PCN: Point Completion Network

Wentao Yuan Tejas Khot David Held Christoph Mertz Martial Hebert

Robotics Institute, Carnegie Mellon University
{wyuan1, tkhot, dheld, cmertz, hebert}@cs.cmu.edu

Abstract

Shape completion, the problem of estimating the complete geometry of objects from partial observations, lies at the core of many vision and robotics applications. In this work, we propose Point Completion Network (PCN), a novel learning-based approach for shape completion. Unlike existing shape completion methods, PCN directly operates on raw point clouds without any structural assumption (e.g. symmetry) or annotation (e.g. semantic class) about the underlying shape. It features a decoder design that enables the generation of fine-grained completions while maintaining a small number of parameters. Our experiments show that PCN produces dense, complete point clouds with realistic structures in the missing regions on inputs with various levels of incompleteness and noise, including cars from LiDAR scans in the KITTI dataset. Code, data and trained models are available at https://www.cs.cmu.edu/~wyuan1/pcn/.

1. Introduction

Real-world 3D data are often incomplete, causing loss in geometric and semantic information. For example, the cars in the LiDAR scan shown in Figure 1 are hardly recognizable due to sparsity of the data points and missing regions caused by limited sensor resolution and occlusion. In this work, we present a novel learning-based method to complete these partial data using an encoder-decoder network that directly maps partial shapes to complete shapes, both represented as 3D point clouds.

Our work is inspired by a number of recent works [9, 48] which leverage large dataset of synthetic shapes to train deep neural networks that can infer the complete geometry from a single or a combination of partial views. However, a key difference between our approach and existing ones is in the representation for 3D data. The majority of existing methods voxelize the 3D data into occupancy grids or distance fields where convolutional networks can be applied. However, the cubically growing memory cost of 3D voxel grids limits the output resolution of these methods. Further, detailed geometry is often lost as an artifact of discretization. In contrast, our network is designed to operate on raw point clouds. This prevents the high memory cost and loss of geometric information caused by voxelization and allows our network to generate more fine-grained completions.

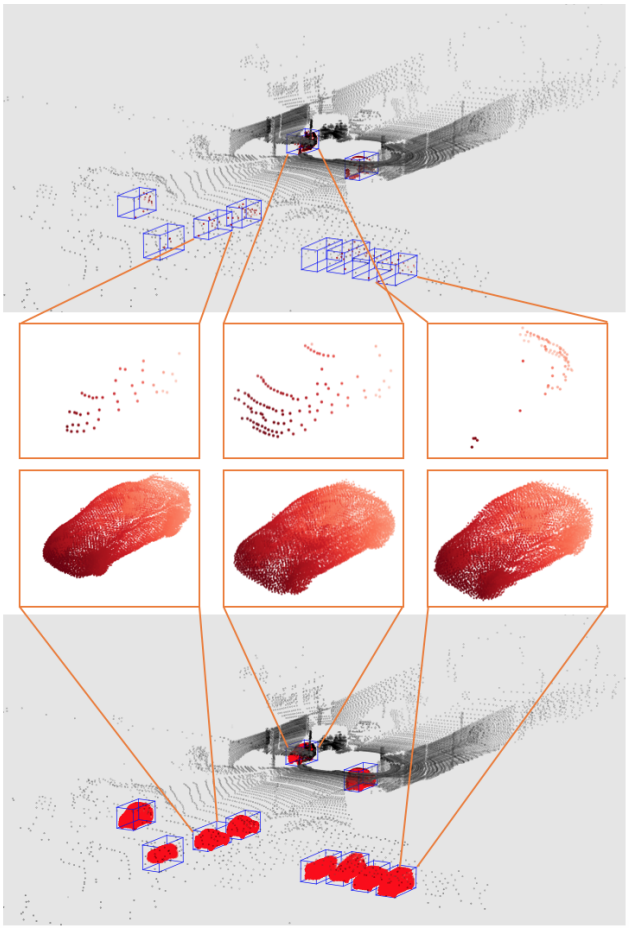


Figure 1: **(Top) Raw LiDAR scan from KITTI** [13]. Note how the cars are barely recognizable due to incompleteness of the data. **(Bottom) Completed scan generated by PCN** on individual car point clouds segmented from the scene.

Designing a network that consumes and generates point clouds involves several challenges. First, a point cloud is

an unordered set, which means permutations of the points do not change the geometry they represent. This necessitates the design of a feature extractor and a loss function that are permutation invariant. Second, there is no clear definition of local neighbourhoods in point clouds, making it difficult to apply any convolutional operation. Lastly, existing point cloud generation networks only generate a small set of points, which is not sufficient to capture enough detail in the output shape. Our proposed model tackles these challenges by combining a permutation invariant, non-convolutional feature extractor and a coarse-to-fine point set generator in a single network that is trained end-to-end.

The main contributions of this work are:

- a learning-based shape completion method that operates directly on 3D point clouds without intermediate voxelization;
- a novel network architecture that generates a dense, complete point cloud in a coarse-to-fine fashion;
- extensive experiments showing improved completion results over strong baselines, robustness against noise and sparsity, generalization to real-world data and how shape completion can aid downstream tasks.

2. Related Work

3D Shape Completion Existing methods for 3D shape completion can be roughly categorized into geometry-based, alignment-based and learning-based approaches.

Geometry-based approaches complete shapes using geometric cues from the partial input without any external data. For example, surface reconstruction methods [3, 10, 33, 41, 49, 54, 62] generate smooth interpolations to fill holes in locally incomplete scans. Symmetry-driven methods [29, 30, 35, 36, 46, 52, 55] identify symmetry axes and repeating regular structures in the partial input in order to copy parts from observed regions to unobserved regions. These approaches assume moderately complete inputs where the geometry of the missing regions can be inferred directly from the observed regions. This assumption does not hold on most incomplete data from the real world.

Alignment-based approaches complete shapes by matching the partial input with template models from a large shape database. Some [15, 24, 32, 34, 43] retrieve the complete shape directly while some [17, 19, 28, 45, 52] retrieve object parts and then assemble them to obtain the complete shape. Other works [6, 12, 14, 21, 22, 39] deform the retrieved model to synthesize shapes that are more consistent with the input. There are also works [7, 25, 31, 42, 61] that use geometric primitives such as planes and quadrics in place of a shape database. These methods require expensive optimization during inference, making them impractical for online applications. They are also sensitive to noise.

Learning-based approaches complete shapes with a parameterized model (often a deep neural network) that directly maps the partial input to a complete shape, which offers fast inference and better generalization. Our method falls into this category. While most existing learning-based methods [9, 16, 44, 47, 50, 54, 56, 59] represents shapes using voxels, which are convenient for convolutional neural networks, our method uses point clouds, which preserve complete geometric information about the shapes while being memory efficient. One recent work [26] also explores deformable meshes as the shape representation. However, their method assumes all the shapes are in correspondence with a common reference shape, which limits its applicability to certain shape categories such as humans or faces.

Deep Learning on Point Clouds Our method is built upon several recent advances in deep neural networks that operates on point clouds. PointNet and its extension [37, 38] is the pioneer in this area and the state-of-the-art while this work was developed. It combines pointwise multi-layer perceptrons with a symmetric aggregation function that achieves invariance to permutation and robustness to perturbation, which are essential for effective feature learning on point clouds. Several alternatives [23, 51, 53, 57, 58] have been proposed since then. Any of these can be incorporated into our proposed model as the encoder.

There are relatively fewer works on decoder networks which generates point sets from learned features. [1] uses a simple fully-connected decoder, while [11] proposes a multi-branch decoder combining fully-connected and deconvolution layers. Recently, [60] introduces an interesting decoder design which mimics the deformation of a 2D plane into a 3D surface. However, none of these methods generates more than 2048 points. Our model combines the advantages of these designs to generate higher resolution outputs in an efficient manner.

3. Problem Statement

Let X be a set of 3D points lying on the observed surfaces of an object obtained by a single observation or a sequence of observations from a 3D sensor. Let Y be a dense set of 3D points uniformly sampled from the observed and unobserved surfaces of the object. We define the shape completion problem as predicting Y given X. Note that under this formulation, X is not necessarily a subset of Y and there is no explicit correspondence between points in X and points in Y, because they are independently sampled from the underlying object surfaces.

We tackle this problem using supervised learning. Leveraging a large-scale synthetic dataset where samples of X and Y can be easily acquired, we train a neural network to predict Y directly from X. The network is generic across multiple object categories and does not assume anything about the structure of underlying objects such symmetry or planarity. The network architecture is described in Section 4 and the training process is described in Section 5.1.

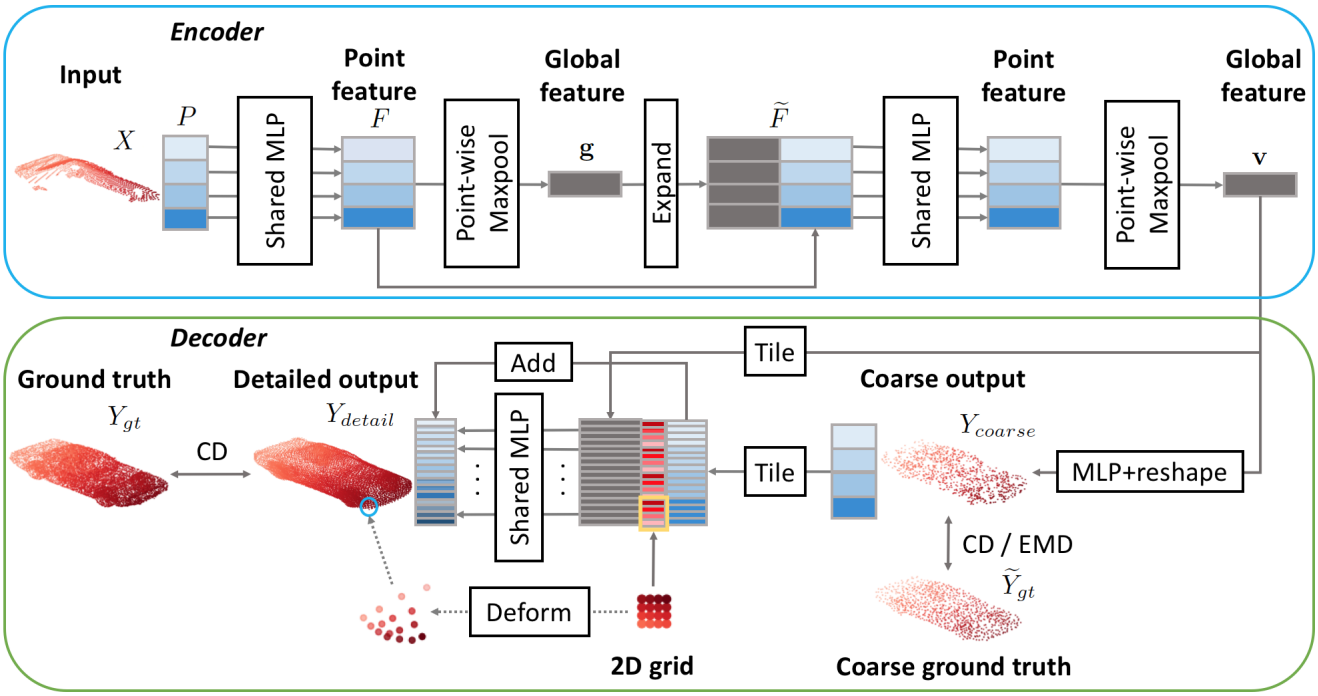


Figure 2: **PCN Architecture**. The encoder abstracts the input point cloud X as a feature vector $\mathbf{v}$. The decoder uses $\mathbf{v}$ to first generate a coarse output Y_{coarse} followed by a detailed output Y_{detail}. Each colored rectangle denotes a matrix row. Same color indicates same content.

4. Point Completion Network

In this section, we describe the architecture of our proposed model, the Point Completion Network (PCN). As shown in Figure 2, PCN is an encoder-decoder network. The encoder takes the input point cloud X and outputs a k-dimensional feature vector. The decoder takes this feature vector and produces a coarse output point cloud Y_{coarse} and a detailed output point cloud Y_{detail}. The loss function L is computed between the ground truth point cloud Y_{gt} and the outputs of the decoder, which is used to train the entire network through backpropagation. Note that, unlike an autoencoder, we do not explicitly enforce the network to retain the input points in its output. Instead, it learns a projection from the space of partial observations to the space of complete shapes. Next, we describe the specific design of the encoder, decoder and the loss function used.

4.1. Point Feature Encoding

The encoder is in charge of summarizing the geometric information in the input point cloud as a feature vector $\mathbf{v} \in \mathbb{R}^k$ where $k = 1024$. Our proposed encoder is an extended version of PointNet [37]. It inherits the invariance to permutation and tolerance to noise from PointNet and can handle inputs with various number of points.

Specifically, the encoder consists of two stacked PointNet (PN) layers. The first layer consumes m input points represented as an $m \times 3$ matrix P where each row is the 3D coordinate of a point $\mathbf{p}_i = (x, y, z)$. A shared multilayer perceptron (MLP) consisting of two linear layers with ReLU activation is used to transform each $\mathbf{p}_i$ into a point feature vector $\mathbf{f}_i$. This gives us a feature matrix F whose rows are the learned point features $\mathbf{f}_i$. Then, a point-wise maxpooling is performed on F to obtain a k-dimensional global feature $\mathbf{g}$, where $\mathbf{g}_j = \max_{i=1,\dots,m}\{F_{ij}\}$ for $j = 1, \dots, k$. The second PN layer takes F and $\mathbf{g}$ as input. It first concatenates $\mathbf{g}$ to each $\mathbf{f}_i$ to obtain an augmented point feature matrix $\widetilde{F}$ whose rows are the concatenated feature vectors $[\mathbf{f}_i\ \mathbf{g}]$. Then, $\widetilde{F}$ is passed through another shared MLP and point-wise max pooling similar to the ones in the first layer, which gives the final feature vector $\mathbf{v}$.

4.2. Multistage Point Generation

The decoder is responsible for generating the output point cloud from the feature vector $\mathbf{v}$. Our proposed decoder combines the advantages of the fully-connected decoder [1] and the folding-based decoder [60] in a multistage point generation pipeline. In our experiments, we show that our decoder outperforms either the fully-connected or the folding-based decoder on its own.

Our key observation is that the fully-connected decoder is good at predicting a sparse set of points which represents the global geometry of a shape. Meanwhile, the folding-based decoder is good at approximating a smooth surface

which represents the local geometry of a shape. Thus, we divide the generation of the output point cloud into two stages. In the first stage, a coarse output Y_{coarse} of s points is generated by passing $\mathbf{v}$ through a fully-connected network with $3s$ output units and reshaping the output into a $s \times 3$ matrix. In the second stage, for each point $\mathbf{q}_i$ in Y_{coarse}, a patch of $t = u^2$ points is generated in the local coordinates centered at $\mathbf{q}_i$ via the folding operation (refer to Section B in the supplementary for details), and transformed into the global coordinates by adding $\mathbf{q}_i$ to the output. Combining all s patches gives the detailed output Y_{detail} consisting of $n = st$ points. This multistage process allows our network to generate a dense output point cloud with fewer parameters than fully-connected decoder (see Table 1) and more flexibility than folding-based decoder.

4.3. Loss Function

The loss function measures the difference between the output point cloud and the ground truth point cloud. Since both point clouds are unordered, the loss needs to be invariant to permutations of the points. Two candidates of permutation invariant functions are introduced by [11] – Chamfer Distance (CD) and Earth Mover's Distance (EMD).

$$\begin{aligned} CD(S_1, S_2) = &\frac{1}{|S_1|} \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2 \\ &+ \frac{1}{|S_2|} \sum_{y \in S_2} \min_{x \in S_1} \|y - x\|_2 \end{aligned} \tag{1}$$

CD (1) calculates the average closest point distance between the output point cloud S_1 and the ground truth point cloud S_2. We use the symmetric version of CD where the first term forces output points to lie close to ground truth points and the second term ensures the ground truth point cloud is covered by the output point cloud. Note that S_1 and S_2 need not be the same size to calculate CD.

$$EMD(S_1, S_2) = \min_{\phi: S_1 \to S_2} \frac{1}{|S_1|} \sum_{x \in S_1} \|x - \phi(x)\|_2 \tag{2}$$

EMD (2) finds a bijection $\phi : S_1 \to S_2$ which minimizes the average distance between corresponding points. In practice, finding the optimal ϕ is too expensive, so we use an iterative $(1 + \epsilon)$ approximation scheme [4]. Unlike CD, EMD requires S_1 and S_2 to be the same size.

$$\begin{aligned} L(Y_{coarse}, Y_{detail}, Y_{gt}) = &\, d_1(Y_{coarse}, \widetilde{Y}_{gt}) \\ &+ \alpha \, d_2(Y_{detail}, Y_{gt}) \end{aligned} \tag{3}$$

Our proposed loss function (3) consists of two terms, d_1 and d_2, weighted by hyperparameter α. The first term is the distance between the coarse output Y_{coarse} and the subsampled ground truth $\widetilde{Y}_{gt}$ which has the same size as Y_{coarse}. The second term is the distance between the detailed output Y_{detail} and the full ground truth Y_{gt}.

In our experiments, we use both CD and EMD for d_1 but only CD for d_2. This is because the $O(n^2)$ complexity of the EMD approximation scheme makes it too expensive to compute during training when n is large, while CD can be computed with $O(n \log n)$ complexity using efficient data structure for nearest neighbour search such as KDTree.

5. Experiments

In this section, we first describe the creation of a large-scale, multi-category dataset to train our model. Next, we compare our method against existing methods and ablated versions of our method on synthetic shapes. Finally, we show completion results on real-world point clouds and demonstrate how they can help downstream tasks such as point cloud registration.

5.1. Data Generation and Model Training

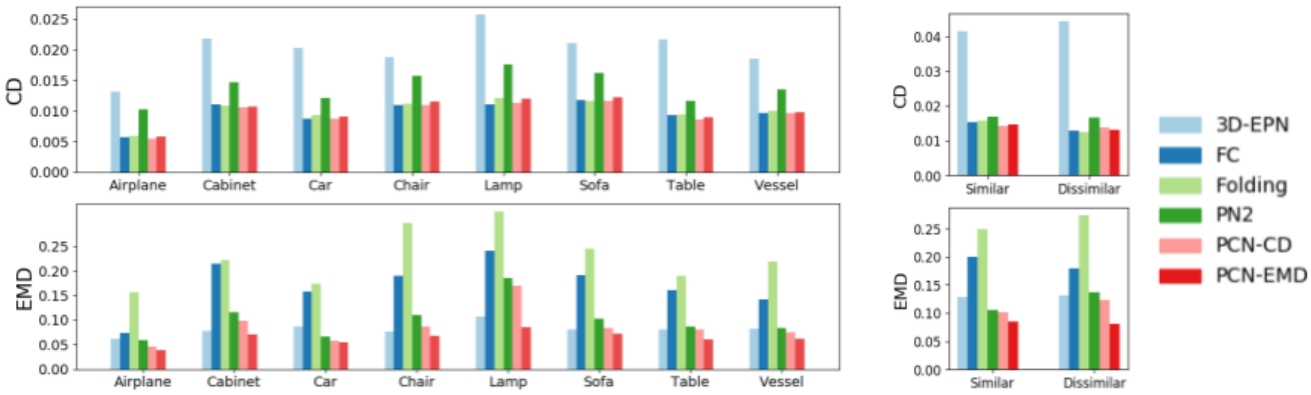
To train our model, we use synthetic CAD models from ShapeNet to create a large-scale dataset containing pairs of partial and complete point clouds (X, Y). Specifically, we take 30974 models from 8 categories: airplane, cabinet, car, chair, lamp, sofa, table, vessel. The complete point clouds are created by sampling 16384 points uniformly on the mesh surfaces and the partial point clouds are generated by back-projecting 2.5D depth images into 3D. We use back-projected depth images for partial inputs instead of subsets of the complete point cloud in order to bring the input distribution closer to real-world sensor data. For each model, 8 partial point clouds are generated from 8 randomly distributed viewpoints. Note that the partial point clouds can have different sizes.

We choose to use a synthetic dataset to generate training data because it contains complete, detailed 3D models of objects that are not available in real-world datasets. Despite the fact that recent datasets such as ScanNet [8] or S3DIS [2] have very high quality 3D reconstructions, these reconstructions have missing regions due to the limitations of the scanner's view, and thus are not good enough to use as ground truth for our model.

We reserve 100 models for validation and 150 models for testing. The rest is used for training. All our models are trained using the Adam [20] optimizer with an initial learning rate of 0.0001 for 50 epochs and a batch size of 32. The learning rate is decayed by 0.7 every 50K iterations.

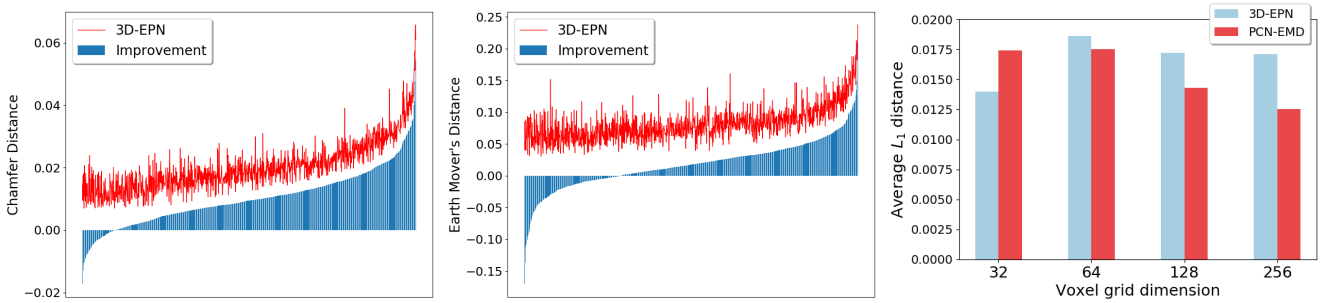
5.2. Completion Results on ShapeNet

In this subsection, we compare our method against several strong baselines, including a representative volumetric network and modified versions of our model, on synthetic point clouds from ShapeNet. We also test the generalizability of these methods to novel shapes and the robustness of our model against occlusion and noise.



(a) Results on trained categories (b) Results on novel categories

Figure 3: **Quantitative comparison on ShapeNet**. For both CD (top) and EMD (below), lower is better. (a) shows results for test instances from the same categories used in training. (b) shows results for test instances from categories not included in training, which are divided into similar (bus, bed, bookshelf, bench) and dissimilar (guitar, motorbike, skateboard, pistol).



(a) Chamfer Distance (CD) (b) Earth Mover's Distance (EMD) (c) L_1 distance

Figure 4: **Comparison between PCN-EMD and 3D-EPN**. (a) and (b) shows comparison of point cloud outputs. The x-axis represents different object instances. The height of the blue bar indicates the amount of improvement of PCN-EMD over 3D-EPN. The red curve is the error of 3D-EPN and the difference between the red curve and the blue bar is the error of PCN-EMD. PCN-EMD improves on the majority of instances. (c) shows comparison of distance field outputs. On the y-axis is the average L_1 distance on occluded voxels between output and ground truth distance fields. PCN-EMD achieves lower L_1 distance on higher resolutions.

Baselines Previous point-based completion methods either assume more complete inputs than we have [18] or prior knowledge of the shape such as semantic class, symmetry and part segmentation [52], and thus are not directly comparable to our method. Here, we compare our model against four strong baselines which, like our method, work on objects from multiple categories with different levels of incompleteness.

1) **3D-EPN** [9]: This is a representative of the class of volumetric completion methods that is also trained end-to-end on large synthetic dataset. To compare the distance field outputs of 3D-EPN with the point cloud outputs of PCN, we convert the distance fields into point clouds by extracting the isosurface at a small value d and uniformly sampling 16384 points on the resulting mesh. To ensure fair comparison, we also convert the point cloud outputs of PCN into distance fields by calculating the distance from grid centers to the closest point in the output.

2) **FC**: This is a network that uses the same encoder as PCN but the decoder is a 3-layer fully-connected network which directly outputs the coordinates of 16384 points.

3) **Folding**: This is a network that also uses the same encoder as PCN but the decoder is purely folding-based [60], which deforms a 128-by-128 2D grid into a 3D point cloud.

4) **PN2**: This is a network that uses the same decoder as our proposed model but the encoder is PointNet++ [38].

We provide two versions of our model for comparison, PCN-CD and PCN-EMD. The number of points in the coarse and detailed outputs are $s = 1024$ and $n = 16384$ respectively. For the loss on coarse output, PCN-CD uses CD and PCN-EMD uses EMD. Note that both models use CD for the loss on detailed output due to the computational complexity of EMD.

Test Set We created two test sets: one consists of 150 reserved shapes from the 8 object categories on which the models are trained; the other consists of 150 models from 8 novel categories that are not in the training set. We divide the novel categories into two groups: one that is visually similar to the training categories – bed, bench, bookshelf and bus, and another that is visually dissimilar to the training categories – guitar, motorbike, pistol and skateboard. The quantitative comparisons are shown in Figure 3 and some qualitative examples are shown in Figure 9.

Metrics The metrics we use on point clouds are CD and EMD between the output and ground truth point clouds, as defined in 4.3. An illustration of the difference between the two metrics is shown in Figure 5. We can see that CD is high where the global structure is different, e.g. around the corners of the chair back. On the other hand, EMD is more evenly distributed, as it penalizes density difference between the two point clouds. Note that on average, EMD is much higher than CD. This is because EMD requires one-to-one correspondences between the points, whereas the point correspondences used by CD can be one-to-many.

The metric we use on distance fields is the L_1 distance between the output and ground truth distance fields, same as in [8]. To have comparable numbers across different dimensions, we convert the error from the voxel distance to distance in the model's metric space.

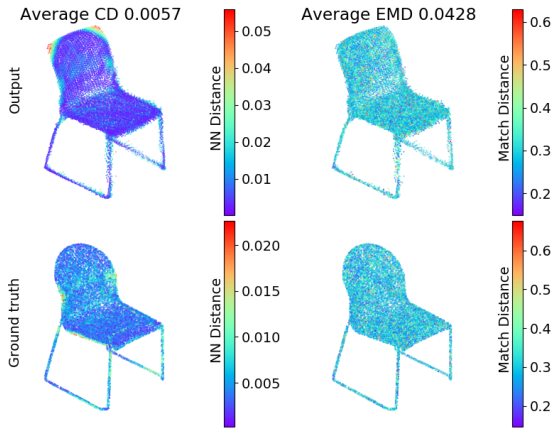


Figure 5: **Illustration of CD (left) and EMD (right)**. The top row shows the output of our model and the bottom row shows the ground truth. The points on the left are colored by their distance to the closest point in the other point cloud (nearest neighbor (NN) distance). The points on the right are colored by their distance to the corresponding point under the optimal bijection (match distance). Average CD is the mean of the NN distances and average EMD is the mean of the match distances.

Generalizability to Novel Objects As shown in Figure 3b, our method outperforms all baselines on object from novel categories. More importantly, our model's performance is not significantly affected even on visually dissimilar categories (e.g. the pistol in Figure 9). This shows the generality of the shape prior learned by our model.

Comparison to Volumetric Method It can be seen that our method outperforms 3D-EPN by a large margin on both CD and EMD. To better interpret the numbers, in Figures 4a, 4b, we show the amount of improvement of our completion results over that of 3D-EPN on CD and EMD for each instance in the test set. The results of our method improve on the majority of instances. Further, they improve the most on examples where the error of 3D-EPN is high, indicating its ability to handle challenging cases where previous methods fail.

In Figure 4c, we show that our method achieves lower L_1 distance when its outputs are converted to a distance field. Moreover, the improvement of our method over 3D-EPN is more significant at higher resolutions.

Decoder Comparison The results in Figure 3 show how our proposed decoder compares with existing decoder designs. Our multistage design which combines the advantages of fully-connected and folding-based decoders outperforms either design on its own. From the qualitative results, we observe that the fully-connected decoder does not have any constraints on the local density of the output points, and thus the output points are often over-concentrated in areas such as table top, which results in high EMD. On the other hand, the folding-based decoder often produces points that are floating in space and not consistent with the global geometry of the ground truth shape, which results in high CD. This is because the shapes in our dataset contain many concavities and sharp edges, which makes globally folding a 2D plane into a 3D shape very challenging. FoldingNet [60] addresses this by chaining two folding operations. However, by only doing the folding operation locally, our decoder is able to achieve better performance with only one folding operation.

Encoder Comparison Another pair of comparison shown in Figure 3 is between the stacked PN and PN2 [38] as the encoder. PN2 is a representative of the class of hierarchical feature extraction networks that aggregate local information before global pooling. Our results show that it is outperformed by our stacked PN encoder which uses only global pooling. We believe this is because local pooling is less stable than global pooling due to suboptimality in the selection of local neighbourhoods for the partial data we are dealing with. Thus, we argue that stacking PN layers instead of doing local pooling is a better way of mixing local and global information.

Number of Parameters As shown in Table 1, our model has an order of magnitude fewer parameters than 3D-EPN and FC while achieving significantly better performance.

Table 1: Number of trainable model parameters

Method	3D-EPN	FC	Folding	PN2	Ours
# Params	52.4M	53.2M	2.40M	6.79M	6.85M

Robustness to occlusion and noise Now, we test the robustness of our method to sensor noise and large occlusions. Specifically, we perturbed the depth map with Gaussian noise whose standard deviation is 0.01 times the scale of the depth measurements, and occluded it with a mask that covers p percent of points, where p ranges from 0% to 80%. Additionally, we randomly set 1% of the measurements to $d_{max} = 1.6$.

As we can see from Figure 6, the errors (CD and EMD) increase only gradually as more and more regions are occluded. Note that our model is *not* trained with these occluded and noisy examples, but it is still robust to them. The strong shape prior that the model has learned helps it to ignore the noisy points and predict reasonable outputs under occlusions. This in part explains its strong generalizability to real-world data, as we will show in the following section.

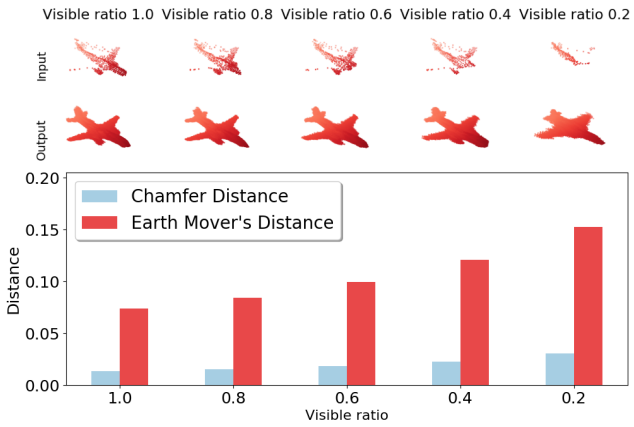


Figure 6: Qualitative (top) and quantitative (bottom) results on noisy inputs with different level of visibility

5.3. Completion Results on KITTI

In this experiment, we test our completion method on partial point clouds from real-world LiDAR scans. Specifically, we take a sequence of Velodyne scans from the KITTI dataset [13]. For each frame, we extract points within the object bounding boxes labeled as cars, which results in 2483 partial point clouds. Each point cloud is then transformed to the box's coordinates, completed with a PCN trained on cars from ShapeNet, and transformed back to the world frame. The process is illustrated in Figure 1. We use a model trained specifically on cars here to incorporate prior knowledge of the object class. Having such prior knowledge is not necessary for our method but will help the model achieve better performance.

Table 2: Quantitative results on KITTI.

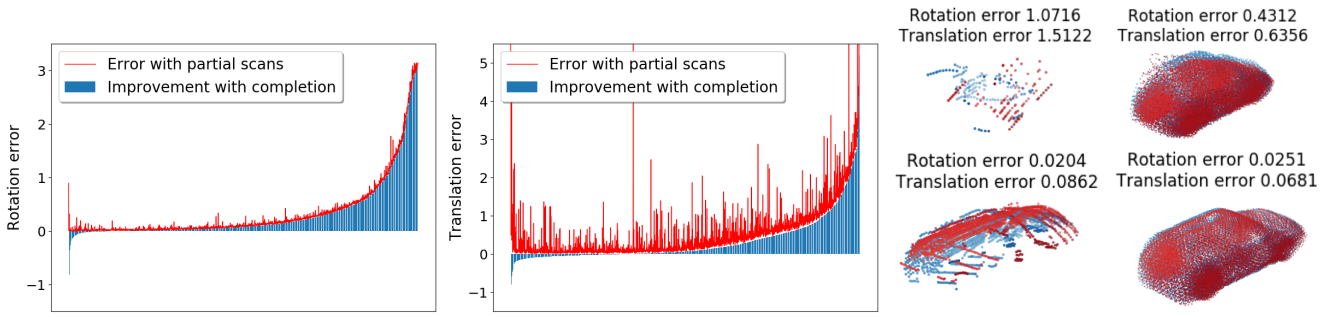
Fidelity	MMD	Consistency	Consistency (input)
0.02800	0.01850	0.01163	0.05121

We do not have the complete ground truth point clouds for KITTI. Thus, we propose three alternative metrics to evaluate the performance of our model: 1) Fidelity, which is the average distance from each point in the input to its nearest neighbour in the output. This measures how well the input is preserved; 2) Minimal Matching Distance (MMD), which is the Chamfer Distance (CD) between the output and the car point cloud from ShapeNet that is closest to the output point cloud in terms of CD. This measures how much the output resembles a typical car; 3) Consistency, which is the average CD between the completion outputs of the same instance in consecutive frames. This measures how consistent the network's outputs are against variations in the inputs. As a comparison, we also compute the average CD between the inputs in consecutive frames, denoted as Consistency (input). These metrics are reported in Table 2.

Unlike point clouds back-projected from 2.5D images, point clouds from LiDAR scans are very sparse. The 2483 partial point clouds here contain 440 points on average, with some having fewer than 10 points. In contrast, point clouds from 2.5D images used in training usually contain more than 1000 points. In spite of this, our model is able to transfer easily between the two distributions without any fine tuning, producing consistent completions from extremely partial inputs. This can be attributed to the use of point-based representation, which is less sensitive to input density than volumetric representations. In addition, each prediction with our model takes only 0.0012s on a Nvidia GeForce 1080Ti GPU and 2s on a 3.60GHz Intel Core i7-7700 CPU, making it suitable for real-time applications.

5.4. Point Cloud Registration with Completion

Many common tasks on point clouds can benefit from a more complete and denser input. Here, as an example of such applications, we show that the output of our network can improve the results of point cloud registration. Specifically, we perform registration between car point clouds from neighboring frames in the same Velodyne sequence from Section 5.3, using a simple point-to-point ICP [5] algorithm implemented in PCL [40]. This results in 2396 registration instances. We provide two kinds of inputs to the registration algorithm – one is partial point clouds from the original scans, another is completed point clouds by a



(a) Rotation error (b) Translation error (c) Qualitative example

Figure 7: **Improvements on point cloud registration**. In (a) and (b), the x-axis represents different registration instances. The height of the blue bar indicates the amount of improvement of registration with complete point clouds over registration with partial point clouds. The red curve is the error of registration with partial point clouds and the difference between the red curve and the blue bar is the error of registration with complete point clouds. In (c), registered partial point clouds are shown on the left and registered complete point clouds of the same instances are shown on the right.

PCN trained on cars from ShapeNet. We compare the rotational and translational error on the registration results with partial and complete inputs. The rotational error is computed as $2\cos^{-1}(2\langle q_1, q_2\rangle^2 - 1)$, where q_1 and q_2 are the quaternion representations of the ground truth rotation and the rotation computed by ICP. This measures the angle between q_1 and q_2. The translational error is computed as $\|t_1 - t_2\|_2$, where t_1 is the ground truth translation and t_2 is the translation computed by ICP.

As shown in Figure 7a and 7b, both rotation and translation estimations are more accurate with complete point clouds produced by PCN, and the improvement is most significant when the error with partial point clouds is large. Figure 7c shows some qualitative examples. As can be seen, the complete point clouds are much easier to register because they contain larger overlaps, a lot of which are regions completed by PCN. Note that the improvement brought by our completion results is not specific to ICP, but can be applied to any registration algorithm.

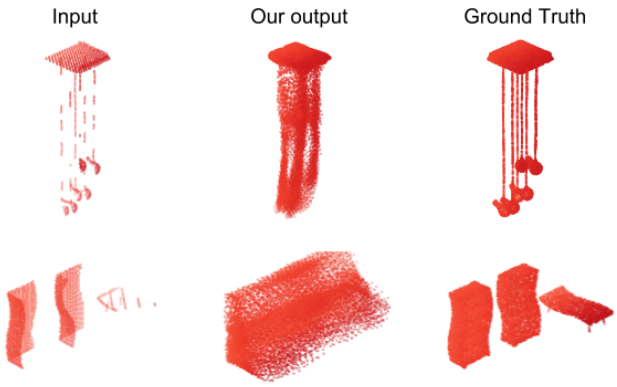


Figure 8: **Failure modes**: thin structures (top) and disconnected parts (bottom).

6. Discussion

We have identified two prominent failure modes for our model. First, there are some object instances consisting of multiple disconnected parts. Our model fails to recognize this and incorrectly connects the parts. This is likely a result of the strong priors learned from the training dataset where almost all objects are connected. Second, some objects contain very thin structures such as wires. Our model is occasionally unable to recover these structures. There are two possible reasons. First, the points from these structures are often sparse since they have small surface areas, which makes the 3D feature extraction more difficult. Second, unlike most object surfaces the local geometry of thin structures does not resemble the 2D grid, making it challenging for our model to deform a 2D grid into these thin structures. Some visualizations of these failures are shown in Figure 8.

7. Conclusion

We have presented a new approach to shape completion using point clouds without any voxelization. To this end, we have designed a deep learning architecture which combines advantages from existing architectures to generate a dense point cloud in a coarse-to-fine fashion, enabling high resolution completion with much fewer parameters than voxel-based models. Our method is effective across multiple object categories and works with inputs from different sensors. In addition, it shows strong generalization performance on unseen objects and real-world data. Our point-based completion method is more scalable and robust than voxel-based methods, which makes it a better candidate for completion of more complex data such as scenes.

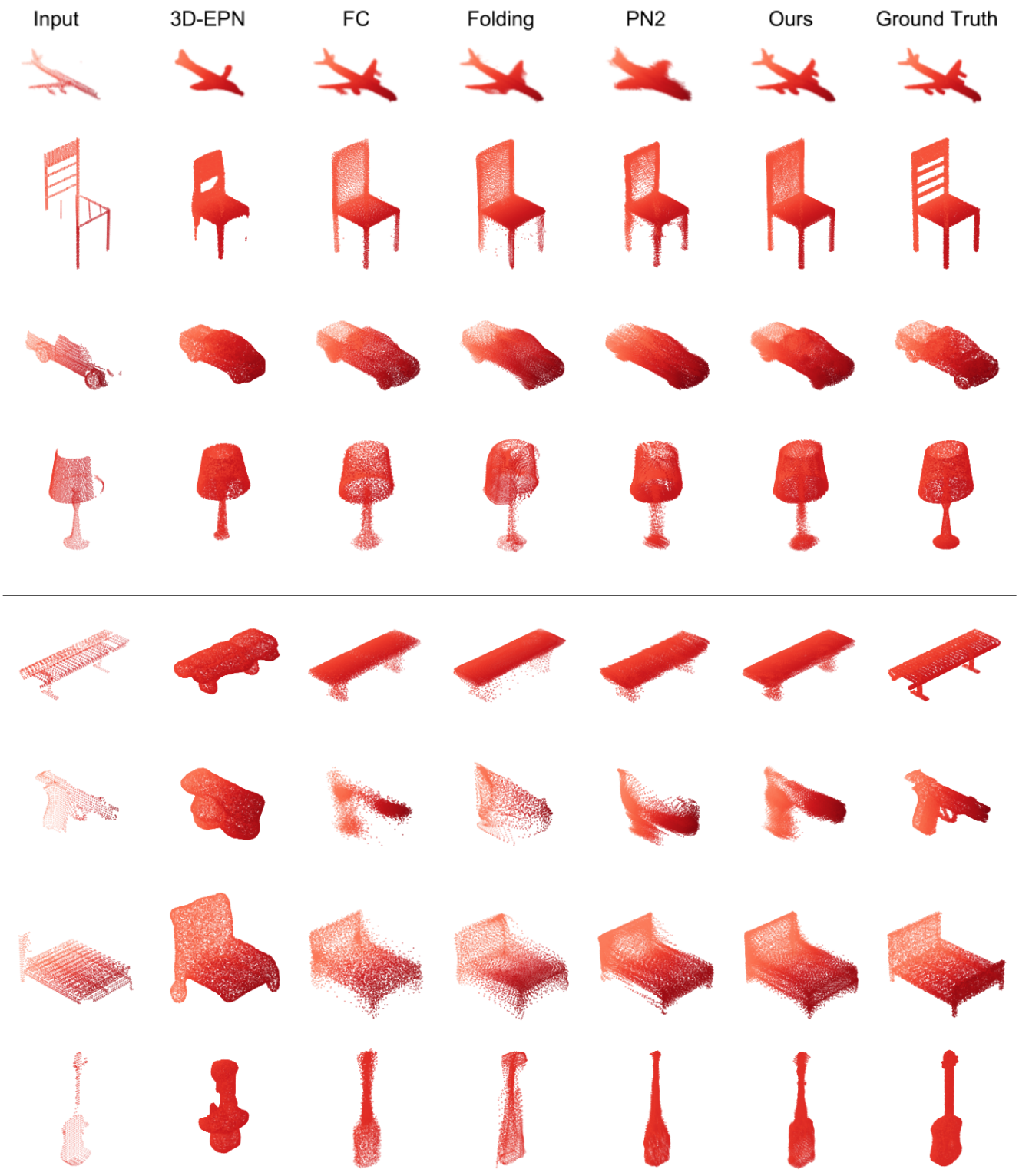


Figure 9: **Qualitative completion results on ShapeNet**. Top four rows are results on categories used during training. Bottom four rows are results on categories not seen during training.

Acknowledgements

This project is supported by Carnegie Mellon University's Mobility21 National University Transportation Center, which is sponsored by the US Department of Transportation. We would also like to thank Adam Harley, Leonid Keselman and Rui Zhu for their helpful comments and suggestions.

References

[1] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017. 2, 3

[2] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 4

[3] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva. State of the art in surface reconstruction from point clouds. In *EUROGRAPHICS star reports*, volume 1, pages 161–185, 2014. 2

[4] D. P. Bertsekas. A distributed asynchronous relaxation algorithm for the assignment problem. In *Decision and Control, 1985 24th IEEE Conference on*, pages 1703–1704. IEEE, 1985. 4

[5] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 7

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999. 2

[7] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1261–1268. IEEE, 2010. 2

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 1, 2017. 4, 6

[9] A. Dai, C. R. Qi, and M. Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 1, 2, 5, 12

[10] J. Davis, S. R. Marschner, M. Garr, and M. Levoy. Filling holes in complex surfaces using volumetric diffusion. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 428–441. IEEE, 2002. 2

[11] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 38, 2017. 2, 4

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010. 2

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 7

[14] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740, 2015. 2

[15] F. Han and S.-C. Zhu. Bottom-up/top-down image parsing with attribute grammar. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(1):59–73, 2009. 2

[16] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu. High-resolution shape completion using deep neural networks for global structure and local geometry inference. *arXiv preprint arXiv:1709.07599*, 2017. 2

[17] E. Kalogerakis, S. Chaudhuri, D. Koller, and V. Koltun. A probabilistic model for component-based shape synthesis. *ACM Transactions on Graphics (TOG)*, 31(4):55, 2012. 2

[18] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):29, 2013. 5

[19] V. G. Kim, W. Li, N. J. Mitra, S. Chaudhuri, S. DiVerdi, and T. Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013. 2

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[21] D. Li, T. Shao, H. Wu, and K. Zhou. Shape completion from a single rgbd image. *IEEE transactions on visualization and computer graphics*, 23(7):1809–1822, 2017. 2

[22] G. Li, L. Liu, H. Zheng, and N. J. Mitra. Analysis, reconstruction and manipulation using arterial snakes. *ACM SIGGRAPH Asia 2010 papers on-SIGGRAPH ASIA'10*, 2010. 2

[23] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn. *arXiv preprint arXiv:1801.07791*, 2018. 2

[24] Y. Li, A. Dai, L. Guibas, and M. Nießner. Database-assisted object retrieval for real-time 3d reconstruction. In *Computer Graphics Forum*, volume 34, pages 435–446. Wiley Online Library, 2015. 2

[25] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. In *ACM Transactions on Graphics (TOG)*, volume 30, page 52. ACM, 2011. 2

[26] O. Litany, A. Bronstein, M. Bronstein, and A. Makadia. Deformable shape completion with graph convolutional autoencoders. *arXiv preprint arXiv:1712.00268*, 2017. 2

[27] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 14

[28] A. Martinovic and L. Van Gool. Bayesian grammar learning for inverse procedural modeling. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 201–208. IEEE, 2013. 2

[29] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006. 2

[30] N. J. Mitra, M. Pauly, M. Wand, and D. Ceylan. Symmetry in 3d geometry: Extraction and applications. In *Computer*

Graphics Forum, volume 32, pages 1–23. Wiley Online Library, 2013. 2

[31] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Transactions on Graphics (TOG)*, 29(4):93, 2010. 2

[32] L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understanding. *ACM Transactions on Graphics (TOG)*, 31(6):137, 2012. 2

[33] A. Nealen, T. Igarashi, O. Sorkine, and M. Alexa. Laplacian mesh optimization. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 381–389. ACM, 2006. 2

[34] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas. Example-based 3d scan completion. In *Symposium on Geometry Processing*, number EPFL-CONF-149337, pages 23–32, 2005. 2

[35] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3d geometry. *ACM transactions on graphics (TOG)*, 27(3):43, 2008. 2

[36] J. Podolak, P. Shilane, A. Golovinskiy, S. Rusinkiewicz, and T. Funkhouser. A planar-reflective symmetry transform for 3d shapes. *ACM Transactions on Graphics (TOG)*, 25(3):549–559, 2006. 2

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 2, 3, 14

[38] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017. 2, 5, 6, 12

[39] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2484–2493, 2015. 2

[40] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011. 7

[41] K. Sarkar, K. Varanasi, and D. Stricker. Learning quadrangulated patches for 3d shape parameterization and completion. *arXiv preprint arXiv:1709.06868*, 2017. 2

[42] R. Schnabel, P. Degener, and R. Klein. Completion and reconstruction with primitive shapes. In *Computer Graphics Forum*, volume 28, pages 503–512. Wiley Online Library, 2009. 2

[43] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012. 2

[44] A. Sharma, O. Grau, and M. Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision*, pages 236–250. Springer, 2016. 2

[45] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 31(6):180, 2012. 2

[46] I. Sipiran, R. Gregor, and T. Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014. 2

[47] E. Smith and D. Meger. Improved adversarial systems for 3d object generation and reconstruction. *arXiv preprint arXiv:1707.09557*, 2017. 2

[48] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 190–198. IEEE, 2017. 1

[49] O. Sorkine and D. Cohen-Or. Least-squares meshes. In *Shape Modeling Applications, 2004. Proceedings*, pages 191–199. IEEE, 2004. 2

[50] D. Stutz and A. Geiger. Learning 3d shape completion from laser scan data with weak supervision. 2

[51] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 2

[52] M. Sung, V. G. Kim, R. Angst, and L. Guibas. Data-driven structural priors for shape completion. *ACM Transactions on Graphics (TOG)*, 34(6):175, 2015. 2, 5

[53] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2

[54] D. Thanh Nguyen, B.-S. Hua, K. Tran, Q.-H. Pham, and S.-K. Yeung. A field model for repairing 3d shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5676–5684, 2016. 2

[55] S. Thrun and B. Wegbreit. Shape from symmetry. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1824–1831. IEEE, 2005. 2

[56] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2442–2447. IEEE, 2017. 2

[57] S. Wang, S. Suo, W.-C. M. A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2

[58] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 2

[59] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen. 3d object dense reconstruction from a single depth view. *arXiv preprint arXiv:1802.00411*, 2018. 2

[60] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *arXiv preprint arXiv:1712.07262*, 2017. 2, 3, 5, 6, 12

[61] K. Yin, H. Huang, H. Zhang, M. Gong, D. Cohen-Or, and B. Chen. Morfit: interactive surface reconstruction from incomplete point clouds with curve-driven topology and geometry control. *ACM Trans. Graph.*, 33(6):202–1, 2014. 2

[62] W. Zhao, S. Gao, and H. Lin. A robust hole-filling algorithm for triangular mesh. *The Visual Computer*, 23(12):987–997, 2007. 2

A. Overview

In this document we provide technical details and additional quantitative and qualitative results to the main paper.

In Section B, we describe the local folding operation in detail. Section C provides specific parameters for the models compared in Section 5.2. Section D and E present more results on ShapeNet and KITTI, including failure cases. Section F provides further analysis on the network design. Section G shows more visualization results.

B. Local Folding Operation

Here we describe the local folding operation mentioned in Section 4.2 in detail. As shown in Figure 10, the folding operation takes a point $\mathbf{q}_i$ in the coarse output Y_{coarse} and the k-dimensional global feature $\mathbf{v}$ as inputs, and generates a patch of $t = u^2$ points in local coordinates centered at $\mathbf{q}_i$ by deforming a $u \times u$ grid. It first takes points on a zero-centered $u \times u$ grid with side length r (r controls the scale of the output patch) and organize their coordinates into a $t \times 2$ matrix G. Then, it concatenates each row of G with the coordinates of the center point $\mathbf{q}_i$ and the global feature vector $\mathbf{v}$, and passes the resulting matrix through a shared MLP that generates a $t \times 3$ matrix Q, i.e. the local patch centered at $\mathbf{q}_i$. This shared MLP can be interpreted as a non-linear transformation that deforms the 2D grid into a smooth 2D manifold in 3D space. Note that the same MLP is used in the local patch generation for each $\mathbf{q}_i$ so the number of parameters in the local folding operation does not grow with the output size.

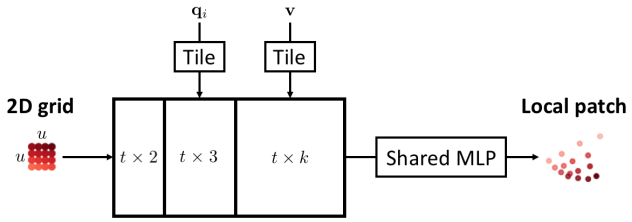


Figure 10: The folding operation

C. Network Architecture Details

Here we describe the detailed architecture of the models compared in Section 5.2. The 3D-EPN model we used is 3D-EPN-unet-class, the best performing model from [9]. For the seen categories, we use the 128^3 output which involves an additional database retrieval step. For the unseen categories, we use the original 32^3 output from the model.

The stacked PN encoder used by FC, Folding, PCN-CD and PCN-EMD includes 2 PN layers. The shared MLP in the first PN layer has 2 layers with 128 and 256 units. The shared MLP in the second PN layer has 2 layers with 512 and 1024 units. The PN2 encoder follows the same architecture as the SSG network in [38].

The FC decoder contains 3 fully-connected layers with 1024, 1024 and $16384 \cdot 3$ units. The Folding decoder contains 2 folding layers as in [60], where the second layer takes the output of the first layer instead of a 2D grid. Note that these folding layers are different from the one described in Section B in that they do not take the center point coordinates as input. Each folding layer has a 3-layer shared MLP with 512, 512 and 3 units. The grid size is $u = 128$ and the grid scale is $r = 0.5$.

The multistage decoder in PCN-CD and PCN-EMD has 3 fully-connected layers with 1024, 1024 and $1024 \cdot 3$ units, followed by 1 folding layer as described in Section B, where the grid size is $u = 4$ and the grid scale is $r = 0.05$. The folding layer contains a 3-layer shared MLP with 512, 512 and 3 units.

D. Additional Results on ShapeNet

Table 4, 5, 6 and 7 show the quantitative results on test instances from ShapeNet corresponding to Figure 3 in the main paper. Figure 9 shows the qualitative comparisons on shapes from seen as well as unseen categories. As can be seen, the outputs of 3D-EPN often contain missing or extra parts. The outputs of FC are accurate but the points are overly concentrated in certain regions. The outputs of Folding contain many floating points and the outputs of PN2 are blurry. The outputs of our model best match the ground truth in terms of global geometry and local point density.

E. Additional Results on KITTI

Figure 11 shows the distribution of fidelity error and minimal matching distance on the completion results on KITTI. It can be seen that there are a few failure cases with very high error that can bias the mean value reported in Section 5.3. Figure 12 shows some qualitative examples. We observe that in most cases, our model produces a valid car shape that matches the input while being different from the matched model in ShapeNet, as the one shown in the top figure. However, we also observe some failure cases, e.g. the one shown in the bottom figure, caused by extra points from the ground or nearby objects that are within the car's bounding box. This problem can potentially be resolved by adding a segmentation step before passing the partial point cloud to PCN.

F. More Architecture Analysis

Effect of stacked PN layers Here we test variants of PCN-CD with different number of stacked PN layers in the encoder. The mean CD and EMD on the ShapeNet test set are shown in Figure 13. It can be seen that the advantage of using 2 stacked PN layers over 1 is quite apparent, whereas the benefit of using more stacked PN layers is almost negligible. This shows that 2 stacked PN layers are sufficient for mixing the local and global geometry information in the

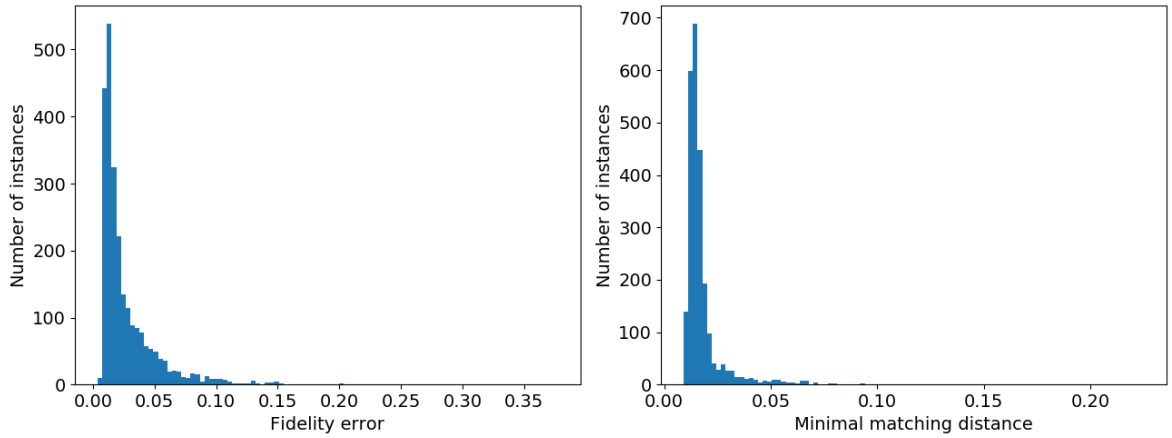


Figure 11: Distribution of fidelity error and minimal matching distance on KITTI car completions

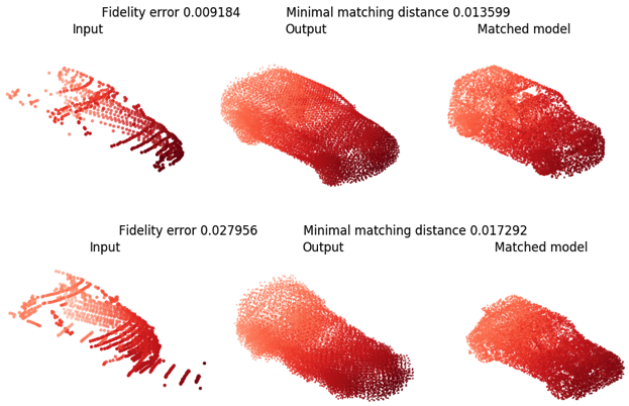


Figure 12: **Completion (middle) and matched model (right) for cars in KITTI (left)**. The matched model on the right is the car point cloud from ShapeNet that is closest to the completion output in Chamfer Distance. Top figure shows a successful completion and bottom figure shows a failure case caused by incorrect segmentation.

input. Thus, we keep the number of stacked PN layers as 2 in our experiments, even though PCN's performance can be further improved by using more stacked PN layers.

Effect of bottleneck size Here we test variants of PCN-EMD with different bottleneck size, i.e. the length of the global feature vector $\mathbf{v}$. The mean CD and EMD on the ShapeNet test set are shown in Figure 14. It can be seen that PCN's performance improves as the bottleneck size increases. In our experiments, we choose the bottleneck size to be 1024 because a larger bottleneck of size 2048 cannot fit into the memory of a single GPU. This implies that if multiple GPUs are used for training, PCN's performance can further improve with a larger bottleneck.

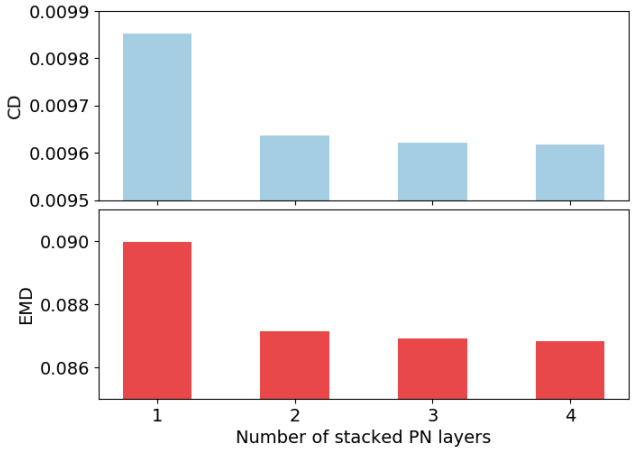


Figure 13: Test error with different number of PN layers

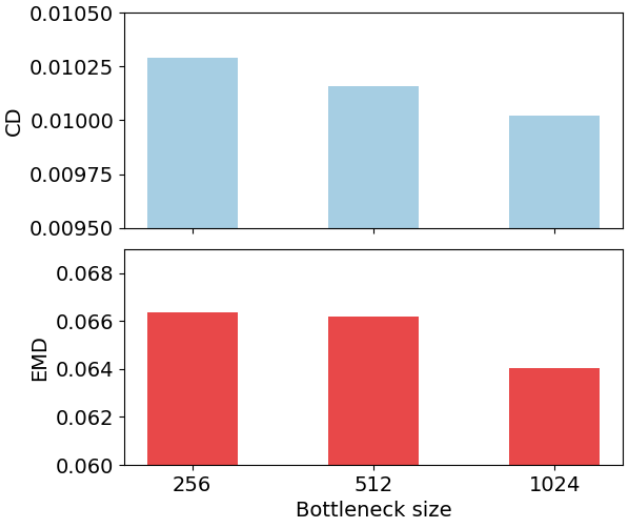


Figure 14: Test error with different bottleneck sizes

G. More Visualizations

Keypoint Visualization As noted in [37], the PN layer can be interpreted as selecting a set of keypoints which describes the shape. More specifically, each output unit of the shared MLP can be considered as a function on the points. The keypoints are points that achieve the maximum value for at least one of these point functions. In other words, they are points whose feature values are "selected" by the maxpooling operation to be in the final feature vector. Note that a point can be selected more than once by achieving the maximum for multiple feature functions. In fact, as shown in Table 3, the number of keypoints is usually far less than the bottleneck size (1024) or the number of input points. This implies that as long as these keypoints are preserved, the learned features won't change. This property contributes to our model's robustness against noise.

In Figure 15, we visualize the keypoints selected by the two PN layers in our stacked PN encoder. It can be observed that the two PN layers summarizes the shape in a coarse-to-fine fashion – the first layer selects just a few points that compose the outline of the shape, while the second layer select more points that further delineate the visible surfaces. Note that this coarse-to-fine description emerges without any explicit supervision.

	Average number of points
Input	1105
Keypoints (1st PN layer)	101
Keypoints (2nd PN layer)	363

Table 3: Number of keypoints versus number of input points

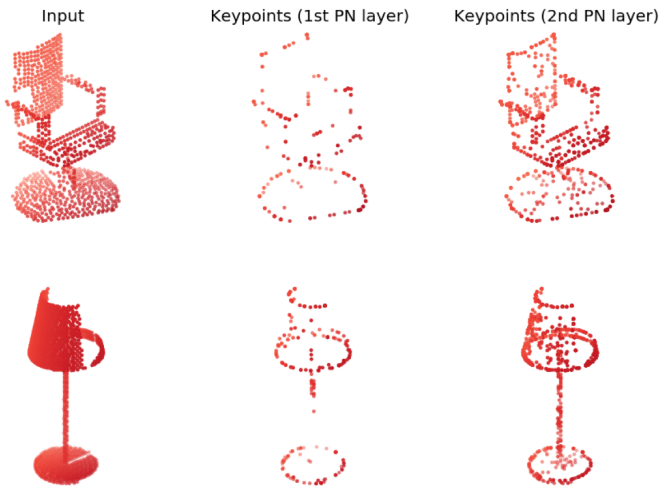


Figure 15: Keypoints visualization

Feature Space Visualization In Figure 17, we use t-SNE [27] to embed the 1024-dimensional global features of the ShapeNet test instances into a 2D space. It can be seen that shapes are clustered together by their semantic categories. Note that this is also an emerging behavior without any supervision, since we do not use the category labels at all during training.

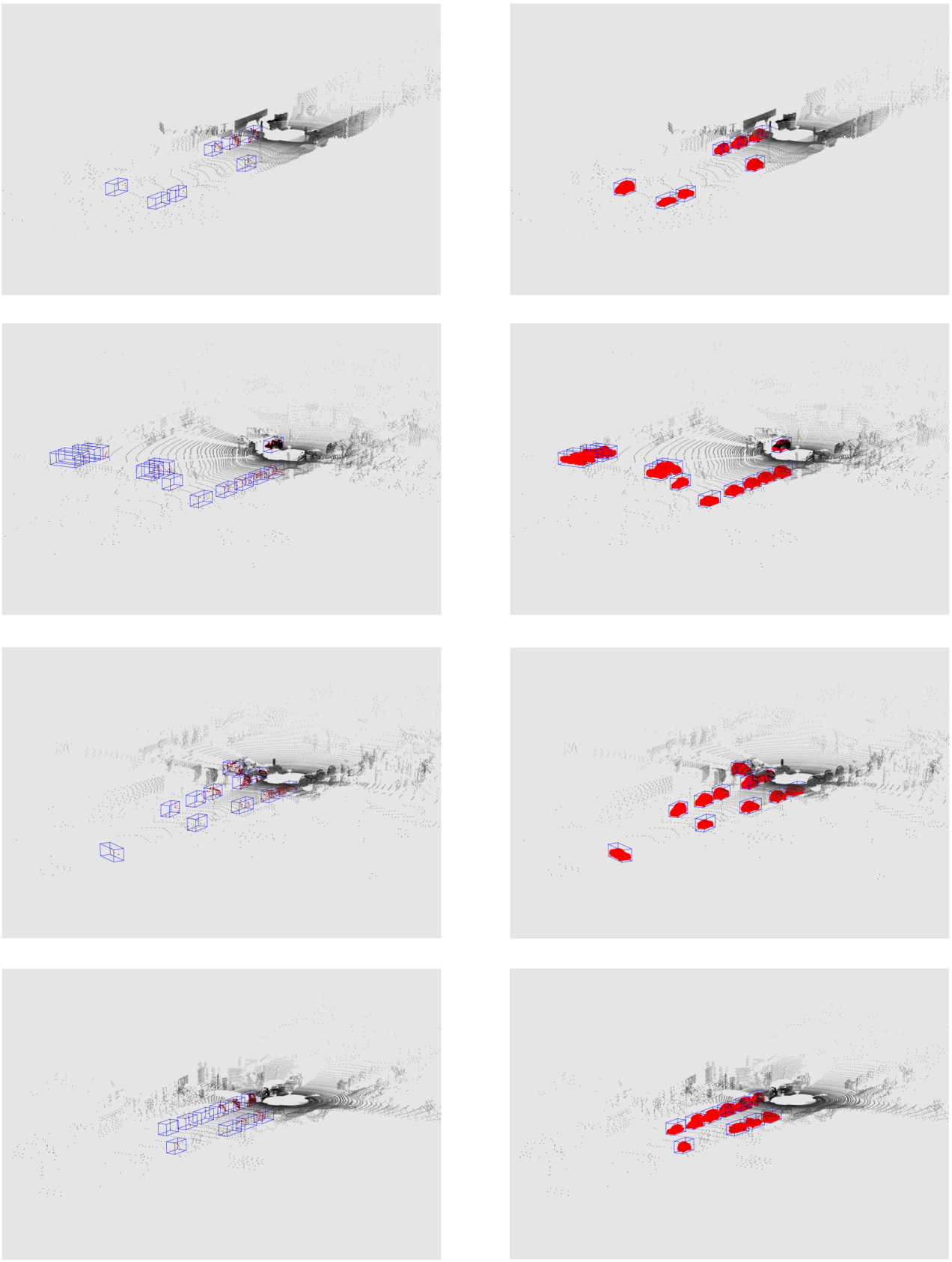


Figure 16: Qualitative completion results on KITTI

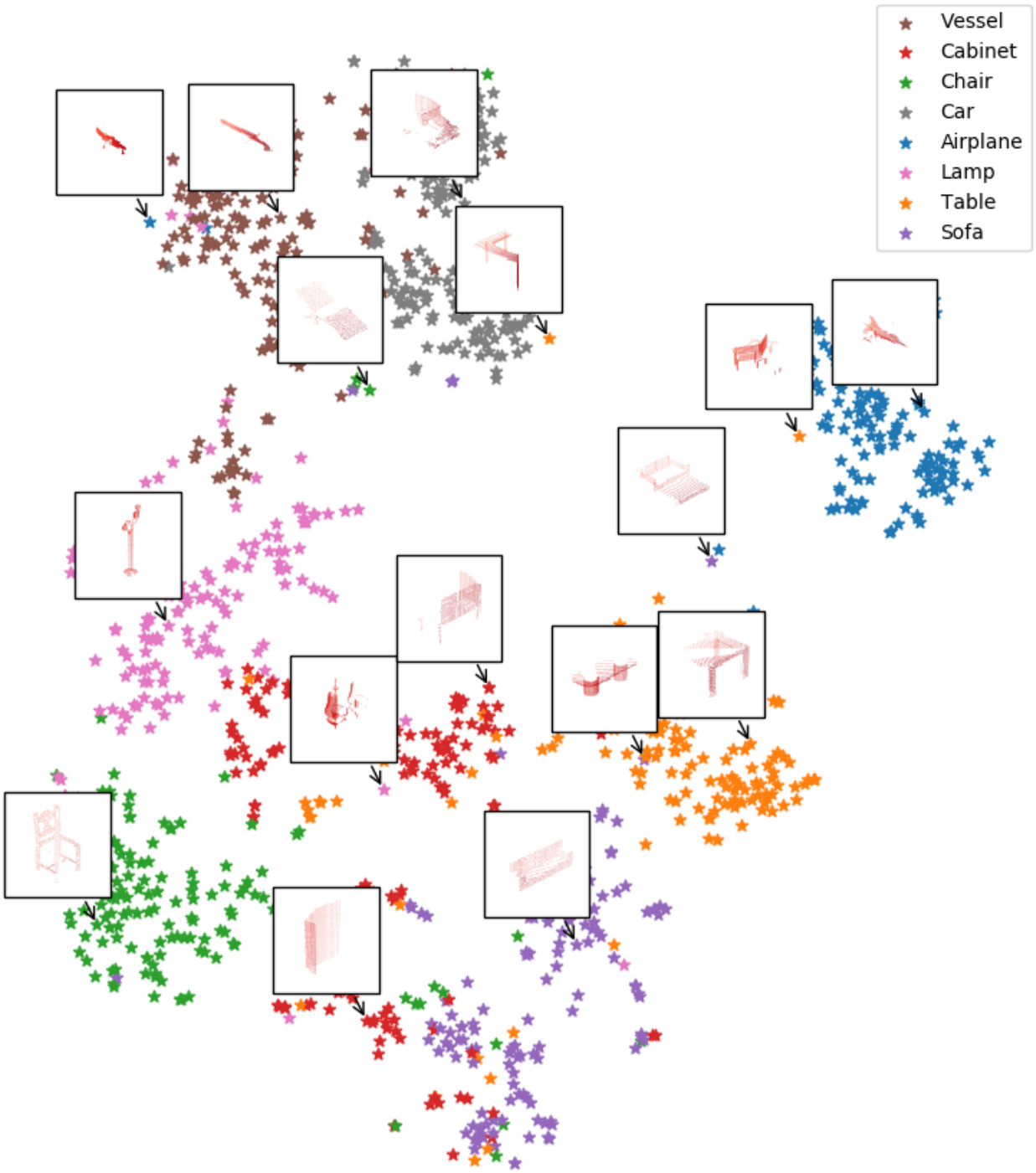


Figure 17: T-SNE embedding of learned features on partial point clouds

Table 4: Seen categories of ShapeNet dataset - Chamfer Distance

Method	Mean Chamfer Distance per point								
	Avg	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel
3D-EPN	0.020147	0.013161	0.021803	0.020306	0.018813	0.025746	0.021089	0.021716	0.018543
FC	0.009799	0.005698	0.011023	0.008775	**0.010969**	**0.011131**	0.011756	0.009320	0.009720
Folding	0.010074	0.005965	0.010831	0.009272	0.011245	0.012172	0.011630	0.009453	0.010027
PN2	0.013999	0.010300	0.014735	0.012187	0.015775	0.017615	0.016183	0.011676	0.013521
PCN-CD	**0.009636**	**0.005502**	**0.010625**	**0.008696**	0.010998	0.011339	**0.011676**	**0.008590**	**0.009665**
PCN-EMD	0.010021	0.005849	0.010685	0.009080	0.011580	0.011961	0.012206	0.009014	0.009789

Table 5: Seen categories of ShapeNet dataset - Earth Mover's Distance

Method	Mean Earth Mover's Distance per point								
	Avg	Airplane	Cabinet	Car	Chair	Lamp	Sofa	Table	Vessel
3D-EPN	0.081785	0.061960	0.077630	0.087044	0.076802	0.107317	0.080802	0.080996	0.081732
FC	0.171280	0.073556	0.214723	0.157297	0.189727	0.240547	0.191488	0.161117	0.141782
Folding	0.228015	0.156438	0.221349	0.174567	0.297427	0.319983	0.245664	0.189904	0.218788
PN2	0.101445	0.059574	0.116179	0.066942	0.110595	0.185817	0.102642	0.086053	0.083755
PCN-CD	0.087142	0.046637	0.097691	0.057178	0.086787	0.169540	0.083425	0.080783	0.075094
PCN-EMD	**0.064044**	**0.038752**	**0.070729**	**0.054967**	**0.068074**	**0.084613**	**0.072437**	**0.060069**	**0.062713**

Table 6: Unseen categories of ShapeNet dataset - Chamfer Distance

Method	Mean Chamfer Distance per point									
	Avg	Bus	Bed	Bookshelf	Bench	Avg	Guitar	Motorbike	Skateboard	Pistol
3D-EPN	0.0415	0.03594	0.04785	0.03912	0.04307	0.0443	0.04735	0.04067	0.04784	0.04136
FC	0.0142	0.00982	0.02123	0.01512	0.01081	0.0129	0.00992	**0.01456**	0.01200	0.01497
Folding	**0.0138**	0.01058	**0.01908**	0.01488	**0.01055**	**0.0124**	**0.00906**	0.01556	**0.01191**	**0.01313**
PN2	0.0169	0.01260	0.02378	0.01687	0.01445	0.0168	0.01429	0.01635	0.01290	0.02353
PCN-CD	0.0142	**0.00946**	0.02163	**0.01479**	0.01102	0.0129	0.01040	0.01475	0.01204	0.01423
PCN-EMD	0.0146	0.00972	0.02236	0.01496	0.01139	0.0131	0.01147	0.01525	0.01211	0.01359

Table 7: Unseen categories of ShapeNet dataset - Earth Mover's Distance

Method	Mean Earth Mover's Distance per point									
	Avg	Bus	Bed	Bookshelf	Bench	Avg	Guitar	Motorbike	Skateboard	Pistol
3D-EPN	0.1189	0.10681	0.13318	0.11856	0.11711	0.1309	0.16255	0.11893	0.11328	0.12873
FC	0.1998	0.16686	0.25567	0.20619	0.17050	0.1790	0.17635	0.18132	0.16706	0.19134
Folding	0.2494	0.22150	0.32603	0.22555	0.22457	0.2733	0.32040	0.25137	0.25435	0.26689
PN2	0.1062	0.08081	0.14900	0.12155	0.07349	0.1270	0.17836	0.10372	0.08825	0.13749
PCN-CD	0.0908	0.06270	0.13556	0.10332	0.06161	0.1130	0.16834	**0.09206**	0.08464	0.10702
PCN-EMD	**0.0705**	**0.05991**	**0.10350**	**0.07607**	**0.06044**	**0.0816**	**0.07478**	0.09471	**0.06249**	**0.09426**