

Self-Supervised Point Cloud Completion via Inpainting

Himangi Mittal
hmittal@andrew.cmu.edu

Brian Okorn
bokorn@andrew.cmu.edu

Arpit Jangid
ajangid@andrew.cmu.edu

David Held
dheld@andrew.cmu.edu

Carnegie Mellon University
Pittsburgh, PA, USA

Abstract

When navigating in urban environments, many of the objects that need to be tracked and avoided are heavily occluded. Planning and tracking using these partial scans can be challenging. The aim of this work is to learn to complete these partial point clouds, giving us a full understanding of the object’s geometry using only partial observations. Previous methods achieve this with the help of complete, ground-truth annotations of the target objects, which are available only for simulated datasets. However, such ground truth is unavailable for real-world LiDAR data. In this work, we present a self-supervised point cloud completion algorithm, PointPnCNet, which is trained only on partial scans without assuming access to complete, ground-truth annotations. Our method achieves this via inpainting. We remove a portion of the input data and train the network to complete the missing region. As it is difficult to determine which regions were occluded in the initial cloud and which were synthetically removed, our network learns to complete the full cloud, including the missing regions in the initial partial cloud. We show that our method outperforms previous unsupervised and weakly-supervised methods on both the synthetic dataset, ShapeNet, and real-world LiDAR dataset, Semantic KITTI.

1 Introduction

Autonomous vehicles often understand the world around them using depth sensors such as LiDAR. However, the LiDAR point clouds are often incomplete even when recorded from multiple viewpoints over time. To accurately track objects and plan routes to avoid collisions, it is important for autonomous vehicles to understand the complete shape of surrounding objects.

Previous methods [14, 18, 20, 21, 22] have learned to complete partial point clouds but they strongly rely on the availability of ground truth complete shapes as supervision. Since complete point clouds are costly to obtain for real-world scenarios, these methods typically train only from simulated data where ground-truth completions are available. This limitation motivates our approach to learn only from partial point clouds to complete shapes without ever observing the ground-truth completed point clouds during training.

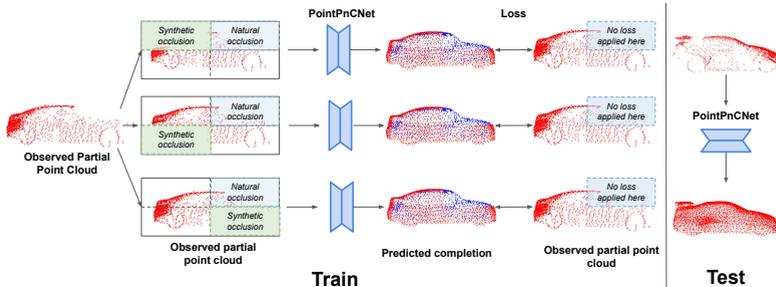


Figure 1: We adopt an *inpainting*-based approach for self-supervised point cloud completion to train our network using only partial point clouds. Given a partial point cloud as input, we randomly remove regions from it and train the network to complete these regions using the input as the pseudo-ground truth. The loss is only applied to the regions which have points in the observed input partial point cloud (red). Since, the network cannot differentiate between synthetic and natural occlusions, the network predicts a complete point cloud.

To this end, our method leverages self-supervision via an inpainting-based approach where we randomly remove regions from the partial point clouds and train the network to complete the entire point cloud. Across multiple training examples, different regions will be occluded and varying regions will be synthetically removed. Because the network does not know which regions were artificially removed and which were naturally occluded in each original partial point cloud, the network learns to attempt to complete the entire point cloud.

In contrast to images, where a mask can specify the region to inpaint [1, 11, 13, 23, 24], the unstructured nature of point clouds makes it challenging to define which regions the network needs to inpaint. We solve this using a region-aware loss which penalizes only the regions where the original point cloud was present. Additionally, we partition the point cloud into local regions using intersecting half-spaces and encode/decode them separately. This allows the network to learn data-driven embeddings separately for each local region that specialize in individual region-level object parts. We also encode/decode at the global point cloud level to further allow the network focus on each region jointly with each other. Previous works [6, 10] depend on aligning multiple viewpoints of an object during training which can be sensitive to pose alignment errors. While we also incorporate a multi-viewpoint loss, we show that our use of inpainting allows our method to be robust to alignment errors.

The key contributions of this paper are as follows: 1). We present a novel inpainting-based self-supervised algorithm that learns to complete missing local regions in an incomplete point cloud without the need for ground truth point cloud completions, 2). Our multi-level encoder-decoder based architecture, PointPnCNet, partitions the point clouds to learn local and global embeddings to obtain improved completion performance, 3). Our approach outperforms existing methods for unsupervised point cloud completion [6, 10] when evaluated on the standard completion benchmarks of ShapeNet [2] and SemanticKITTI [10].

2 Related Work

Supervised Point Cloud Completion Most of the existing 3D shape completion methods [9, 14, 18, 19, 21, 22, 27] make use of complete ground-truth shape labels. A common approach for point cloud completion is to use an encoder-decoder style architecture [12, 17, 21, 27]. On the other hand, Tchapmi *et al.* [24] proposed to generate a point cloud using a hierarchical rooted tree structure. Our architecture builds on the typical encoder-decoder style of previous work [27]. In contrast to the above supervised methods, our proposed approach does not

require ground truth annotations. This allows our method to be trained using LiDAR data in the wild, as opposed to the previous methods which are trained only with simulated data.

Weakly-Supervised Methods Recently, Gu *et al.* [9] proposed a weakly-supervised approach for point cloud completion where the pose of the input partial point cloud and 3D canonical shape are jointly optimized. Their method is weakly-supervised via multi-view consistency among the multiple partial observations of the same instance. Our method also uses partial point clouds, however, using our inpainting-based approach, our method is able to learn a more accurate completion and is robust to view alignment errors. Other methods also learn 3D shape reconstruction using weak supervision [16, 22, 30]. Among these, Differentiable Point Clouds (DPC) [10] jointly predicts camera poses and a 3D shape representation given two image views of the same instance. The geometric consistency between the estimated 3D shape and the input images is enforced using an end-to-end differentiable point cloud projection. We show in the results that we significantly outperform this method.

Image Inpainting In the area of image inpainting [0, 11, 13, 23, 24, 28], Zhan *et al.* [28] proposed self-supervised partial completion networks (PCNets) to complete an occluded object’s mask and content in the input image. Our method takes inspiration from Zhan *et al.* [28] for shape completion in 3D point clouds. However, due to the structured nature of images, often a mask can be used to specify the region to inpaint. In 3D point clouds, where the data is unstructured and sparse in nature, it is difficult to specify a “mask” for the regions to inpaint. There is ambiguity between regions that have been “masked out” and regions that are naturally occluded, making the task of inpainting challenging for point cloud data.

Point Cloud Inpainting Some of the previous methods [3, 4, 5, 8, 25, 29] have also explored inpainting in the point cloud domain. However, these methods either use ground truth during training [25, 29], rely on template-matching within a data sample [4, 5, 8], or project a point cloud into 2D structured representation [3]. Our method is novel in the sense that it uses inpainting directly on the point clouds without any ground truth information while leveraging large datasets to learn domain-specific priors.

3 Method

The point cloud completion problem can be defined as follows: given an incomplete set of sparse 3D points X , sampled from a partial view of an underlying dense object geometry G , the goal is to predict a new set of points Y , which mimics a uniform sampling of G .

3.1 Self-supervised Inpainting

In our self-supervised inpainting-based approach to learn to complete full point clouds using only partial point clouds, we randomly remove regions of points from a given partial point cloud and train the network to inpaint these synthetically removed regions. The original partial point cloud is then used as a pseudo-ground truth to supervise the completion. Since we do not have the complete ground-truth point cloud, supervision is only applied to the regions of the original point cloud that contain points (i.e. unoccluded regions).

The network leverages the information of available regions across samples and embeds each region separately that can generalize across partially occluded samples with different missing regions. Further, due to the stochastic nature of region removal, the network cannot easily differentiate between the synthetic and original occlusions of the input partial point cloud, making the network learn to complete the point cloud. Thus, the combination of inpainting, random-region removal, and region-specific embeddings enables the model to generate all the regions and create a complete point cloud.

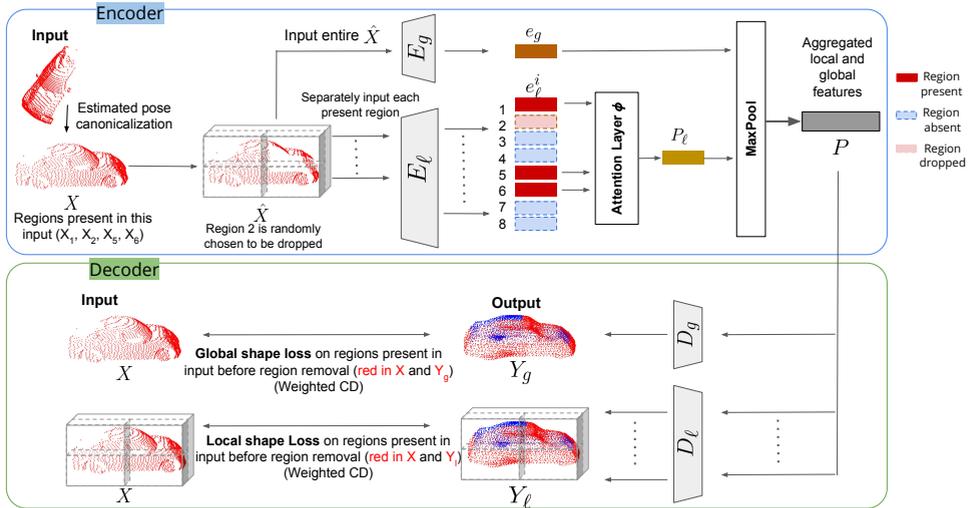


Figure 2: PointPnCNet Architecture: Our method first estimates a canonicalized orientation of a partial point cloud, which has some regions missing due to natural occlusions. We then randomly drop one or more of the regions to create additional synthetic occlusions. We compute global features e_g and local features P_ℓ which we combine into an encoding P . Our multi-level decoder uses the encoding P to generate a completed point cloud. The global shape loss and local shape loss are only applied to the regions of the output where points are present in the original cloud (before synthetic occlusions) which are shown in red in X , Y_g , and Y_ℓ . The blue points in Y_g and Y_ℓ are not present in the original cloud, so we have no ground truth about their positions; thus they are not penalized in the loss. The final output of the network is the concatenation of the outputs from Y_g and Y_ℓ .

3.2 Network Architecture

Figure 2 depicts the architecture and training flow of our network, Point Cloud Partition-and-Completion Network (PointPnCNet). We use a multi-level encoder-decoder architecture to allow the network to focus on different parts of an object. We present the evaluations of various alternate designs of our method in the appendix.

3.2.1 Multi-Level Encoder

Our encoder consists of multiple, parallel encoder streams that encode the input partial point cloud at global and local levels. The global-level encoder operates on the full-scope of the object while a local-level encoder focuses on a particular region of the object. Since a local encoder only sees points in a given local region and is invariant to other parts of the shape which may be missing, local encoders make the network robust to occlusions by focusing on individual object parts separately. Global encoder further enhances shape consistency by focusing on regions jointly with each other. Given a partial point cloud, we estimate its canonical frame using a learned method (Sec. 4) and transform it to obtain a canonicalized partial point cloud X . We show that our method is robust to errors in this canonicalization (Sec. 4.6). We then partition the canonicalized partial point cloud using intersecting half-spaces that are produced by the coordinate planes after canonicalization. This effectively separates the space into eight 3D octants as shown in Figure 2. While other types of partitioning could be used, we found this subdivision to be simple and reasonably effective. Rather than a strict partitioning, we allow a small overlap between neighboring

regions such that points in the overlap are present in both regions. This helps to avoid seams at boundaries. Let X_i consist of the points in the region i . After partitioning, we remove points of any particular region with a probability p to simulate a synthetic occlusion. We use this synthetically occluded point cloud \hat{X} as input to our inpainting network.

The points in the remaining regions are aggregated together and passed as input to the global encoder, E_g , to give a global embedding e_g (Figure 2). In parallel, each remaining region X_i is separately encoded by a local encoder, E_ℓ to obtain a local embedding for that region, e_ℓ^i . To aggregate the local feature embeddings, each embedding e_ℓ^i is fed as input into an attention module, consisting of an MLP layer, that generates a set of weights $w_i = \phi(e_\ell^i)$. These weights are used to weigh each of the embeddings e_ℓ^i in a linear combination to form the aggregate embedding $P_\ell = \sum_i \mathbb{1}_i w_i e_\ell^i$, where $\mathbb{1}_i$ is an indicator function which equals 1 if region i is present (i.e. present in the original partial point cloud X and not randomly removed) and 0 otherwise. We then perform a channel-wise max-pooling across the global encoding e_g and the attention-weighted local encoding P_ℓ , as $P = \max(e_g, P_\ell)$.

3.2.2 Multi-Level Decoder

Our decoder consists of multiple decoder streams that work in parallel to decode the fused embedding P (Figure 2). The global decoder D_g takes the embedding P as input and attempts to generate an entire completed point cloud Y_g . In parallel, we use a local decoder D_ℓ to decode the points in each region of the input space. The embedding P is concatenated with a one-hot vector indicating each region’s location and create a region-specific embedding. Through one-hot encoding, each decoder specializes in completing a certain region and learns a region-specific embedding. The decoder takes as input these region-specific embeddings and generates a subset of the output point cloud localized to the respective region Y_ℓ^i . The generated local regions are combined together to obtain the full point cloud Y_ℓ . The multi-level output generated by the network captures the details of the object at global and local levels. The outputs of the multi-level decoder streams, D_ℓ and D_g , are concatenated to form the final prediction of our network as Y .

3.3 Point Cloud Completion Losses

The standard loss used for comparing two point clouds is the Chamfer Distance (CD). It is a bi-directional permutation invariant loss over two point clouds representing the nearest neighbor distance between each point and its closest point in the other cloud. In our method, we use an asymmetric Weighted Chamfer Distance loss, \mathcal{L}_{wcd} , defined as,

$$\mathcal{L}_{wcd}(X, Y) = \frac{(1-\beta)}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2 + \frac{\beta}{|Y|} \sum_{y \in Y} \min_{x \in X} \|y - x\|_2 \quad (1)$$

where X is the original partial point cloud used here as pseudo-ground truth and Y is the output. Importantly, we only compute the loss for the regions that are present in X . A weight of $(1 - \beta)$ is applied to the first term in Eqn. 1 which penalizes the distance from each point in X to its nearest neighbor in Y . This term enforces that the output point cloud Y should contain points that are close to those in X . Note that the input to the network is \hat{X} , which has synthetic occlusions, not X , which is the original partial point cloud. A weight of β is applied to the second term in Eqn. 1 to penalize the distance from each point in Y to its nearest neighbor in X . We do not expect this term to reach 0 for a well-trained network since X only contains a partial point cloud, while output Y contains the entire point cloud; we still find it a helpful regularization. We impose the following variants of \mathcal{L}_{wcd} on the model,

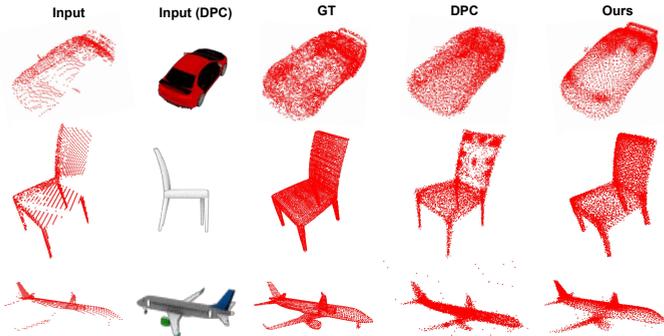


Figure 3: Qualitative results on the ShapeNet dataset compared with our baseline, DPC [10]. Our method is better able to reconstruct fine-grained object details (back portion of the car and engines on the airplane), produces fewer noisy points for the airplane and produces more uniformly distributed points in the chairs than the baseline.

Inpainting-Global Loss: This loss acts as a *global shape loss*, focusing on the overall shape of an object. We impose it as the Weighted Chamfer Distance (Eqn. 1) between original partial point cloud X and output of the global decoder Y_g and define it as $\mathcal{L}_{wcd}(X, Y_g)$.

Inpainting-Local Loss: We impose Inpainting-Local loss as the Weighted Chamfer distance between each region output Y_ℓ^i from local decoder D_ℓ and corresponding partitioned region X_i in the original partial point cloud X where i indexes over regions. While Inpainting-Global loss considers the entire X to find the nearest neighbor, Inpainting-Local loss differs in that it only considers the partitioned region X_i to find the nearest neighbor. Thus, it acts as a *local shape loss* that enables the network to learn region-specific shapes and embeddings and focus on the finer details of an object. We do not penalize regions that are missing in X where a region is considered missing if the number of points in that region is below a certain threshold. The Inpainting-Local Loss is therefore defined as, $\sum_i \mathbb{1}_i \cdot \mathcal{L}_{wcd}(X_i, Y_\ell^i)$ where the indicator function $\mathbb{1}_i$ equals one if region i is present in X and zero otherwise.

Multi-View Consistency: Similar to Gu *et al.* [9], our method uses multi-view consistency as an auxiliary loss. Their method explicitly performs pose estimation. Similarly, we perform an estimated pose canonicalization (weakly supervised, Sec. 4.2). We also show later (Sec. 4.6) that our method is robust to canonicalization errors. During training, we sample a view k from V partial views of an object X given as X^1, \dots, X^V . Since all the views of an object correspond to the same object, for input partial point cloud X^k , the loss is computed with all views X^1, \dots, X^V . We define a global inpainting multi-view consistency loss as $\sum_{j=1}^V \mathcal{L}_{wcd}(X^j, Y_g^k)$ where X^j is the j^{th} view of X and Y_g^k is the global output from decoder D_g . We also define local inpainting multi-view consistency loss as $\sum_i \sum_{j=1}^V \mathbb{1}_i^j \cdot \mathcal{L}_{wcd}(X_i^j, Y_\ell^{i,k})$ where i indexes over regions, X_i^j is the i^{th} region of view X^j , $Y_\ell^{i,k}$ is the region output from local decoder D_ℓ for input X_i^k , and $\mathbb{1}_i^j$ is 1 if region i is present in X^j and 0 otherwise. During training, we sum the losses as, $\sum_{j=1}^V \mathcal{L}_{wcd}(X^j, Y_g^k) + \sum_i \sum_{j=1}^V \mathbb{1}_i \cdot \mathcal{L}_{wcd}(X_i^j, Y_\ell^{i,k})$. This multi-view information is only available at training time.

4 Experiments

4.1 Implementation Details

During test-time, we use a single view of an object. The multiple views are only available during training. To get the final completed point cloud, we concatenate the output from multi-level decoders D_g and D_ℓ . We do not remove regions at inference time. Otherwise,

the network during inference is the same as described above. For consistency with prior work [6], we resample each partial point cloud X to have a total of 3096 points. PointPnCNNet uses architecture from PCN [27] for encoder and decoder blocks. The model is trained from scratch for 400K iterations with batch size of 32, learning rate of $5e-4$ decayed by 0.5 after every 100K iterations and $\beta = 0.25$ in \mathcal{L}_{wcd} . Please refer to appendix for more details.

4.2 Experimental Setup

Following the evaluation protocol of Gu *et al.* [6], we test our approach on ShapeNet [2] and Semantic KITTI [10]. ShapeNet has ground truth annotations for each object class which allows us to evaluate how well our method generates completed shape. On the other hand, Semantic KITTI allows us to evaluate the robustness of our method on real LiDAR data.

The observations are transformed to a canonical frame (a shared reference frame which aligns all instances of a class) using canonical frame predictions generated via IT-Net [26] for ShapeNet and predicted bounding boxes obtained from OpenPCDet [15] for Semantic KITTI. We use IT-Net for ShapeNet as it is trained in a weakly-supervised manner from only classification labels and learns to align the instances of each class, without any pose supervision. In general, any pose estimator can be used here. We evaluate the robustness of our method to this canonical frame estimation in Sec. 4.6.

ShapeNet: ShapeNet [2] is a synthetic dataset with 3D CAD models. We report our results on three categories, airplanes, cars, and chairs, that are commonly used in the related works [6, 10]. We use the same data split provided by DPC [10], where RGB-D data is generated for random camera views with fixed translation, similar to Gu *et al.* [6]. For evaluation, we use ground truth point clouds provided by DPC [10] which are densely sampled from ShapeNet meshes and downsampled to 8192 points.

Semantic KITTI: We evaluate our method for a real-world scenario using KITTI [10]. Previous methods [6, 20, 21, 22] have a standard protocol of evaluation on real-world data by testing on the cars of KITTI only. We adopt the same protocol in our work. Following Gu *et al.* [6], we train over the parked car instances (which have multiple views captured when a LiDAR sensor moves through the scene and scans a parked car from different locations) with sequences 00 to 10 (excluding 08) as train set and sequence 08 as test set. The train set consists of 507 parked car instances and 46152 observations, while the test set has 229 parked car instances and 16296 observations.

Although having no complete ground truth information in KITTI creates some limitations in its evaluation, testing on this dataset shows the ability of our method to handle real-world LiDAR data. By combining the evaluations on a real-world dataset (KITTI) and a synthetic dataset (ShapeNet), which has ground truth annotations, we are able to present a more thorough evaluation. This is the standard evaluation procedure following Gu *et al.* [6].

Metrics: Our primary metric for quantitatively evaluating shape completion is the *Chamfer Distance (CD)*, as is used in previous works [6, 18, 22]. We define this metric in its weighted form in Equation 1. For evaluation, to compare with the ground-truth completed point cloud, we equally weight each component with a β of 0.5. Additionally, we follow Gu *et al.* [6] and report each component of the Chamfer distance independently: the mean distance from each predicted point to its nearest true point described as *Precision*, and the mean distance from each true point to its nearest predicted point described as *Coverage*. *Precision* describes how well the predicted points match the local shape, while *Coverage* is related to how much of the shape is completed. We also evaluate the Earth Mover’s Distance (EMD) [22], which finds a bijection between the predicted point cloud and the ground truth point cloud that minimizes the average distance between corresponding points. Like

| Method | Airplane | | | Car | | | Chair | | |
|----------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| | CD | Precision | Coverage | CD | Precision | Coverage | CD | Precision | Coverage |
| DPC [14] | 3.91 | - | - | 3.47 | - | - | 4.30 | - | - |
| Gu <i>et al.</i> [6] | 1.95 | 0.91 | 1.05 | 2.68 | 1.27 | 1.41 | 3.33 | 1.69 | 1.64 |
| PointPnCNet (Ours) | 1.66 | 0.79 | 0.87 | 2.48 | 0.99 | 1.49 | 2.70 | 1.36 | 1.34 |

Table 1: Quantitative results on the Airplane, Cars, and Chairs categories of the ShapeNet dataset. All results are multiplied by a factor of 100, following Gu *et al.* [6].

| Method | CD | Precision | Coverage | Dataset | Model without | | | |
|----------------------|--------------|--------------|--------------|----------|---------------|-----------------|-------------|------------|
| | | | | | Inpainting | Multi-View Loss | Global Loss | Local Loss |
| Gu <i>et al.</i> [6] | 0.194 | 0.087 | 0.107 | ShapeNet | +0.98 | +0.27 | +0.50 | +0.32 |
| Densified Input | 0.130 | 0.025 | 0.105 | KITTI | +0.05 | +0.03 | +0.30 | +0.24 |
| PointPnCNet (Ours) | 0.095 | 0.045 | 0.050 | | | | | |

(a)

(b)

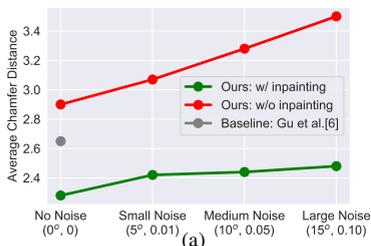
Table 2: (a). Quantitative results on the Semantic KITTI dataset, (b). Increase in mean Chamfer distance on ShapeNet and KITTI datasets for various ablations of our method. ShapeNet results are averaged across each object category.

previous work, we also evaluate the F-score@1% [20].

4.3 Point Cloud Completion Results

We compare with the current state-of-the-art unsupervised methods, DPC [14] and Gu *et al.* [6]. Table 1 shows our method outperforming the baseline methods on the synthetic ShapeNet dataset, producing lower Chamfer distances across all shape categories. *Precision* and *Coverage* metrics also improve, showing that our method produces more accurate points and better covers the full object shape. Our method is also able to outperform DPC [14] as per the Earth Mover Distance (EMD) metric (Table 3b). Since the code for [6] is not publicly available, the EMD metric on that method cannot be evaluated.

We further show in Table 2a that our method outperforms the previous state-of-the-art [6] on the Semantic KITTI dataset, generating outputs that are significantly more accurate than [6]. The KITTI dataset is more realistic than ShapeNet. With samples having a range of sparsity (since real-world LIDAR gets sparse with distance), it represents the data available in self-driving scenarios. We also show improvement compared to a simple densification baseline (Densified Input in Table 2a) which suggests that our method is indeed completing the partial point clouds rather than simply densifying them. This densification baseline uniformly samples points within the volume of a local surface that is approximated as an ellipsoid, formed using eigenvalues for 10 nearest neighbors of each point in input partial point cloud. We also conduct a uniformity analysis whose results we report in the appendix.



(a)

| Training Method | Airplane | Car | Chair |
|--------------------|--------------|--------------|--------------|
| DPC [14] | 0.146 | 0.160 | 0.194 |
| PointPnCNet (Ours) | 0.108 | 0.120 | 0.140 |

(b)

| Training Method | F-score@1% ↑ | EMD ↓ |
|---------------------|--------------|-------------|
| PointPnCNet (Ours) | 0.67 | 0.45 |
| Ours w/o inpainting | 0.44 | 0.62 |

(c)

Table 3: (a). Mean of the chamfer distance across ShapeNet categories (Airplane, Car, Chair). Our method is trained with noisy poses, with & without inpainting, shown in green and red, respectively. Baseline Gu *et al.* [6] has no added noise, (b). Earth Mover Distance (EMD) metric on Shapenet. Lower EMD is better, (c). F-Score@1% and EMD metrics on Semantic KITTI. We evaluate them on our method vs our ablation of without inpainting.

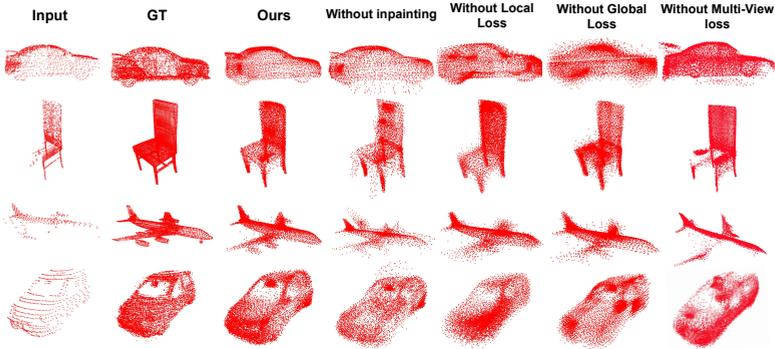


Figure 4: Qualitative results for ablation study on ShapeNet and KITTI. Without inpainting, local loss, global loss, and multi-view loss, the network yields noisy output.

4.4 Qualitative Results

We present the qualitative results of our method for each category of ShapeNet in Figure 3 and KITTI in Figure 4. In comparison to the baseline DPC [14], we observe that our method is able to better cover the target object with a more uniform distribution over the target surface and accurately reconstructs the fine-grained object details. For example, our method is able to complete the back of the car and the side mirror whereas the baseline outputs noisy points. For chairs, our method generates more uniformly distributed points whereas the baseline outputs patches/clusters of points in that location. This highlights the fact that our method is better able to generalize and complete the unseen regions of incomplete shapes.

4.5 Ablation Study

Inpainting Loss: Region removal to create synthetic occlusions and the task of inpainting are removed, **Multi-View Loss:** The multi-view loss is removed from our training method. Each partial point cloud is used to supervise its own, synthetically occluded completion, **Global Level:** We remove the Inpainting-Global Loss, global encoder E_g and global decoder D_g from our completion pipeline. **Local Level:** The Inpainting-Local loss, local encoder E_ℓ and local decoder D_ℓ are removed from our method. The number of output points for the *global shape* and *local shape* ablations are kept consistent with our full method.

We report the ablation results in Table 2b on ShapeNet, as an average over all categories, and on Semantic KITTI. We find that all components of our system are crucial for optimal performance across both datasets. We further report the F-score@1% [24] and the EMD metric on the Semantic KITTI dataset with the ablation of removing inpainting in Table 3c. We find that inpainting greatly improves our results across both of these metrics.

The qualitative effects of our ablation study can be seen in Figure 4. We observe that inpainting generates an object-specific, less noisy output, when comparing "Ours" and "Without Inpainting". Our method without local loss fails to complete local details of an object such as back of a car or wings of a plane and without the global loss predicts a generic, noisy shape of an object. Since each local encoder and decoder only observe the points within that region and not the points in the other potentially occluded regions, they allow the network to focus on individual parts of an object and be robust to different occlusion patterns. The local loss helps in creating a more uniform completion, since it completes its associated region and the global loss reasons about the entire shape of the object. Finally, without multi-view loss, the output point cloud is noisy and incomplete as can be seen in all shapes.

4.6 Robustness to Canonical Frame Estimation

Previous work [6] depends on multiple views which can be sensitive to the pose alignment errors. While we also use a multi-view loss, our inpainting losses make the model robust to noisy alignment allowing it to learn from poorly aligned data. The mean rotation/translation difference, after using IT-Net for pose canonicalization, between the multiple partially observed shapes during both training and inference is $5.46^\circ / 0.008$, $12.33^\circ / 0.013$ and $7.12^\circ / 0.010$ for Car, Chair, and Plane, respectively (unit of translation is object diameter). This shows that even the canonicalized poses are not perfectly aligned and due to inpainting, our method is still able to learn from this poorly aligned data (particularly with respect to rotation). To further highlight the contribution of inpainting to this robustness, we add noise to the predicted IT-Net poses with rotations and translations sampled uniformly with a maximum displacement of $5^\circ / 0.01$, $10^\circ / 0.05$, and $15^\circ / 0.10$. Figure in Table 3a shows that without inpainting (in red), our method is extremely sensitive to alignment noise but with inpainting (in green), our method only degrades slightly with higher noise, and remains more accurate at all levels of noise than the baseline [6] with no noise added.

4.7 Impact of β in the Weighted Chamfer Distance loss

We present an analysis in Table 4 in which we train PointPnCNet with different values of β to understand the contribution of the second term in the asymmetric Weighted Chamfer Distance loss (Eqn. 1), \mathcal{L}_{wcd} .

From Table 4, we can observe that the optimal performance occurs at $\beta = 0.25$ across both the ShapeNet and KITTI datasets. Our intuition is that a larger value of β imposes a penalty for generating points in Y in the regions that were occluded in the input; this contradicts our goal of completing those missing regions. Nonetheless, setting $\beta = 0$ also leads to worse performance because the second term in Eqn. 1 is needed to (minimally) penalize the network for predicting points in Y that are far from the original partial point cloud X . Setting $\beta = 0.25$ provides the appropriate balance between these competing objectives. As explained in Section 3.3, this tradeoff only occurs for the global loss; the local loss uses a regional indicator that only applies the loss to regions for which we have ground truth information.

5 Conclusion

We propose a self-supervised method for point cloud completion via inpainting and random region removal that can be trained using only LiDAR-based partial point clouds. Our method produces significantly more accurate point cloud completions and outperforms the previous unsupervised methods on ShapeNet and Semantic KITTI. Through exhaustive ablation, we show the importance of each component of our method and the robustness to alignment errors. While the current method uses intersecting half-spaces defined by coordinate planes, other methods for point cloud partitioning can be explored in future work. We hope that our method will improve real-world 3D object understanding.

6 Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1849154, and the CMU Argo AI Center for Autonomous Vehicle Research.

| β | Airplane | Car | Chair | KITTI |
|---------|-------------|-------------|-------------|--------------|
| 0 | 2.10 | 2.63 | 3.02 | 0.132 |
| 0.25 | 1.66 | 2.48 | 2.70 | 0.095 |
| 0.5 | 2.31 | 3.00 | 3.28 | 0.121 |
| 0.75 | 2.59 | 3.50 | 3.78 | 0.142 |
| 1 | 3.83 | 4.72 | 4.95 | 0.196 |

Table 4: Performance analysis on different values of hyperparameter β used in Equation 1.

References

- [1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019.
- [2] 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.
- [3] Jingdao Chen, John Seon Keun Yi, Mark Kahoush, Erin S Cho, and Yong K Cho. Point cloud scene completion of obstructed building facades with generative adversarial inpainting. *Sensors*, 20(18):5029, 2020.
- [4] Zeqing Fu, Wei Hu, and Zongming Guo. Point cloud inpainting on graphs from non-local self-similarity. In *ICIP*, pages 2137–2141. IEEE, 2018.
- [5] Zeqing Fu, Wei Hu, and Zongming Guo. 3d dynamic point cloud inpainting via temporal consistency on graphs. In *ICME*, pages 1–6. IEEE, 2020.
- [6] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. Weakly-supervised 3d shape completion in the wild. pages 283–299, 2020.
- [7] Xin Hong, Pengfei Xiong, Renhe Ji, and Haoqiang Fan. Deep fusion network for image completion. In *ACM International Conference on Multimedia*, pages 2033–2042, 2019.
- [8] Wei Hu, Zeqing Fu, and Zongming Guo. Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting. *Transactions on Image Processing*, 28(8):4087–4100, 2019.
- [9] Zitian Huang, Yikuan Yu, Jiawen Xu, Feng Ni, and Xinyi Le. Pf-net: Point fractal network for 3d point cloud completion. In *CVPR*, pages 7662–7670, 2020.
- [10] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*, pages 2802–2812, 2018.
- [11] Hongyu Liu, Bin Jiang, Yibing Song, Wei Huang, and Chao Yang. Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. pages 725–741, 2020.
- [12] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *AAAI Conference on Artificial Intelligence*, pages 11596–11603, 2020.
- [13] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.
- [14] Lyne P Tchampi, Vineet Kosaraju, Hamid Rezaatofghi, Ian Reid, and Silvio Savarese. Topnet: Structural point cloud decoder. In *CVPR*, pages 383–392, 2019.
- [15] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. <https://github.com/open-mmlab/OpenPCDet>, 2020.

- [16] Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view consistency as supervisory signal for learning shape and pose prediction. In *CVPR*, pages 2897–2905, 2018.
- [17] Xiaogang Wang, Marcelo H Ang, and Gim Hee Lee. Point cloud completion by learning shape priors. pages 10719–10726, 2020.
- [18] Xiaogang Wang, Marcelo H Ang Jr, and Gim Hee Lee. Cascaded refinement network for point cloud completion. In *CVPR*, pages 790–799, 2020.
- [19] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Softpoolnet: Shape descriptor for point cloud completion and classification. pages 70–85, 2020.
- [20] Xin Wen, Tianyang Li, Zhizhong Han, and Yu-Shen Liu. Point cloud completion by skip-attention network with hierarchical folding. In *CVPR*, pages 1939–1948, 2020.
- [21] Haozhe Xie, Hongxun Yao, Shangchen Zhou, Jiageng Mao, Shengping Zhang, and Wenxiu Sun. Grnet: Gridding residual network for dense point cloud completion. pages 365–381, 2020.
- [22] Xinchun Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NeurIPS*, pages 1704–1712, 2016.
- [23] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *CVPR*, pages 5485–5493, 2017.
- [24] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *CVPR*, pages 5505–5514, 2018.
- [25] Yikuan Yu, Zitian Huang, Fei Li, Haodong Zhang, and Xinyi Le. Point encoder gan: A deep learning model for 3d point cloud inpainting. *Neurocomputing*, 384:192–199, 2020.
- [26] Wentao Yuan, David Held, Christoph Mertz, and Martial Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018.
- [27] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *3DV*, pages 728–737, 2018.
- [28] Xiaohang Zhan, Xingang Pan, Bo Dai, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised scene de-occlusion. In *CVPR*, June 2020.
- [29] Yifan Zhao, Jin Xie, Jianjun Qian, and Jian Yang. Pui-net: A point cloud upsampling and inpainting network. In *PRCV*, pages 328–340. Springer, 2020.
- [30] Rui Zhu, Hamed Kiani Galoogahi, Chaoyang Wang, and Simon Lucey. Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. In *ICCV*, pages 57–65, 2017.