end-to-end for large spacetime volumes on standard GPU hardware. We minimize the loss function:

$$\min_{f,\phi} \mathbb{E}_{(\mathcal{X},\mathcal{Y})} \left[\sum_{(\boldsymbol{p}_q, t_q, \boldsymbol{y}_q) \in \mathcal{Y} \cup \mathcal{N}} \mathcal{L}\left(\hat{\boldsymbol{y}}(\boldsymbol{p}_q, t_q), \boldsymbol{y}_q\right) \right] \quad (7)$$

where $\mathcal{N}$ is a set of negative points randomly sampled uniformly from $\mathbb{R}^4$. Since the training data $\mathcal{Y}$ only contains solid points, the negative points cause the model to learn to distinguish which regions are empty space.

3.4. Tasks

Our framework is able to learn to reveal occlusions for several different tasks on point clouds. For every query point, the model produces a vector $\hat{\boldsymbol{y}}_i \in \mathbb{R}^d$, and we can supervise different dimensions of $\hat{\boldsymbol{y}}_i$ for various tasks. We select the loss function $\mathcal{L}$ depending on the dataset and task. We describe several options for the loss terms below.

Geometry completion distinguishes solid objects ($\sigma = 1$) from free space ($\sigma = 0$) within the scene, where the ground truth occupancy σ is inferred for every query point by thresholding its proximity to the target point cloud. Denoting $\hat{y}_\sigma$ as the relevant dimension of the $\hat{\boldsymbol{y}}$ vector, we apply a standard binary cross-entropy comparison as follows: $\mathcal{L}_\sigma = \mathcal{L}_{BCE}(\hat{y}_\sigma, \sigma)$.

Visual reconstruction means that, in addition to completing the missing regions, the model must also predict a color $\hat{\boldsymbol{y}}_c$ in RGB space. For the loss function, we use the $\mathcal{L}_1$-distance between the relevant output dimensions and the target $\boldsymbol{c}$: $\mathcal{L}_{\boldsymbol{c}} = \|\hat{y}_{\boldsymbol{c}} - \boldsymbol{c}\|_1$.

Semantic segmentation classifies every query point into S possible categories. The output is supervised with a cross-entropy loss between the predicted categories $\hat{\boldsymbol{y}}_s$ and the ground truth semantic label s: $\mathcal{L}_{\boldsymbol{s}} = \mathcal{L}_{CE}(\hat{\boldsymbol{y}}_s, s)$

Instance tracking tasks the model with localizing an object, even through total occlusions, that was highlighted with a mask in only the first frame.[3] To do this, we add an extra dimension τ_i to the input point cloud $\mathcal{X}$, that indicates which points belong to the object of interest. We then train

[3] This is similar to most *semi-supervised video object segmentation* setups [6, 59], but in 3D space instead. Note that the object may be partially not completely occluded at the beginning of the video for this to work.

the model to propagate this indicator throughout the rest of the video, where $\hat{y}_\tau$ is the relevant dimension in the output. We use the binary cross-entropy loss between the tracking flag $\hat{y}_\tau$ and τ: $\mathcal{L}_\tau = \mathcal{L}_{BCE}(\hat{y}_\tau, \tau)$.

These four loss terms can be linearly combined to form the overall objective:

$$\mathcal{L} = \lambda_\sigma \mathcal{L}_\sigma + \lambda_{\boldsymbol{c}} \mathcal{L}_{\boldsymbol{c}} + \lambda_{\boldsymbol{s}} \mathcal{L}_{\boldsymbol{s}} + \lambda_\tau \mathcal{L}_\tau \tag{8}$$

3.5. Inference

After learning, we will be able to estimate a continuous representation of a point cloud from a video. For many applications, we need a sampling procedure to convert the continuous cloud into a discrete cloud. Depending on our choice of sampling technique, we can construct arbitrarily detailed point clouds at test time.

Since the target is unknown at test time, we sample query coordinates $(\boldsymbol{p}_q, t_q)$ uniformly at random within a 4D spacetime volume of interest. We generate discrete point clouds by filtering predictions according to solidity, only retaining a query point whenever the predicted occupancy is above some threshold, *i.e.* $\hat{y}_\sigma \geq \sigma_T$.

For visualization purposes, we can also convert the predictions to scene meshes. The surface $\mathcal{S}$ of a mesh at time t is implicitly defined as the zero-level set of the predicted occupancy $\hat{\sigma}$ relative to the threshold σ_T, *i.e.* $\mathcal{S} = \{ \boldsymbol{x} \in \mathbb{R}^3 \mid \hat{y}_\sigma(\boldsymbol{x}, t) = \sigma_T \}$, where $\sigma_T = 0.5$.

After sampling a point cloud, or a mesh via the cube marching algorithm, we colorize it by retrieving either the predicted color $\hat{\boldsymbol{y}}_c$, the semantic category $\hat{\boldsymbol{y}}_s$, or the tracking flag $\hat{y}_\tau$, associated with every coordinate.

3.6. Implementation Details

The feature encoder ϕ interleaves 4 self-attention layers with 3 down transition modules [69] to generate the featurized point cloud $\mathcal{Z}$ from $\mathcal{X}$. The continuous representation f, conditioned on $\mathcal{Z}$, accepts arbitrary 4D query coordinates $(\boldsymbol{p}_q, t_q)$ as input, applies Fourier encoding [34], and interleaves 6 residual MLP blocks [67] with 2 cross-attention layers to produce $\hat{\boldsymbol{y}}$.

We feed in $T = 12$ frames with $|\mathcal{X}| = 14,336$ points in total, and train the model to predict the last $U = 4$ frames, such that the first $T - U = 8$ frames serve as an opportunity to aggregate and process spatiotemporal context. More details can be found in the supplementary material.

4. Datasets

In order to train and evaluate our model, particularly in terms of its ability to handle occlusions, we require multi-view RGB-D video from highly cluttered scenes. To this end, we contribute two high-quality synthetic datasets, shown in Figs. 5 and 6. Brief descriptions are provided below, with further details in the supplementary material.

4.1. GREATER

We extend CATER [18] (which is in turn based on CLEVR [23]) in order to increase the degree of occlusions, and call our proposed dataset GREATER. Each scene in GREATER contains 8 to 12 cubes, cones, cylinders and spheres that move around, occluding one another in random ways. Partial and complete occlusions are happening constantly to the input view, which are only revealed by the target point clouds, allowing for effective learning and benchmarking of our model. We capture 7,000 scenes lasting 12 seconds each, with data captured from 3 random views spaced at least 45° apart horizontally, and a train/val/test split of 80%/10%/10%. On the GREATER dataset, we train our model to predict geometry, color, and tracking.

4.2. CARLA

While GREATER already exhibits many non-trivial scene configurations and movement patterns, it may be desirable to apply 4D video completion within more realistic environments as well. Since object permanence is paramount for situational awareness in the context of driving and traffic scenarios, we employ the state-of-the-art driving simulator CARLA [15] to generate a dataset of complex, dynamic road scenes. We sample 500 scenes lasting 100 seconds each, with data captured from 4 fixed views, and a train/val/test split of 80%/8%/12%. The scenes cover a wide variety of different towns, vehicles, pedestrians, traffic scenarios, and weather conditions. On the CARLA dataset, we teach our model to perform geometric as well as semantic scene completion.

5. Experiments

To display the generality of our framework, we test it on a variety of tasks across different datasets. Crucially, all the tasks can be trained end-to-end simultaneously in the same model. We set $(\lambda_\sigma, \lambda_{\boldsymbol{c}}, \lambda_{\boldsymbol{s}}, \lambda_\tau) = (1, 1, 0, 1)$ for GREATER and $(\lambda_\sigma, \lambda_{\boldsymbol{c}}, \lambda_{\boldsymbol{s}}, \lambda_\tau) = (1, 0, 0.6, 0)$ for CARLA.

5.1. Evaluation Metrics

We evaluate models using the Chamfer Distance (CD) metric between the predicted point cloud $\hat{\mathcal{Y}}$ and target point cloud $\mathcal{Y}$:

$$CD\left(\hat{\mathcal{Y}}, \mathcal{Y}\right) = \frac{1}{|\hat{\mathcal{Y}}|} \sum_{i \in \hat{\mathcal{Y}}} \min_{j \in \mathcal{Y}} |\boldsymbol{p}_i - \boldsymbol{p}_j|_2 + \frac{1}{|\mathcal{Y}|} \sum_{j \in \mathcal{Y}} \min_{i \in \hat{\mathcal{Y}}} |\boldsymbol{p}_j - \boldsymbol{p}_i|_2 \tag{9}$$

For geometry completion, we initially consider all points, but wish to specifically study occlusions as well. To that end, we filter all points by whether they belong to an occluded instance or not, which we can approximate by comparing different views with each other. If the filtered output

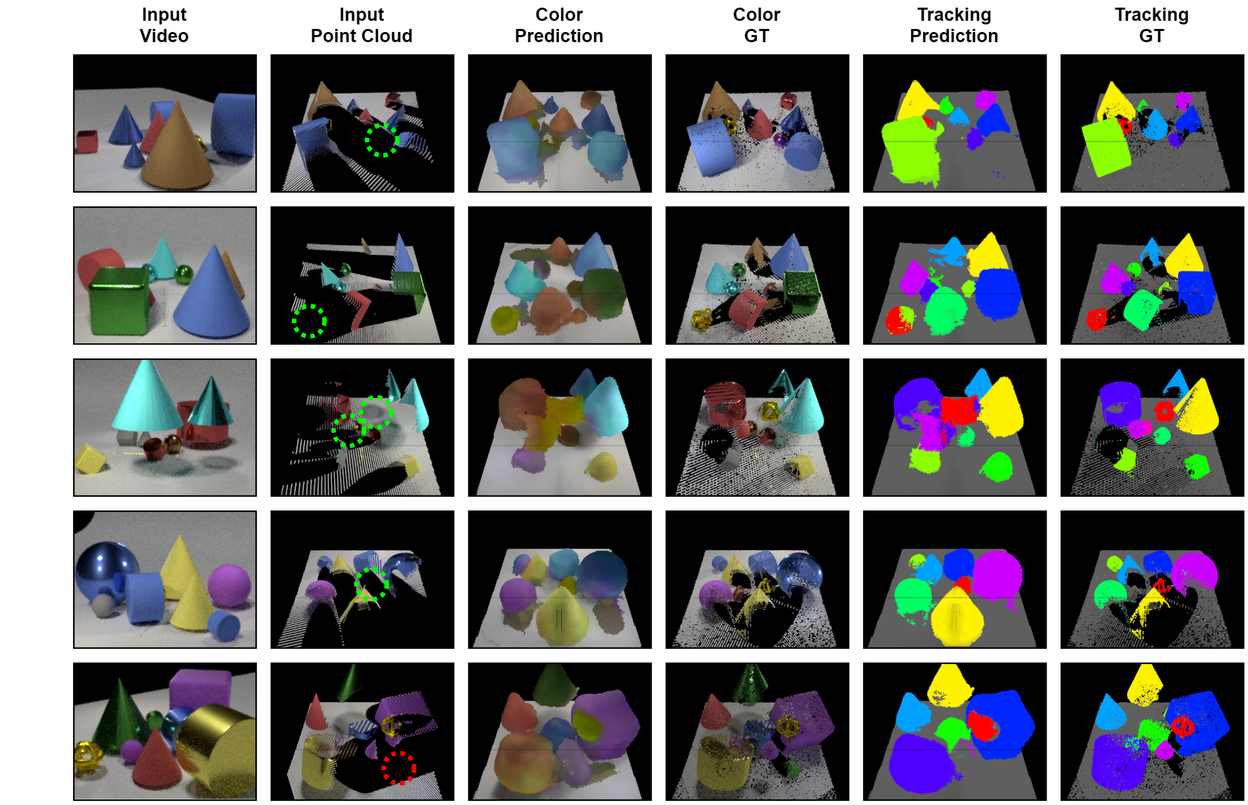

Figure 5. **Results for GREATER –** We show inputs, predictions, and ground truths. Our model receives color point clouds as input (second column), and we show the corresponding video frame in column one as reference. The third column represents *both* the geometry reconstruction and color prediction tasks. We note how the model is able to (1) perform scene completion by filling in partially observed objects, *i.e*. resolve amodal completion, and even (2) recover totally occluded objects, including when there are multiple occurring at once. For *total* occlusions, we circle the corresponding locations in the input for **true positives** in green and **false negatives** in red. While we show only the last frames in this figure, the model predicts the scene at different time steps, capturing scene dynamics.

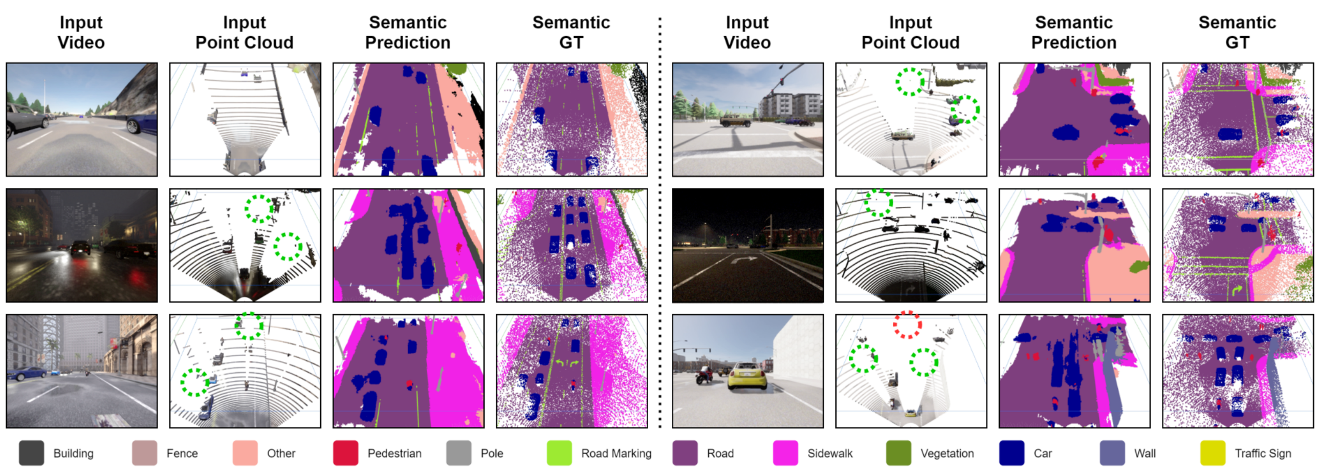

Figure 6. **Results for CARLA –** We show inputs, predictions and ground truths. Our model receives the point cloud video whose last frame is depicted in the second column, and predicts scene occupancy and semantic completion data for every sampled query point. Considering the limited input information, our model is capable of reconstructing the whole scene with high accuracy. Just as in Figure 5, all inputs and outputs are 4D meaning that they actually consist of multiple frames – please see our webpage for animated visualizations.

Method	Geometry		Tracking	
	All	Occ.	Avg. Inst.	Occ.
No local features	0.78	0.73	4.50	4.86
No time	0.26	0.49	1.59	4.25
No self-attention	0.26	0.37	1.12	1.58
No cross-attention	0.32	0.41	1.21	1.73
No attention	0.40	0.48	1.42	2.00
Copy input view	0.48	1.92	-	-
Assume stationary	-	-	2.34	3.43
PCN [68]	0.59	0.97	-	-
Ours	**0.22**	**0.33**	**1.05**	**1.32**

Table 1. **Results for GREATER – geometry completion and instance tracking tasks**. We report the Chamfer Distance (lower is better). In addition to outperforming all ablations and baselines on both tasks, our model predicts occluded objects nearly as well as visible objects.

Method	Geometry	Semantic Segmentation				
	All	Avg. Cls.	Ped.	Veh.	Occ. Ped.	Occ. Veh.
No local features	1.12	15.23	19.26	6.81	19.95	6.97
No time	0.55	6.76	10.16	5.51	15.56	10.99
No self-attention	0.50	6.60	5.19	**3.29**	5.98	4.73
No cross-attention	0.73	7.11	7.57	4.22	11.33	7.55
No attention	0.71	9.21	9.68	4.61	13.66	7.51
Copy input view	1.39	-	-	-	-	-
PCN [68]	11.79	-	-	-	-	-
Ours	**0.47**	**5.82**	**4.76**	3.42	**5.71**	**4.14**

Table 2. **Results for CARLA – geometry completion and semantic completion tasks**. We report the Chamfer Distance (lower is better). Our model significantly outperforms almost all baselines and ablations, especially for occluded pedestrians ("Ped.") and vehicles ("Veh.").

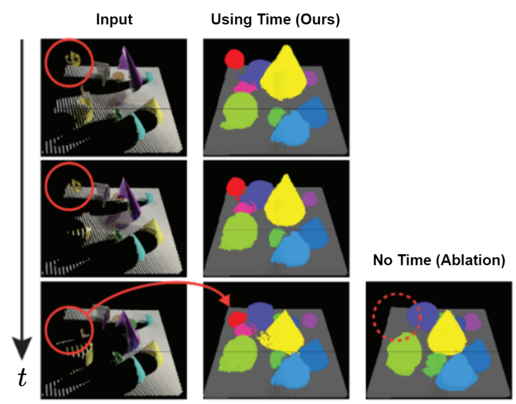

Figure 7. **Importance of Time –** In this figure, we show a video where an occlusion takes place (see yellow orb, highlighted by the red circle), and tracking predictions of our model. Because our model takes video as input, it is capable of tracking the orb through the occlusion. The single-frame baseline ("no time") cannot recover the occluded orb, as it does not have access to temporal context. This example shows that (1) temporal context is important, and (2) our model learns to use it.

point cloud is empty (which typically corresponds to false negatives), we substitute the prediction for a single point at the center of the scene, as the CD would become undefined otherwise.

For instance tracking in GREATER, we track one object at a time, and merge the resulting predictions at test time. Concretely, we obtain multiple tracks by assigning the instance tag with the most confident score $\hat{y}_\mathcal{T}$ to each point, but only if $\hat{y}_\mathcal{T} \geq 0.5$. Then, for every instance tag, we calculate the CD between only its corresponding predicted points and the ground truth object points, and subsequently average this value over all instances within a scene. We also report the average over occluded objects only.

For semantic segmentation in CARLA, we use a similar workflow as for tracking, but average over all categories instead of instances. We study two important classes (pedestrians and vehicles) separately, which implies filtering both the predictions and targets by whether their ground truth semantic categories belong to those respective classes before reporting the CD values. Additionally, we filter for *occluded* pedestrians and vehicles. In both cases, we average over all instances per scene such that every pedestrian or car is treated equally.

5.2. Ablations and Baselines

Ablations. To show how various architectural choices affect our model's performance, we perform ablations to its four main components by: (1) removing local features from f; (2) removing the temporal dimension; (3) removing self-attention from ϕ; (4) removing cross-attention from f; (5) combining (3) and (4). Ablation (1) implies that instead of conditioning on $\mathcal{Z}$, we only pass a global embedding (that is the average of the features over all points in $\mathcal{Z}$) from ϕ to f. In (2), the model is no longer burdened with predicting a 4D dynamic representation consisting of multiple frames, and the task becomes 3D scene completion instead – given a single frame, predict a single frame. For (3), we replace self-attention layers in the point transformer with a simple point-wise linear projection. For (4), since f cannot attend to $\mathcal{Z}$ anymore, we feed in the nearest neighbor in $\mathcal{Z}$ of every query point to f along with the query point itself.

Baselines. For our primary scene geometry reconstruction task, we adapt Point Completion Network (PCN) [68] to our setup. Additionally, we evaluate a 'Copy input view' baseline, where the prediction is simply the input point cloud that the model sees, *i.e.* the identity operation. Comparison with this baseline shows the benefits of our approach in revealing occlusions. Finally, for tracking, we evaluate the baseline where the marked instance is propagated but remains stationary after the first frame, which is also the only time that the model sees its mask.

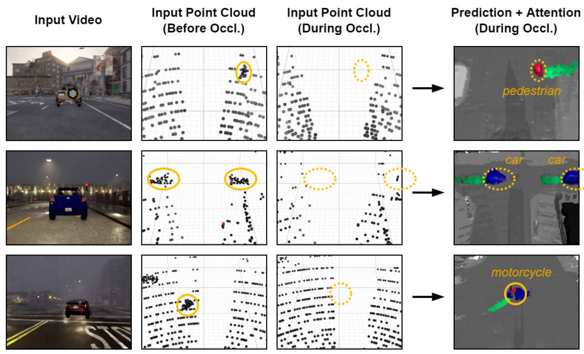

Figure 8. **Visualizing Attention –** Why did the model predict the **occluded object(s)**? By backtracking neuron activations through all cross-attention and self-attention layers, we see that a mechanism of temporal correspondence emerges. In this example, the **attention weights** highlight input points that represent the trajectory of the object(s) over time, which suggests that our model implicitly learns to track them in order to succeed at 4D scene completion.

5.3. Quantitative Results

See Tables 1 and 2. Occlusion metrics ("Occ.") are for objects that are more than 80% occluded, as inferred from the number of points per instance that are visible from each view. Our non-ablated model consistently outperforms most baselines and ablations with significant margins. In particular, these results demonstrate that incorporating attention mechanisms is clearly beneficial for handling occlusions and performing spatiotemporal inpainting, suggesting that a robust notion of object permanence was successfully learned.

Although the ablations without time succeed at reconstructing a 3D snapshot of the scene with relatively good quality, they are significantly worse at predicting occluded objects such as vehicles or pedestrians in CARLA, which is a critical aspect of interpreting traffic scenes. Figure 7 further demonstrates that temporal context is essential to understand scene dynamics. Moreover, providing contextualized local features is also vital for the performance of the model.

5.4. Visualizations

In this section, we visualize the inputs and predictions of our model, for both datasets.

Figures 5 and 6 show our model predictions for the GREATER and CARLA datasets. In both cases, the model is capable of simultaneously performing geometry completion along with other prediction tasks such as visual reconstruction, instance tracking, or semantic segmentation. Note that geometry completion is a prerequisite to solving any other prediction task, since other tasks also require knowledge of objects that are *not visible* in the input frame.

Our model is capable of completing the scene with great

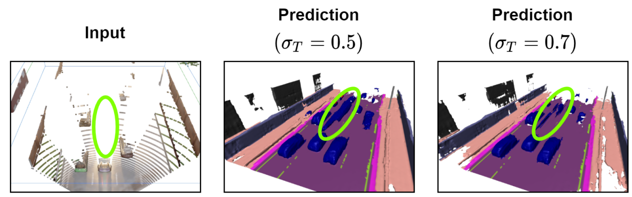

Figure 9. **Interpreting Uncertainty –** Because it is often hard to perceive what is ahead of the car in front of you, some regions of the input can remain unseen in the LiDAR point cloud (shown on the left) throughout the entire input video, causing artefacts such as a 'long car' to crop up. By varying the occupancy threshold σ_T of the implicit surface, we can control the degree of certainty at which to visualize the prediction. For example, in the above example the model is found to be less certain about the presence of the 'long car' relative to the rest of the scene (which remains reconstructed accurately), as indicated by the artefact's disappearance with increasing σ_T.

detail, even when presented with a limited density of input points. Specifically, when trained on CARLA the model is capable of reconstructing—and predicting class information about—relatively small objects such as street poles (gray), pedestrians (red), or traffic signs (yellow). It does so for different temporal steps, and even when there are occlusions. We observe that the model trained on CARLA does sometimes struggle in hard cases that involve objects moving across long-term occlusions (which presents a limitation and opportunity for future work), but it usually generates an accurate, complete reconstruction in most other scenarios, especially when just amodal completion is involved.

In Figure 8, we visualize the attention of the model during occlusions in order to understand the mechanism it uses to resolve them. Specifically, we adapt attention rollout [1] to our architecture, and visualize the input points (across time) that contribute the most to the specific output class we want to analyze.

6. Discussion

We introduce the task of 4D dynamic scene completion, along with two datasets for understanding occlusions, and showcase a continuous representation that incorporates cross-attention as an initial attempt toward solving this challenge. We believe these these techniques and benchmarks will be useful in the context of scene completion, spatiotemporal inpainting, and object permanence.

Acknowledgements: This research is based on work supported by Toyota Research Institute, the NSF CAREER Award #2046910, and the DARPA MCS program under Federal Agreement No. N660011924032. DS is supported by the Microsoft PhD Fellowship. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the sponsors.

References

[1] Samira Abnar and Willem Zuidema. Quantifying attention flow in transformers. In *ACL*, 2020. 8

[2] Andréa Aguiar and Renée Baillargeon. 2.5-month-old infants' reasoning about when objects should and should not be occluded. *Cognitive psychology*, 39(2):116–157, 1999. 1

[3] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic lidar segmentation. In *CVPR*, 2021. 2

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

[5] Renee Baillargeon, Elizabeth S Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3):191–208, 1985. 1

[6] Sergi Caelles, Alberto Montes, Kevis-Kokitsi Maninis, Yuhua Chen, Luc Van Gool, Federico Perazzi, and Jordi Pont-Tuset. The 2018 davis challenge on video object segmentation. *arXiv preprint arXiv:1803.00557*, 2018. 4

[7] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 2

[9] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 1

[10] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. When vision transformers outperform resnets without pretraining or strong data augmentations. *arXiv preprint arXiv:2106.01548*, 2021. 2

[11] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 2

[12] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 2

[13] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 2

[15] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *CoRL*. PMLR, 2017. 5

[16] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4d view synthesis and video processing. In *ICCV*, 2021. 2, 3

[17] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *ICCV*, 2021. 2, 3

[18] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *ICLR*, 2020. 5, 3

[19] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 1

[20] Yan Huang and Irfan Essa. Tracking multiple objects through occlusions. In *CVPR*, 2005. 2

[21] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *CVPR*, 2020. 2

[22] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. 2

[23] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017. 5

[24] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet for joint object categorization and unsupervised pose estimation from multi-view images. *TPAMI*, 43(1):269–283, 2021. 2

[25] Tarasha Khurana, Achal Dave, and Deva Ramanan. Detecting invisible people. In *ICCV*, 2021. 2

[26] 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

[27] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019. 2

[28] Xinhai Liu, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *AAAI*, 2019. 2

[29] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9246–9255, 2019. 2

[30] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 2

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 2

[32] Lori Marino. Thinking chickens: a review of cognition, emotion, and behavior in the domestic chicken. *Animal Cognition*, 20(2):127–147, 2017. 1

[33] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 2

[34] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 5, 3

[35] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*, 2021. 2

[36] Hieu Tat Nguyen, Marcel Worring, and Rein Van Den Boomgaard. Occlusion robust adaptive template tracking. In *ICCV*, 2001. 2

[37] Jiyan Pan and Bo Hu. Robust occlusion handling in object tracking. In *CVPR*, 2007. 2

[38] Zizheng Pan, Bohan Zhuang, Jing Liu, Haoyu He, and Jianfei Cai. Scalable vision transformers with hierarchical pooling. In *ICCV*, 2021. 2

[39] 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

[40] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. NeRFies: Deformable neural radiance fields. *ICCV*, 2021. 2, 3

[41] Irene M Pepperberg and Mildred S Funk. Object permanence in four species of psittacine birds: An african grey parrot (psittacus erithacus), an illiger mini macaw (ara maracana), a parakeet (melopsittacus undulatus), and a cockatiel (nymphicus hollandicus). *Animal Learning & Behavior*, 18(1):97–108, 1990. 1

[42] Jean Piaget. *The construction of reality in the child*, volume 82. Routledge, 2013. 1

[43] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2, 3

[44] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2, 3

[45] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2, 3

[46] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 2

[47] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 2

[48] Christoph Rist, David Emmerichs, Markus Enzweiler, and Dariu Gavrila. Semantic scene completion using local deep implicit functions on lidar data. *TPAMI*, 2021. 2

[49] Aviv Shamsian, Ofri Kleinfeld, Amir Globerson, and Gal Chechik. Learning object permanence from video. In *ECCV*, 2020. 2

[50] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 2

[51] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *CVPR*, 2017. 2

[52] Elizabeth S Spelke and Gretchen Van de Walle. Perceiving and reasoning about objects: Insights from infants. *Spatial representation: Problems in philosophy and psychology*, pages 132–161, 1993. 1

[53] Andreas Steiner, Alexander Kolesnikov, , Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your ViT? Data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 2

[54] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 2

[55] Lyne P Tchapmi, Vineet Kosaraju, Hamid Rezatofighi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *CVPR*, 2019. 2

[56] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019. 2

[57] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *ICCV*, 2021. 2

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

[59] Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *CVPR*, 2019. 4

[60] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 790–799, 2020. 2

[61] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, , and Justin M. Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *CoRL*, 2021. 2

[62] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1939–1948, 2020. 2

[63] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *CVPR*, 2021. 2, 3

[64] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018. 2

[65] Zhenjia Xu, Zhanpeng He, Jiajun Wu, and Shuran Song. Learning 3d dynamic scene representations for robot manipulation. In *CoRL*, 2020. 2

[66] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 1

[67] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. PixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 5, 3

[68] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point completion network. In *3DV*, 2018. 1, 2, 3, 7

[69] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 1, 2, 3, 4, 5