Dynamic Plane Convolutional Occupancy Networks

Stefan Lionar[1*] Daniil Emtsev[1*] Dusan Svilarkovic[1*] Songyou Peng[1,2]
[1]ETH Zurich [2] Max Planck ETH Center for Learning Systems
{slionar, demtsev, dsvilarko}@ethz.ch songyou.peng@inf.ethz.ch

Abstract

Learning-based 3D reconstruction using implicit neural representations has shown promising progress not only at the object level but also in more complicated scenes. In this paper, we propose Dynamic Plane Convolutional Occupancy Networks, a novel implicit representation pushing further the quality of 3D surface reconstruction. The input noisy point clouds are encoded into per-point features that are projected onto multiple 2D dynamic planes. A fully-connected network learns to predict plane parameters that best describe the shapes of objects or scenes. To further exploit translational equivariance, convolutional neural networks are applied to process the plane features. Our method shows superior performance in surface reconstruction from unoriented point clouds in ShapeNet as well as an indoor scene dataset. Moreover, we also provide interesting observations on the distribution of learned dynamic planes.

1. Introduction

Exploiting 3D information, such as point clouds, has become increasingly popular for various computer vision tasks, such as self-driving vehicles, indoor navigation, and robotics [35, 41]. 3D surface reconstruction from point clouds promises better precision for these applications. In recent years, learning-based 3D reconstruction using implicit neural representations in the continuous domain has gained much attention due to its ability to produce smooth and expressive reconstruction with a significant reduction of memory footprint [5, 23, 28].

However, the pioneering implicit 3D reconstruction approaches are limited to single objects and do not scale to larger scenes due to the use of global embeddings. Some recent works [2, 5, 13, 17, 28] noticed this problem and introduced various ways to exploit the information of local structures. While those works have introduced a significant improvement in the object or scene-level reconstruction, the work of Peng *et al*. on convolutional occupancy networks (ConvONet) [28] is the first to demonstrate an accurate and efficient reconstruction of large-scale scenes from point clouds without the need for online optimization. One critical success factor of this work is encoding 3D inputs to 2D canonical planes, which are then processed by convolutional neural networks (CNNs). In this way, the translation equivariant property of CNNs and the local similarity of 3D structures are exploited, enabling the accurate reconstruction of complex scenes. In their work, three canonical planes are pre-defined following Manhattan-world assumption [7] on the dataset orientation.

In this work, we propose *Dynamic Plane Convolutional Occupancy Networks*[1], an implicit representation that enables accurate scene-level reconstruction from 3D point clouds. Instead of learning features on three pre-defined canonical planes as in [28], we use a fully-connected network to learn dynamic planes, on which we project the encoded per-point features. The learned dynamic planes capture rich features over the most informative directions. We systematically investigate the use of up to 7 learned planes and demonstrate progressive improvements by increasing the number of learned planes in our experiment. The detailed architecture of our model is illustrated in Fig. 1. Compared to [28], our model introduces another degree of precision by learning features that are more specific to every object and plane.

In summary, the main contributions of our paper are as follows:

- We fully leverage deep neural networks in feature learning to predict the best planes for 3D surface reconstruction tasks from unoriented point clouds.
- We show superiority over state-of-the-art approaches in the task of 3D surface reconstruction at both object and scene level.
- We provide various observations on the distribution of the dynamic planes from intensive experiments.

*Equal contribution. This is a 3D Vision course project at ETH Zurich.

[1]Code is available at: https://github.com/dsvilarkovic/dynamic_plane_convolutional_onet.

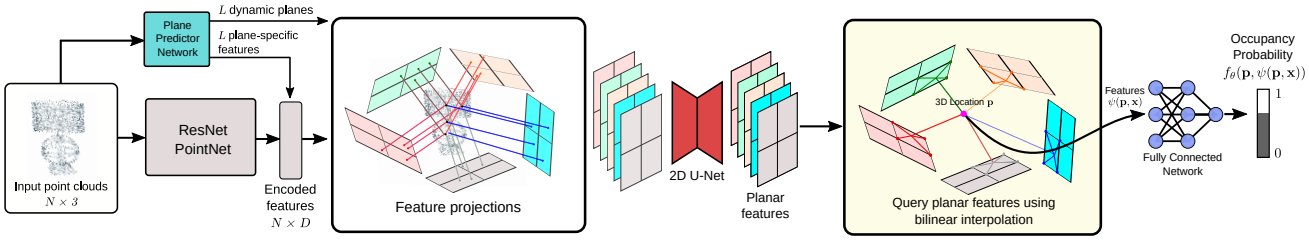


Figure 1: **Dynamic Plane Convolutional Occupancy Networks pipeline.** N input point clouds are encoded to per-point features by ResNet PointNet [30] with D as the feature dimension. Concurrently, a shallow plane predictor network learns L dynamic planes and plane-specific features from the input point clouds. We sum the plane-specific features to all of the encoded per-point features with respect to individual dynamic planes. Next, the summed features are projected to the dynamic planes. The projected plane features are then processed using U-Net [32] with shared weights among planes. In the decoding phase, the occupancy of a uniformly sampled point $\mathbf{p}$ is predicted by a shallow fully-connected network conditioned on the queried local planar features.

In addition, we exploit the use of positional encoding proposed in [24], which maps the low dimensional 3D point coordinates to higher-dimension representations with periodic functions under various frequencies. While [24] shows its effectiveness on image rendering tasks, we demonstrate that the same positional encoding is also useful for 3D reconstruction tasks.

Additionally, we formulate a similarity loss function to govern the orientations of dynamic planes to orient in diverse directions. By using a high number of dynamic planes trained with the additional similarity loss function, we observe a considerable improvement in the reconstruction from point clouds with unseen orientations.

2. Related Work

Existing works on learning-based 3D reconstruction can be broadly categorized by the output representation: voxels, points, meshes, or implicit representations.

Voxel representations: Voxel might be the most widely used representation for 3D reconstruction [6, 36, 39], but is limited in terms of resolution due to its large memory consumption. To alleviate this problem, several works consider the multi-scale scheme or octrees for efficient space partitioning [11, 16, 22]. Even with these modifications, these approaches are still restricted by computation and memory.

Point representations: Point clouds are widely used either in robotics or computer graphics [12, 30, 31]. However, there are no topological relations among points, so extra post-processing steps are required [8, 10]. Several works also propose learning-based convolution operations on point clouds to describe the relation among points, analogous to the 2D convolution on image pixels in convolutional neural networks [19, 29, 34, 38]. Similar to our method in encoding point cloud representation, FPConv [19] aggregates features from point clouds onto 2D grids using a learned local flattening operation. Likewise, Point-PlaneNet [29] introduces a point cloud convolution operation called PlaneConv, which computes distances between points and aggregate them in a set of learned planes. Nevertheless, all these methods do not consider the surface reconstruction task, which is the focus of our paper.

Mesh representations: Meshes [14, 15, 18] emphasize topological relations by constructing vertices and faces, but mesh-based methods suffer from generating either shapes with only simple topology or self-intersecting meshes.

Implicit neural representations: Recently, implicit occupancy [23] and signed distance field [27] have been exploited for 3D reconstruction. In contrast to the aforementioned explicit representations, implicit representation can model shapes in a continuous manner. Therefore, better detail preservation and more complicated shape topologies can be obtained from the implicit representation. Many recent works explore various applications, e.g., learning the implicit representation only from 2D observations [20, 21, 25], encoding texture information [24, 26], or learning gradient fields [1]. Unfortunately, all these methods are still limited to the reconstruction of single objects or small scenes with restricted complexity and struggle to generalize to scenes outside of the training distribution.

The notable exception is Peng *et al*. [28]. They propose an architecture that enables large-scale 3D scene reconstruction by training on synthetic indoor scene dataset and testing its generalization to larger scenes such as ScanNet [9] and MatterPort3D [3]. Specifically, given a point cloud, this method projects point-wise features onto the canonical planes or volume grids and then use U-Net [32] to aggregate both local and global information. In this way, the inductive biases are effectively exploited. However, considering only canonical planes may cause performance loss when some object parts do not align well with the canonical directions (*e.g*., a wired lamp with complicated geometry, see the first row of Fig. 4). Therefore, we propose to learn planes with

a network. With such learned dynamic planes, our system shows better 3D reconstruction quality.

3. Method

Our goal is to reconstruct 3D scenes with fine details from noisy point clouds. To this end, we first encode the input point clouds into 2D feature planes, whose parameters are predicted by a fully-connected network. These feature planes are then processed using convolutional networks and decoded into occupancy probabilities via another shallow fully-connected network. Fig. 1 illustrates the overall workflow of the proposed method.

3.1. Encoder

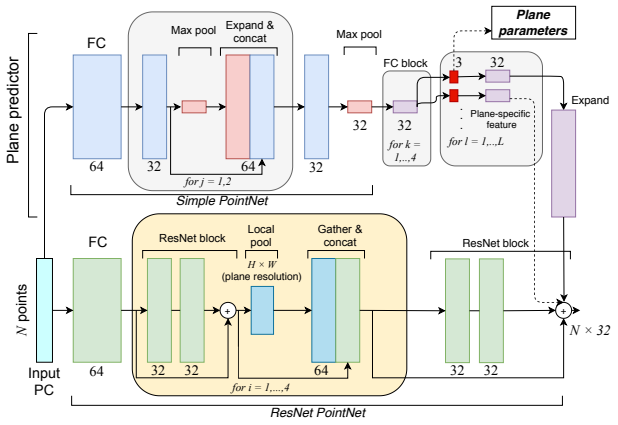


Figure 2: **Encoder architecture.** We use a ResNet PointNet [23, 28] to extract the per-point features. On top of it, we add a plane predictor network to predict the dynamic plane parameters, which consists of a simple PointNet [30].

The architecture of our encoder is illustrated in Fig. 2. We describe each part as follows.

Point cloud encoding: Given a noisy point cloud, we first form a feature embedding for every point with ResNet PointNet [23], in which we perform local pooling according to the predefined plane resolution [28]. We are applying a rather simple network here for the proof of concept, but other advanced feature extractors, e.g., PointNet++ [31] or Tangent Convolution [33], can also be used.

Dynamic plane prediction: Having the point-wise features, we can then construct the planar features. Mathematically, a plane is defined by a normal vector $\mathbf{n} = (a, b, c)$ and a point (x, y, z) which a plane passes through [29]: $ax + by + cz = d$. Peng *et al*. [28] simply project features onto canonical planes, i.e., 3 planes aligned with the axes of the coordinate frame, $x = 0, y = 0, z = 0$. Unlike [28], we introduce another shallow fully-connected network to regress the plane parameter (a, b, c). As illustrated in the upper branch of Fig. 2, we perform max pooling globally on all points in the point cloud because a global context is needed to search for the proposal of the best possible planes. Since different input point clouds might be predicted with different planes, we call this process dynamic plane prediction. Note that we directly set the intercept of the plane d to 0 because the shifts along the normal direction do not change the feature projection process.

After the prediction of plane parameters, we pass it through one layer of FC to obtain a feature for every dynamic plane. This feature is expanded, matching the number of input point cloud and summed up with the last layer of ResNet PointNet with respect to the individual dynamic plane, and thus we call it the plane-specific feature. Our main intention is to allow backpropagation into the plane predictor network, but we also empirically find that this individual summation operation improves the reconstruction quality. One possible reason is that it allows the networks to learn varying *emphasis* over the feature dimension of the last layer of ResNet PointNet with respect to the individual dynamic plane.

Once having the predicted plane parameters, we project the summed-up features onto the dynamic planes with a defined size of $H \times W$ grids and apply max pooling for the features falling into the same grid cell.

Planar projection: In order to project the encoded features to the dynamic planes, whose normals can point to any directions, we sequentially apply basis change, orthographic projection, and normalization to always keep them inside $H \times W$ grids. Denoting the three basis vectors of canonical axes, $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, where $\mathbf{k}$ is the basis vector of the ground plane and $\mathbf{n}$ is the learned plane normal, those operations are detailed as follows and illustrated in Fig. 3.

To perform basis change, we normalize $\mathbf{n}$ into a unit vector $\hat{\mathbf{n}}$ and obtain the rotation matrix R that aligns $\mathbf{k}$ with $\hat{\mathbf{n}}$. Let $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^\top = \mathbf{k} \times \hat{\mathbf{n}}$, the rotation matrix R is defined as:

$$R = I + [\mathbf{v}]_\times + [\mathbf{v}]_\times^2 \frac{1 - \mathbf{k} \cdot \hat{\mathbf{n}}}{\|\mathbf{v}\|^2}, \tag{1}$$

where $[\mathbf{v}]_\times$ is the skew matrix:

$$[\mathbf{v}]_\times := \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{2}$$

With the rotation matrix R, we rotate the axes $\mathbf{i}$ and $\mathbf{j}$ to obtain $\mathbf{i}_p$ and $\mathbf{j}_p$. Now, the vectors $\mathbf{i}_p$, $\mathbf{j}_p$ and $\hat{\mathbf{n}}$ are orthogonal to each other, serving as the basis of the predicted plane coordinate system.

Next, we convert point coordinates from the world coordinate to the plane coordinate system and project the features orthographically to the predicted plane ("new ground

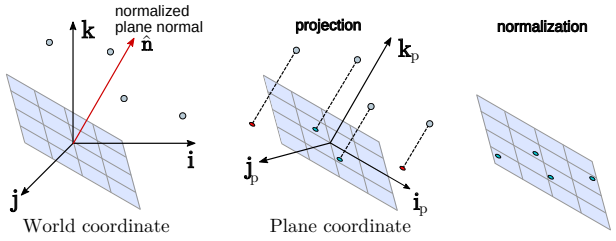


Figure 3: **Planar projection.** To obtain the projected feature plane, we sequentially apply basis change, orthographic projection, and normalization.

plane"). However, as our dynamic plane can orient to any direction, the orthographic projection of a point far from the centroid of 3D space might fall outside the $H \times W$ grids. To ensure all possible points after orthographic projection are inside the grids, we divide the coordinates after projection by a normalization constant $c \geq 1$. To find c, we first convert $\mathbf{i}_p$ and $\mathbf{j}_p$ to be inside the positive octant by taking their absolute values $\mathbf{i}_p^+$ and $\mathbf{j}_p^+$. Next, we obtain orthogonal projections of the vector $\mathbf{1} = [1, 1, 1]^\top$ to $\mathbf{i}_p^+$ and $\mathbf{j}_p^+$:

$$\mathbf{a_i} = \frac{\mathbf{1} \cdot \mathbf{i}_p^+}{\mathbf{i}_p^+ \cdot \mathbf{i}_p^+} \mathbf{i}_p^+, \qquad \mathbf{a_j} = \frac{\mathbf{1} \cdot \mathbf{j}_p^+}{\mathbf{j}_p^+ \cdot \mathbf{j}_p^+} \mathbf{j}_p^+ \tag{3}$$

Subsequently, we set c to be the maximum value between the lengths of these two projected vectors, $c = \max(|\mathbf{a_i}|, |\mathbf{a_j}|)$. The point coordinates under the plane coordinate system are divided by c so that all points lie inside the dynamic plane, where the point features are stored.

Once constructing the projected feature planes, we process them using U-Net [32] with shared weights for every plane. The final planar features have a dimension of $H \times W \times D$, where D is the predetermined hidden dimension.

3.2. Decoder

The goal of the decoder is to obtain the occupancy prediction of any point $\mathbf{p} \in \mathbb{R}^3$ given the aggregated planar features. Similar to how we project features in the encoder, we project $\mathbf{p}$ onto all dynamic planes. Next, we query the feature through bilinear interpolation of the planar features encoded at the four neighboring plane grids.

Occupancy prediction: Given an input point cloud $\mathbf{x}$, we predict the occupancy of $\mathbf{p}$ based on the feature vector at point $\mathbf{p}$, denoted as $\psi(\mathbf{p}, \mathbf{x})$:

$$f_\theta(\mathbf{p}, \psi(\mathbf{p}, \mathbf{x})) \rightarrow [0, 1] \tag{4}$$

We use the same network as [25], which consists of 5 ResNet blocks with ψ added to the input features of every block. The hidden dimension of all fully-connected layers is set to 32, which results in only $16,000$ parameters.

3.3. Training and Inference

During training, we apply binary cross-entropy loss between the occupancy prediction $f_\theta(\mathbf{p}, \psi(\mathbf{p}, \mathbf{x}))$ and the true occupancy value of $\mathbf{p}$. During inference, we apply *Multiresolution IsoSurface Extraction* (MISE) [23] to construct meshes.

3.4. Positional Encoding

The work of Mildenhall *et al*. [24] suggests that mapping input to higher dimension features using high-frequency functions before feeding into neural networks can result in the better fitting of data containing high-frequency variations. They introduce the positional encoding function:

$$\begin{aligned}\gamma(\mathbf{p}) = (&\sin(2^0\pi\mathbf{p}), \cos(2^0\pi\mathbf{p}), \ldots, \\ &\sin(2^{L-1}\pi\mathbf{p}), \cos(2^{L-1}\pi\mathbf{p}))\end{aligned} \tag{5}$$

where L is the frequency band. While [24] verifies its effectiveness for image rendering tasks, we show that its functionality also generalizes to 3D point cloud reconstruction, as seen in Table 1.

Specifically, we apply the positional encoding for the input 3D coordinates. Setting L to 10, we map the input point cloud $\mathbf{x}$ and query points $\mathbf{p}$ from $\mathbb{R}^3$ to $\mathbb{R}^{60}$.

4. Experiments

To evaluate our method, we conduct two experiments on surface reconstruction from noisy point clouds. We perform **object-level reconstruction** using ShapeNet [4] subset of Choy *et al*. [6], and **scene-level reconstruction** using synthetic indoor scene dataset from [28].

Metrics: We follow the metrics used by [28]: *Volumetric Intersection over Union (IoU)* measuring the matching volume of meshes intersection (higher is better), *Chamfer-*L_1 measuring the accuracy and completeness of the mesh surface (lower is better), *Normal Consistency* measuring the accuracy and completeness of the mesh normals (higher is better, and *F-score* measuring the harmonic mean of precision and recall between the reconstruction and ground truth (higher is better). The mathematical details are presented in the supplementary of [28].

Implementation details: We use 32 as the hidden feature dimension for both encoder and decoder in all experiments, and Adam optimizer with a learning rate of 10^{-4}. The depth of U-Net is chosen such that the receptive field is equal to the size of the feature plane. We choose a rather shallow fully-connected network as the plane predictor network. It has only around $13K$ parameters that are insignificant in size compared to the entire model, *e.g*., containing around $1.99M$ parameters for the model with 3 planes with a resolution of 64×64. The same depth of plane predictor network is used for the scene experiment. We run validation

	GPU Memory	IoU	Chamfer-L_1	Normal C.	F-score
Without PE					
ONet [23]	7.7G	0.761	0.087	0.891	0.785
ConvONet (3C) [28]	2.9G	0.884	0.045	0.938	0.943
Ours (3D)	3.2G	0.888	0.044	0.939	0.945
Ours (5D)	4.4G	0.889	0.043	0.940	0.948
Ours (7D)	5.5G	0.888	0.043	0.940	0.947
Ours (3C + 2D)	4.4G	0.890	0.043	0.940	0.947
Ours (3C + 4D)	5.5G	0.890	0.043	0.940	0.947
With PE					
ConvONet (3C) [28]	2.9G	0.889	0.043	0.938	0.945
Ours (3D)	3.2G	0.892	0.043	0.940	0.947
Ours (5D)	4.4G	0.894	**0.042**	**0.941**	0.950
Ours (7D)	5.5G	**0.895**	**0.042**	**0.941**	0.951
Ours (3C + 2D)	4.4G	0.892	0.043	**0.941**	0.948
Ours (3C + 4D)	5.5G	0.894	**0.042**	**0.941**	0.950
With PE + SL					
Ours (3D)	3.2G	0.891	0.043	0.940	0.948
Ours (5D)	4.4G	0.891	0.043	**0.941**	0.949
Ours (7D)	5.5G	**0.895**	**0.042**	**0.941**	**0.952**

PE = positional encoding. C = canonical planes. D = dynamic planes. SL = similarity loss.

Table 1: **Object-level 3D reconstruction from point clouds.** Results under all metrics are the mean for all 13 ShapeNet classes. The results for ONet [23] is taken from [28]. Class-specific results can be found in supplementary.

every 10,000 iterations and choose the best model based on the validation IoU.

4.1. Object-Level Reconstruction

We first evaluate the task of single object reconstruction. We sample 3000 points from the surface of ShapeNet objects and then apply Gaussian noise with zero mean and a standard deviation of 0.05. As for the query points (i.e., occupancy supervision), we follow [28] and uniformly sample 2048 points. We use a plane resolution of 64^2 and U-Net with a depth of 4. The batch size during training is set to 32. All of the object-level models are trained until at least 900,000 iterations to ensure convergence.

We run experiments with different combinations of canonical and dynamic planes. The results are summarized in Table 1. As we can notice, different variants of our method achieve state-of-the-art reconstruction accuracy on all metrics. Specifically, we outperform [28] while keeping the number of parameters at the same scale. We also observe progressive improvement when increasing the number of dynamic planes. Additionally, all results with positional encoding are better than without positional encoding. Moreover, as shown in our supplementary Section 2, we observe that adding positional encoding enables faster convergence. Qualitatively, the comparison against baselines is illustrated in Figure 4. In general, the improvement from our models is more pronounced on the challenging classes and objects with intricate structures, such as thin components and holes. More elaborated results detailing per-category performance and more qualitative results are presented in the supplementary materials.

Observation on plane distribution: Here, we discuss our observations on the distribution of the predicted dynamic planes. In the case of 3 dynamic planes, our network predicts three canonical planes for all objects. This finding is interesting because it verifies the use of canonical planes in [28], and the canonical planes indeed describe various shapes most effectively, as ShapeNet objects are aligned along those axes. In the case of 5 and 7 dynamic planes, there are combinations of flipping sets of normals (*e.g.* one normal pointing upward and the other downward). Such flipping sets of normals are equivalent to applying a horizontal flip on the projected encoded features. This observation is appealing because some recent works [37, 40] explicitly inject the object symmetry prior knowledge during training and show superior performance, while in our case, this symmetric property is implicitly encoded into our learned feature planes. Indeed, many objects in ShapeNet are symmetric about a plane, *e.g.*, most cars and airplanes are horizontally symmetric. We also conduct an ablation study evaluating the performance of ConvONet [28] with 5 and 7 pre-defined static planes. The results of this ablation study further verify the superiority of our method. Details are in the supplementary materials.

Similarity loss: To test whether having diverse plane normals that are neither aligned nor flipping to each other can have a significant impact on the model performance, we try another variant where we restrict the learned plane normals to be diverging by adding a pairwise cosine similarity loss among plane normals, as defined below:

$$\mathcal{L}_{similarity} = \frac{1}{M} \sum_{\substack{i,j \\ i \neq j}}^{M} |(\cos(\theta_{i,j}))^d| \tag{6}$$

where $\theta_{i,j}$ is the angle between the pairwise plane normal pair, and M is the total number of pairs. To ensure diverging plane normals, we set $d = 10$ so that the similarity loss starts penalizing when $\theta_{i,j} < 45°$ or $\theta_{i,j} > 135°$. With the additional similarity loss, the loss function is in the following form:

$$\mathcal{L} = \mathcal{L}_{CE} + C \cdot \mathcal{L}_{similarity} \tag{7}$$

where $\mathcal{L}_{CE}$ is the binary cross-entropy loss on the predicted occupancy in Eq. (4). In our experiments, we set C to be $10 \times M$. The component from the similarity loss quickly converges to 0 when the predicted planes become diverse.

With the similarity loss, diverging plane normals are observed. For 3 dynamic planes, the predicted planes are al-

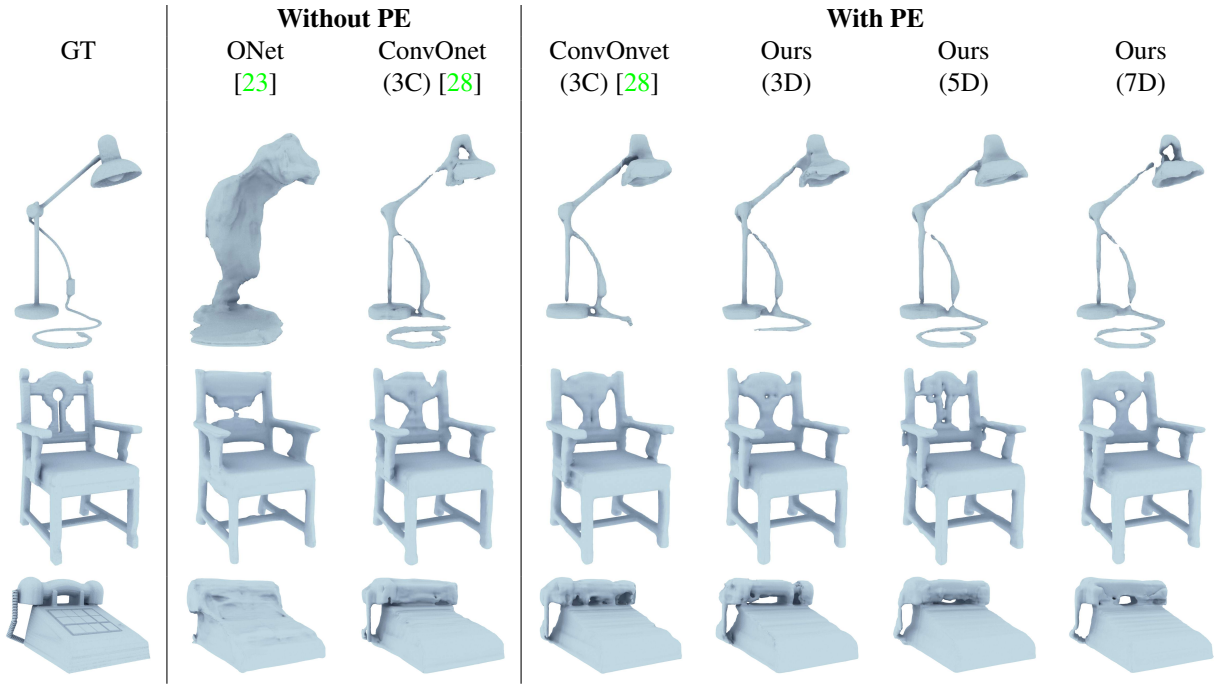
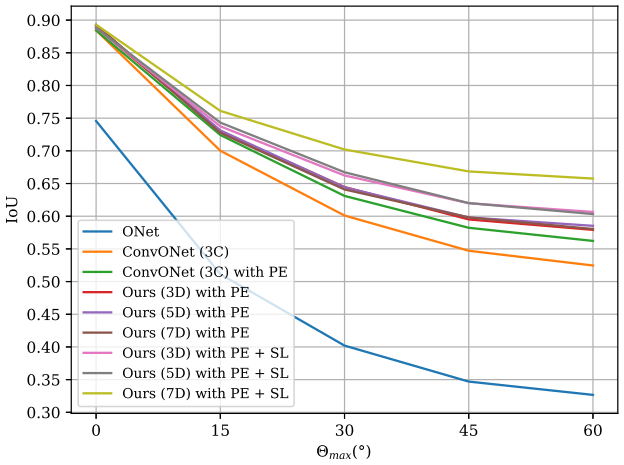


Figure 4: **Object-level 3D reconstruction from point clouds.** Qualitative comparison of our method to ONet [23] and ConvONet [28] on ShapeNet.

most identical sets of planes in canonical axes. Adding more planes, *e.g.* in 5 and 7 dynamic planes, gives predicted planes whose normals diverge from the canonical axes without any flipping or redundant set where slight variations between objects are observed. The plane distributions of the models with 5 and 7 planes trained with similarity loss (both with positional encoding) are illustrated in Fig. 6. We observe that within the plane predictions with slight variations, objects having different global structures favor different regions, corroborating our networks' ability to learn planes that can vary based on the shape of an individual object. In terms of performance, however, we see little or no improvement compared to the unrestricted version. The results with the similarity loss are shown in Table 1.

Interestingly, as shown in Fig. 5, we see that the models trained with similarity loss have better generalization towards inputs with unseen orientations that have not been trained on, especially when using a high number of planes. We test the generalization towards different orientation by applying random rotation to the input in ShapeNet test set along $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ axes with angles θ_x, θ_y, and θ_z drawn uniformly from $[0°, \Theta_{max}]$ for each sample. Figure 5 shows the results of the experiments where the models are trained with all objects in canonical pose and tested on rotated poses. We can clearly notice the progressive drop of IoU when the test set is rotated with random angles up to Θ_{max}.



C = canonical planes. D = dynamic planes. PE = positional encoding. SL = similarity loss.

Figure 5: **Rotation experiment on ShapeNet.** Comparison of IoU on ShapeNet test set rotated with angles uniformly sampled from $[0°, \Theta_{max}]$

It can be seen that there is a considerable generalization improvement when using 7 dynamic planes trained with similarity loss. Comparing [28] with or without positional encoding, we also see that positional encoding improves the generalization towards input in different orientations.

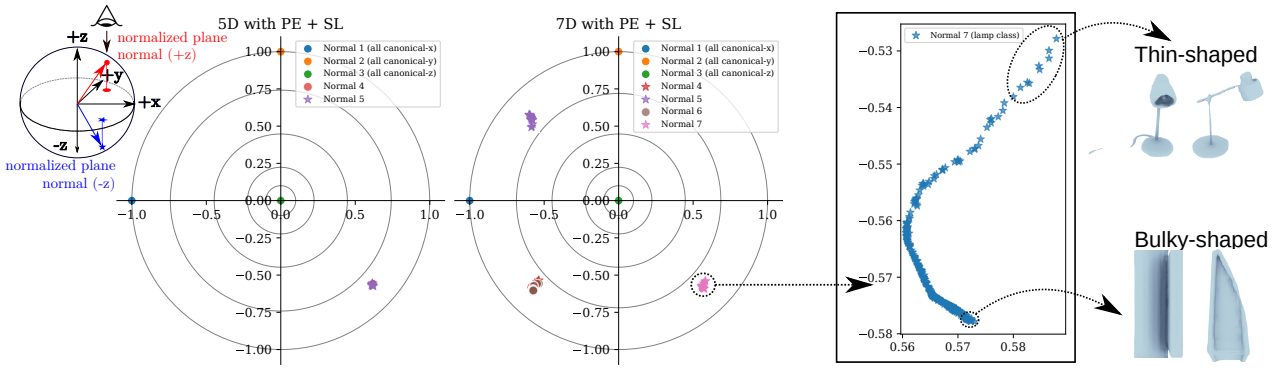


Figure 6: **Plane normal distribution when trained with similarity loss.** We consider 5 and 7 dynamic plane models. In the three figures above, plane normals are normalized into unit lengths and projected as if viewed from the top. "•" indicates the normal has $+z$ direction, while "⋆" indicates $-z$ direction. *Left:* 5 dynamic planes (all classes). *Middle:* 7 dynamic planes (all classes). *Right:* The distribution of one of the plane normals with slight variations of class "lamp" in 7 dynamic planes. It is observed that within this small variation, objects with different global structures favor different regions.

4.2. Scene-Level Reconstruction

For the scene-level experiment, we uniformly sample 10,000 points from the ground truth meshes as input and apply Gaussian noise with a standard deviation of 0.05. During training, we query the occupancy probability of 2048 points. We set the plane resolution 128^2 and use U-Net with a depth of 5. The batch size during training is set to 32 for all experiments with 3 planes, while a batch size of 16 is used for experiments with 5 and 7 planes to accommodate the higher GPU memory requirement. The models are trained for at least 500,000 iterations.

We train our models for synthetic indoor scene dataset by applying the similarity loss (Eq. 6) and disabling after 20,000 iterations, which enables more robust training in our experiments. The reason for doing so is: we find several of our runs without the similarity loss initialization have considerably higher training loss and lower validation score. We observe those models do not predict one of the canonical planes and have planes angled less than $45°$. Our speculation of this occurrence is because the scene dataset has similar global structures of rectangular shapes, it is difficult for our plane predictor networks to recover from bad minimas when the plane prediction is not governed by the similarity loss.

As shown in Table 2, our models achieve better accuracy in all metrics. Moreover, it can be seen from Fig. 7 that our models preserve better fine-grained details than the baseline methods.

	IoU	Chamfer-L_1	Normal C.	F-score
Without PE				
ONet [23]	0.475	0.203	0.783	0.541
ConvONet (3C) [28]	0.789	0.044	0.902	0.950
ConvONet (3C + 32^3 grids) [28]	0.816	0.044	0.905	0.952
Ours (3D)	0.795	0.043	0.907	0.954
Ours (5D)	0.791	0.043	0.905	0.955
Ours (7D)	0.810	**0.042**	0.909	0.957
Ours (3C + 2D)	**0.837**	**0.042**	0.910	0.958
Ours (3C + 4D)	0.831	0.044	0.906	0.953
With PE				
ConvONet (3C) [28]	0.797	0.046	0.902	0.946
Ours (3D)	0.814	**0.042**	0.910	0.958
Ours (5D)	0.800	**0.042**	**0.912**	**0.960**
Ours (7D)	0.819	0.043	0.910	0.957
Ours (3C + 2D)	0.797	0.043	0.908	0.959
Ours (3C + 4D)	0.831	0.043	0.910	0.956

PE = positional encoding. C = canonical planes. D = dynamic planes. SL = similarity loss.

Table 2: **Scene-level reconstruction on synthetic rooms.** Our results on the synthetic indoor scene dataset.

5. Conclusion

In this work, we introduced Dynamic Plane Convolutional Occupancy Networks, a novel implicit representation method for 3D reconstruction from point clouds. We proposed to learn dynamic planes to form informative features. We observe that 3 canonical planes are always predicted, and the symmetric property of objects are implicitly encoded. We also find that enforcing a similarity loss on the predicted plane normals considerably improves the performance on unseen object poses. In future work, we plan to assess the theoretical support for the dynamic plane prediction.

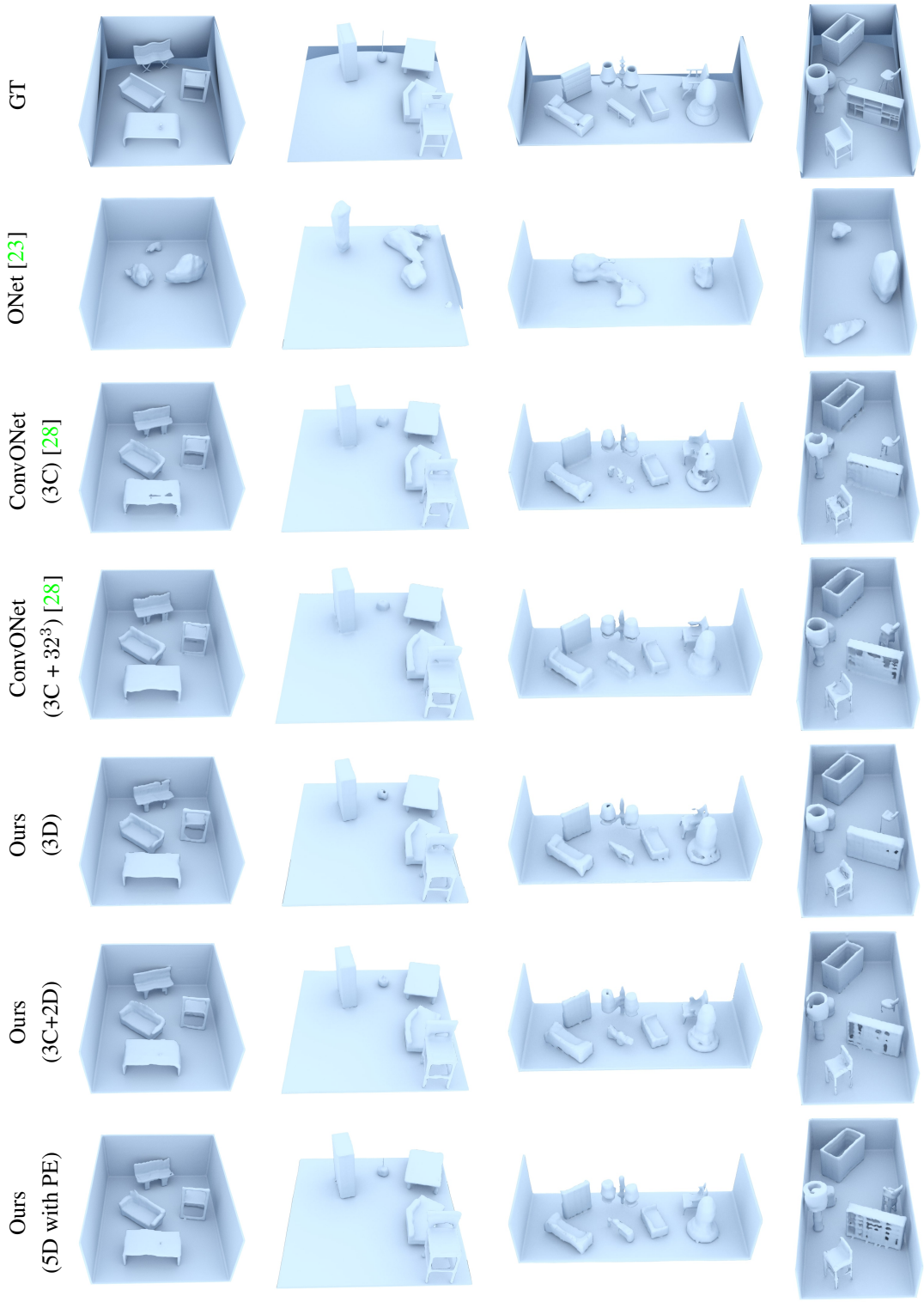


Figure 7: **Scene-level reconstruction on synthetic rooms.** Qualitative comparison of synthetic indoor scene reconstruction from point clouds.

References

[1] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[2] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[3] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[4] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[5] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2016.

[7] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 1999.

[8] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996.

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[10] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[11] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[12] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[14] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[15] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *Proc. of the International Conf. on 3D Vision (3DV)*, 2017.

[17] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[18] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[19] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[20] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[21] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[23] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[24] 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 *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[25] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[26] Michael Oechsle, Lars Mescheder, Michael Niemeyer, Thilo Strauss, and Andreas Geiger. Texture fields: Learning texture representations in function space. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[27] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[28] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[29] Moein Peyghambarzadeh, Fatemeh Azizmalayeri, Hassan Khotanlou, and Amir Salarpour. Point-planenet: Plane kernel based convolutional neural network for point clouds analysis. *Digital Signal Processing*, 2020.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[33] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[34] 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 *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019.

[35] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics (JFR)*, 2008.

[36] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[37] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[40] Yifan Xu, Tianqi Fan, Yi Yuan, and Gurprit Singh. Ladybird: Quasi-monte carlo sampling for deep implicit field based 3d reconstruction with symmetry. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[41] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2017.