MaskNet: A Fully-Convolutional Network to Estimate Inlier Points

Vinit Sarode[*], **Animesh Dhagat**[*], **Rangaprasad Arun Srivatsan**, **Nicolas Zevallos**, **Simon Lucey**, **Howie Choset**
Carnegie Mellon University
vinitsarode5@gmail.com, adhagat@andrew.cmu.edu

Abstract

Point clouds have grown in importance in the way computers perceive the world. From LIDAR sensors in autonomous cars and drones to the time of flight and stereo vision systems in our phones, point clouds are everywhere. Despite their ubiquity, point clouds in the real world are often missing points because of sensor limitations or occlusions, or contain extraneous points from sensor noise or artifacts. These problems challenge algorithms that require computing correspondences between a pair of point clouds. Therefore, this paper presents a fully-convolutional neural network that identifies which points in one point cloud are most similar (inliers) to the points in another. We show improvements in learning-based and classical point cloud registration approaches when retrofitted with our network. We demonstrate these improvements on synthetic and real-world datasets. Finally, our network produces impressive results on test datasets that were unseen during training, thus exhibiting generalizability. Code and videos are available at https://github.com/vinits5/masknet

1. Introduction

Point clouds are unordered sets of points in 3D space, usually describing the surface of an object or a scene. Recently they have been used to recognize [27], locate [10] and track objects [4] in a scene or be stitched together to form more complete point clouds [12]. When being used in cluttered spaces, point clouds describe only parts of the object/scene that are visible to the sensor and not covered by occlusions. In addition, sensor noise, reflective surfaces, or other artifacts can sometimes produce points in the point cloud which do not correspond to any surface on the object or in the scene. Point clouds with missing data as well as those having extraneous points pose challenges to point cloud processing algorithms such as registration [5] [2] and tracking [4]. As a result, it is important to identify which points need to be considered 'inliers' and which points need to be discarded and deemed as 'outliers'.

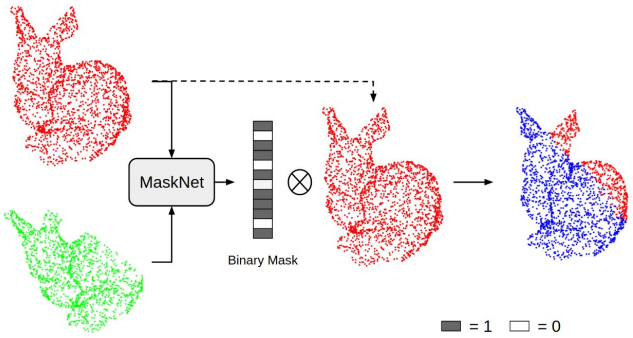


Figure 1: MaskNet estimating inliers (shown on the right in blue) for a pair of point clouds (shown on the left). MaskNet finds a Boolean vector *mask* that only retains inlier points from point cloud in red which most closely approximate the shape of the point cloud in green.

Prior works on inlier/outlier detection have considered taking random consensus (RANSAC) [1], finding geometric primitives such as planes and cylinders [21, 38], finding most probable correspondences [6, 7], or using globally optimal alignment techniques [44, 18, 36]. However, most of these works do not scale well with the number of points in the point clouds, and therefore are generally computationally expensive for most real-time applications.

In this work, we leverage recent advancements in deep learning-based point cloud representations [27], to perform learning-based inlier estimation. Given a template point cloud and a source point cloud (shown by red and green points in Fig. 1 respectively), we create a network that is trained to identify which points from the template are inliers, so that these points closely describe the same part of the object/scene geometry as the source point cloud (the blue points in Fig. 1 are the inliers). In other words, the network learns to 'mask-out' outliers from the template point cloud, hence we call our approach MaskNet.

We evaluate MaskNet on synthetic [42] as well as real-world datasets [51, 3] and compare with state-of-the-art

[*]equal contribution
Correspondence to {vinitsarode5, animeshdhagat}@gmail.com

2475-7888/20/$31.00 ©2020 IEEE
DOI 10.1109/3DV50981.2020.00113

approaches. Further, we demonstrate the benefit of using MaskNet as a preprocessing step for point cloud registration algorithms. Removing outliers in particular improves the registration accuracy of popular deep learning-based approaches such as PointNetLK [2], and deep closest point (DCP) [39]. Finally, MaskNet shows remarkable generalization within and across datasets without the need for additional fine tuning.

In this paper, we review prior work and define the problem as - finding inlier points in a given pair of point clouds describing the same object, where one of them (source) has missing points compared to the other (template) - in Sec. 2. In Sec. 3 we describe our approach for finding inlier points and in Sec. 4 we discuss the veracity of our approach through experiments on synthetic and real-world datasets.

2. Related Work

Classical Inlier/Outlier Estimation Approaches Given a pair of point clouds, inlier estimation determines which points in one point cloud are most similar to the points in another. Estimating inlier point-pairs is a well documented problem that finds application in registration [11], object detection [37], and flow estimation [23]. Locally optimal methods achieve this by finding all point-pairs between point clouds, and retaining only the inliers by using algorithms such as RANSAC [35, 26]. They iteratively sample random point-pairs until the inlier point-pairs minimize the misalignment between the point clouds. However, these methods are very time consuming, especially when the initial misalignment between the point clouds is very high. RANSAC-based approaches also exhibit slow convergence and low accuracy with large number of outliers.

Other works have posed inlier estimation as an optimization problem and used branch-and-bound techniques [45], Gaussian mixture models [25], mixed integer programming (MIP) [17], semi-definite programming [9] and maximal clique selection [8]. These methods provide guarantees on robust detection of all the inliers. With the exception of TEASER++ [43] and [14], most of these conventional optimization-based approaches are computationally expensive and cannot be deployed for any practical realtime applications.

A fast-growing vein in inlier estimation has been through finding point-based features, and using them to significantly reduce the number of outliers. Local geometry-based features are computed from a combination of 3D point coordinates and surface normals [7, 32]. Similarly other works have introduced hand-crafted features [31]. Compared to globally optimal approaches, feature-based inlier detection approaches are computationally faster, but more prone to failure as they are sensitive to the specific choice of geometric features that are chosen.

Learning in Point Clouds Inspired by classical feature-based inlier estimation techniques, Qi et al [27] introduced PointNet, a learning-based approach to learning task specific features from point clouds. PointNet [27] paved the way for using unordered point clouds in a learning paradigm. Recent works such as PointNet++ [28], DGCNN [41],Deep sets [50], PointCNN [22], point pillars [19] and PCPNet [16] improve the performance of PointNet by generating features that consider local neighbourhoods of points. More recently, a point cleaning network [29] was developed based on the PCPNet. PointCleanNet estimates robust local features and use this information to denoise the point cloud. Another variant of learning-based inlier estimation includes SampleNet [20, 13] which finds only a small subset of inliers.

In addition to supervised learning techniques such as PointNet, other self-supervised [34], and unsupervised [52] feature-learning techniques have been introduced. While all these algorithms continue to remain locally optimal, they are faster than conventional geometry-based methods, owing to computationally inexpensive matrix operations. However, as the point clouds become denser, the neural networks suffer as much as their hand-crafted counterparts in terms of computational complexity.

Point Cloud Registration An important application in computer vision that is impacted heavily by the presence of outliers in the point cloud data is that of point cloud registration. Outliers adversely affect both classical [5, 30, 24, 1] and deep learning-based registration methods [2, 33, 39]. Most of the popular deep learning registration-based methods such as PointNetLK [2], PCRNet [33] and DCP [39] work on the assumption that all points in the point clouds are inliers by default. Naturally, they perform poorly when one of the point clouds has missing points, as in the case of partial point clouds. Deep learning-based methods developed specifically to handle partial point cloud registration such as PRNet [40], and RPM-Net [47] have deeper network architectures that learn to handle partial point clouds by explicitly estimating point-correspondences. Unfortunately, these methods do not scale well with the increasing number of points, as the size of the correspondence parameters predicted by the network grows polynomially. They also rely heavily on being explicitly trained with partial point cloud data. Other methods such as deep global registration (DGR) [11] and Multi-view registration [15] use neural networks to filter outliers from a given set of possible correspondences. Such methods are limited to only estimating a subset of inlier points.

3. MaskNet

In this section we discuss the mathematical formulation of MaskNet and describe how MaskNet can be used to im-

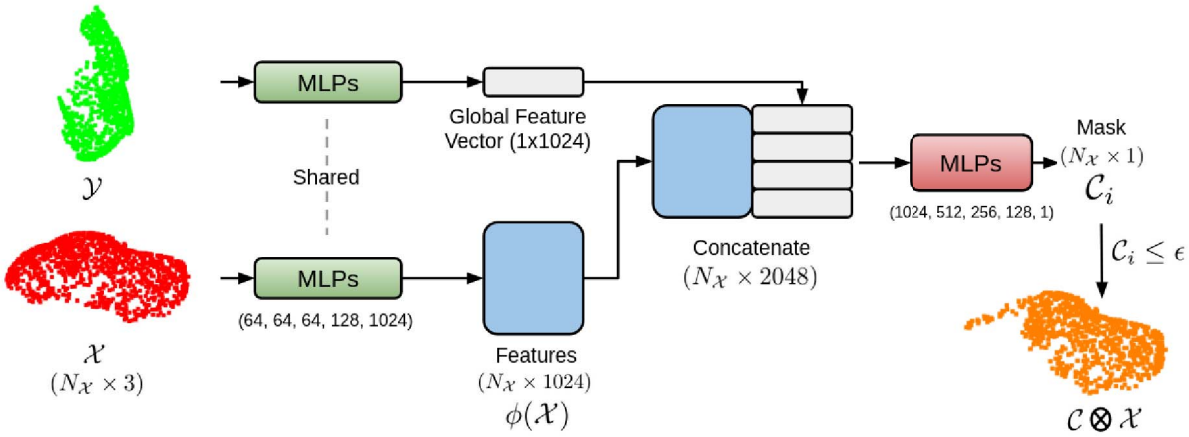


Figure 2: MaskNet Architecture: The model consists of five MLPs of size (64, 64, 64, 128, 1024). The source ($\mathcal{Y}$) and template ($\mathcal{X}$) point clouds are sent as input through a twin set of MLPs, arranged in a Siamese architecture. Using a max-pooling function, we obtain global features for the source point cloud. Weights are shared between MLPs. These features are concatenated and provided as an input to five fully convolutional layers of size (1024, 512, 256, 128), and an output layer of size 1. A sigmoid function is applied on the output to produce the mask ($\mathcal{C}$).

prove the task of point cloud registration.

3.1. Mathematical Formulation

Let us consider two point clouds $\mathcal{X}$ and $\mathcal{Y}$, where $\mathcal{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_{N_\mathcal{X}}] \in \mathbb{R}^{3\times N_\mathcal{X}}$, $\mathcal{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{N_y}] \in \mathbb{R}^{3\times N_y}$, and $N_\mathcal{X} \geq N_y$. Let us consider a binary vector $\mathcal{C} \in \mathbb{R}^{N_x \times 1}$ ($\mathcal{C}_i \in \{0, 1\} \quad \forall i \in [1, ..., N_\mathcal{X}]$) and an operator $\otimes$, defined as follows:

$$\mathcal{X}_1 = \mathcal{C} \otimes \mathcal{X}, \tag{1}$$

where $\mathcal{X}_1 \subseteq \mathcal{X}$ and $\boldsymbol{x}_i \in \mathcal{X}_1$ if $\mathcal{C}_i = 1$. In other words, the set $\mathcal{X}_1$ contains a subset of points from the set $\mathcal{X}$ corresponding to $\mathcal{C}_i = 1$. Effectively the binary vector $\mathcal{C}$ acts as a mask and removes points from the set $\mathcal{X}$. We aim to mask the points from $\mathcal{X}$ such that the resulting point cloud $\mathcal{X}_1$ represents the same overall geometry as $\mathcal{Y}$. We use the K-dimensional vector encoding of PointNet [27], $\phi : \mathbb{R}^{3\times N} \rightarrow \mathbb{R}^K$ as the metric for comparing $\mathcal{X}_1$ and $\mathcal{Y}$. From PointNetLK [2] we know that the PointNet encoding can be sensitive to large misalignment between the point clouds and hence the following condition holds:

$$\phi(\mathcal{C} \otimes \mathcal{X}) = \phi(\mathbf{R}\mathcal{Y} + \mathbf{t}), \tag{2}$$

where $\mathbf{R} \in SO(3)$ is the rotation and $\mathbf{t} \in \mathbb{R}^3$ is the translation between $\mathcal{X}$ and $\mathcal{Y}$. However, since $\mathbf{R}$ and $\mathbf{t}$ are unknown, we use the following weaker condition to relate $\mathcal{X}_1$ and $\mathcal{Y}$:

$$\phi(\mathcal{C} \otimes \mathcal{X}) \approx \phi(\mathcal{Y}). \tag{3}$$

We show later in Fig. 4, that it is possible to iteratively improve the estimate of $\mathcal{C}$. We substitute $\mathcal{C}$ from Eq. 3 into Eq. 2 and find $\mathbf{R}$ and $\mathbf{t}$ as described in Sec. 3.4. We then replace $\mathcal{Y}$ in Eq. 3 with $\mathbf{R}\mathcal{Y}+\mathbf{t}$ and repeat this process until convergence. We learn $\mathcal{C}$ using MaskNet, $f(\cdot)$, as follows:

$$\mathcal{C} = f(\phi(\mathcal{X}), \phi(\mathcal{Y})). \tag{4}$$

Discrete outputs from a neural network during training induce discontinuities during back propagation and prevents MaskNet from producing a binary vector as output. Instead, MaskNet predicts a vector $\mathcal{C}^*$ such that $\mathcal{C}^*_i \in [0, 1] \quad \forall i \in [1, ..., N_\mathcal{X}]$. During evaluation, a binary vector is computed by applying a threshold (ϵ) such that,

$$\mathcal{C} = \left\{1, \quad \text{if} \quad \mathcal{C}^*_i \geq \epsilon \right\} \quad \forall i \in [1, N_\mathcal{X}] \tag{5}$$

3.2. Point Feature Encoding

PointNet uses a set of multi-layer perceptrons (MLPs) to encode each 3D point in a higher dimensional feature vector. Similar to PointNet, MaskNet (see Fig. 2) uses a set of MLPs of size (64, 64, 64, 128, 1024) to estimate the feature vectors ($\phi(\mathcal{X})$, $\phi(\mathcal{Y})$) of input point clouds. Prior works such as PCN [49] and FoldingNet [46] have shown the efficacy of PointNet feature vectors for point cloud completion by comparing the features of partial input point cloud with the features of 2D point grids to create local patches of the complete output point cloud. Taking inspiration from these methods, MaskNet estimates the inliers by concatenating the feature vectors of $\mathcal{X}$ and $\mathcal{Y}$ as follows,

$$\mathcal{C}^* = sigmoid\left(h\left(\left[\phi(\mathcal{X}) \quad \begin{bmatrix} g(\phi(\mathcal{Y})) \\ \vdots \\ g(\phi(\mathcal{Y})) \end{bmatrix}_{N_\mathcal{X}\times 2048}\right)\right) \tag{6}$$

where $h(\cdot)$ represents a set of MLPs of size (1024, 512, 256, 128, 1) used to create the mask ($\mathcal{C}$), $g(\cdot)$ represents the symmetry function (max-pooling operation) where $g(\phi(\mathcal{Y})) \in \mathbb{R}^{1\times1024}$, and $\phi(\mathcal{X}) \in \mathbb{R}^{N_\mathcal{X}\times1024}$. In the last layer of $h(\cdot)$, we use a sigmoid activation function to enforce $\mathcal{C}_i^* \in [0,1] \quad \forall i \in [1,...,N_\mathcal{X}]$.

3.3. Loss Function

During training, the loss function of MaskNet measures the difference between the predicted mask and the ground truth mask, defined as a mean squared error:

$$Loss = \sum_{i=1}^{N_\mathcal{X}} ||\mathbf{C_i} - \mathbf{C_i}^{gt}||_2, \tag{7}$$

where $\mathbf{C}$ represents the mask predicted by network and $\mathbf{C}^{gt}$ represents the ground truth mask.

3.4. MaskNet for Registration

MaskNet can be used as an add-on module with any registration algorithm to estimate rotation $\mathbf{R}$ and translation $\mathbf{t}$ between a pair of point clouds (see Fig. 3). For instance, upon finding the mask $\mathcal{C}$ using MaskNet, we can substitute it into Eq. 2 and iteratively estimate $\mathbf{R}$ and $\mathbf{t}$ using the Lucas-Kanade algorithm as in the case of PointNetLK [2]. Alternatively, one could apply $\mathcal{C}$ to $\mathcal{X}$ to obtain $\mathcal{X}_1 = \mathcal{C} \otimes \mathcal{X}$. Following this, $\mathcal{X}_1$ and $\mathcal{Y}$ can be registered using PCRNet [33], DCP [39], ICP [5], RPM-Net [47], and PRNet [40].

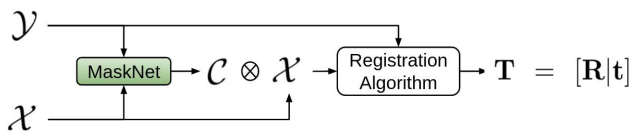


Figure 3: Registration pipeline: A template point cloud, $\mathcal{X}$ and a source point cloud, $\mathcal{Y}$, are provided to a pre-trained MaskNet. The mask, $\mathcal{C}$ obtained from the network is applied to the template to obtain a partial point cloud, $\mathcal{X}_1 = \mathcal{C} \otimes \mathcal{X}$, which is the set of inlier points. $\mathcal{X}_1$ and $\mathcal{Y}$ are provided to a registration algorithm to produce $\mathbf{R}$ & $\mathbf{t}$. In the case of learning-based registration algorithms, a pre-trained network can be used.

3.5. MaskNet for Denoising

MaskNet can also be used to detect noise and remove outliers from a given point cloud $\mathcal{Y}$, using a noise-free template point cloud of the same category, $\mathcal{X}$, by a simple reformulation. Rather than removing points from $\mathcal{X}$, $\mathcal{C}$ can be applied as a mask to remove outliers present in the set $\mathcal{Y}$. The resulting point cloud $\mathcal{Y}_1 = \mathcal{C} \otimes \mathcal{Y}$ represents the same overall geometry as $\mathcal{X}$. In other words, we flip $\mathcal{X}$ and $\mathcal{Y}$ in Eq. 3 as follows,

$$\phi(\mathcal{X}) \approx \phi(\mathcal{C} \otimes \mathcal{Y}). \tag{8}$$

Similar to Eq. 4, we learn $\mathcal{C}$ using MaskNet.

4. Experiments

To validate the ability of MaskNet to predict inlier points, Sec. 4.1 discusses the use of precision as a metric for comparing predicted inlier points to the ground truth inlier points. We then use this precision metric to compare MaskNet with other methods such as RPM-Net and PRNet which explicitly find inlier points based on correspondences. In Sec. 4.2 and Sec. 4.3 we demonstrate the effectiveness of using inlier estimation for the tasks of point cloud denoising and registration of partial point clouds, respectively, on a synthetic dataset [42]. The generalizability of MaskNet is demonstrated on synthetic as well as real-world datasets (S3DIS [3] and 3DMatch [51]) in Sec. 4.4 and Sec. 4.5.

4.1. Inlier detection

In this section we use the ModelNet40 dataset [42] consisting of 9843 meshed CAD models of 40 different object categories for training and 2486 models for evaluation. We follow the protocol used in [27] to create a set of point clouds where 1024 points are uniformly sampled from each mesh model and are scaled to fit in a unit sphere. A random transformation is applied to each of these point clouds with a rotation in $[0, 45]^\circ$ and translation in [-1, 1] units along each axis to create pairs of point clouds. Partial scans of point clouds are obtained by simulating a physical sensor model at a chosen view point. We do this by selecting a random view point and then choosing a set of points from the surface of the object facing the sensor. To obtain these points, we compute nearest neighbors on the surface of object from the view point. A ground truth Boolean vector is computed using the indices of nearest neighbours having 1 for selected neighbours and 0 for other points. During training, this Boolean vector is used as a supervision on the predicted mask.

MaskNet is trained for 300 epochs using a learning rate of 10^{-4} and batch size 32. The network parameters are updated with Adam Optimizer on a single NVIDIA GeForce GTX 1070 GPU and an Intel Core i7 CPU at 4.0GHz. We follow the same settings of training for all the experiments.

Prior works [2, 33, 48] show the sensitivity of PointNet to large initial misalignment between a given pair of point clouds. Taking this into consideration, we evaluate the mask in an iterative manner where we first compute mask $\mathcal{C}$ and then estimate $\mathbf{R}$ and $\mathbf{t}$ by substituting $\mathcal{C}$ in Eq. 2. We train MaskNet to estimate mask ($\mathcal{C}$) and separately train PointNetLK to estimate registration parameters between a pair of

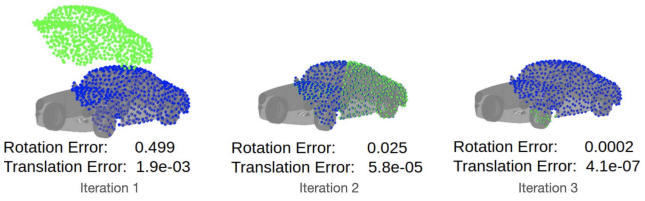


Figure 4: Qualitative results for Section 4.1. For each iteration, we first use MaskNet to compute mask and then PointNetLK to compute registration parameters. The source point cloud is shown in green color, template in gray color and point cloud registered with PointNetLK in blue color. With every iteration, we observe a continuous decrease in rotation as well as translation error.

point clouds from ModelNet40 dataset using all 40 object categories. Fig. 4 shows the results of estimated registration parameters for each iteration. We observe a monotonic decrease in rotation as well as translation error in each iteration. Even though $\mathcal{C}$, $\mathbf{R}$, and $\mathbf{t}$ improve with every iteration, we observe results better than the state-of-art partial registration methods in the first iteration with minimum computational effort. For that reason, in all the further experiments we only report results for a single iteration.

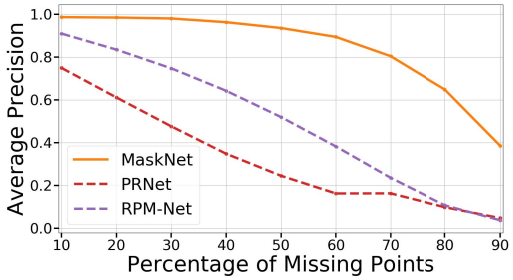


Figure 5: Results for section 4.1. The plot shows the precision of predicted inliers on the y-axis and percentage of missing points in the partial point cloud on x-axis. Mask produced by MaskNet and correspondences produced by PRNet, RPM-Net are used to compute inliers. Even though MaskNet is trained with 30 percent missing points, we observe a reasonably good performance in the range of 10 to 60% missing points as compared to other algorithms.

MaskNet computes a mask giving the probabilities for each point in the full point cloud for having a similar point in the partial ones. Since each element in the mask can be either 0 or 1, it becomes a binary classification problem for each point. We use the "precision metric" which is a common metric in binary classification, to quantify the efficiency of our method. Inlier points having $\mathcal{C}_i^{gt} = 1$ are termed as true positives (TP) and those having $\mathcal{C}_i^{gt} = 0$ are termed as false positives (FP).

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

In this experiment, we train MaskNet, PRNet and RPM-Net on the entire ModelNet40 dataset including all 40 object categories with 30% missing points in partial point clouds. Ground truth mask and the prediction of MaskNet is used to estimate TP and FP. PRNet and RPM-Net provide correspondence matrix as output which are used to find inlier point pairs (TP) and outliers. To evaluate our method, we compute precision using a test set with different percentages of missing points in input point clouds as shown in Fig. 5. We observe that MaskNet performs reasonably well within a range of 10% to 60% of missing data when compared to PRNet and RPM-Net. The decreasing precision of PRNet and RPMNet (as opposed to MaskNet) is attributed to the sensitiveness of Horn's method that is used to find pose parameters from incorrect correspondences.
Fig. 6 shows the inlier points (blue colored) computed using the predicted mask for the given partial point cloud (green colored) and a full point cloud (gray CAD model). The results clearly indicate the accuracy of the mask from the similarity in the geometric shape of blue and green point clouds.

(a) Seen Category (b) Unseen Category (c) Unseen Category

Figure 6: Results showing the template point cloud $\mathcal{X}$ visualized as a CAD model, and source point cloud $\mathcal{Y}$ as green colored points. Blue point cloud represents $\mathcal{X}_1 = \mathcal{C} \otimes \mathcal{X}$ and points from $\mathcal{X}$ having $\mathcal{C}_i = 0$ are shown in red color.

4.2. Point Cloud Denoising

In this section, we show the use of MaskNet to detect the outliers in a given point cloud by comparing it with a standard reference point cloud of the same object category. We train MaskNet with one of the input point clouds from the ModelNet40 dataset having 10% points added as outliers at random locations in 3D space. While training, the ground truth mask $\mathcal{C}^{gt}$ contains 0 for all indices that are outlier points and 1 for inliers. Our training dataset consists of all point clouds from the training set of ModelNet40 while our testing dataset has 1000 randomly chosen point clouds from the test set.

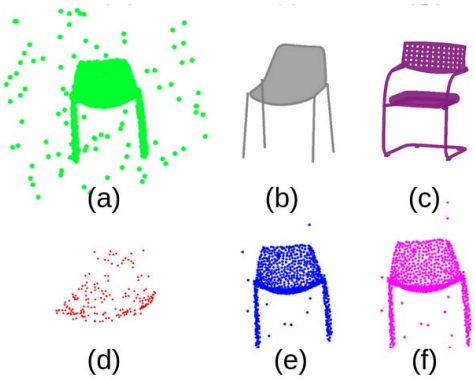


Figure 7: Denoising with different templates models. (a) Noisy source, (b) and (c) are different models of category 'chair'. (d) Result of PointCleanNet [29], (e) and (f) are results of MaskNet when using template models (b) and (c) respectively with source (a).

Fig. 7 (e,f) shows MaskNet's ability to use different template models (of the same category as that of the noisy source point cloud) to denoise the source point cloud. Fig. 7 (a,d) also shows the performance of PointCleanNet [29] on the same noisy source point cloud. While PointCleanNet removes most outlier points, it also removes a fair bit of inliers. In Fig. 8, we show that retrofitting PointNetLK with MaskNet improves its performance significantly. Even 3% to 4% percentage of outliers increases the rotation error by $\approx 70^\circ$ for PointNetLK compared to PointNetLK when retrofitted with MaskNet (referred to as Mask-PointNetLK). A similar trend is observed for translation error. This clearly shows the advantage of denoising the point cloud before registration using our network.

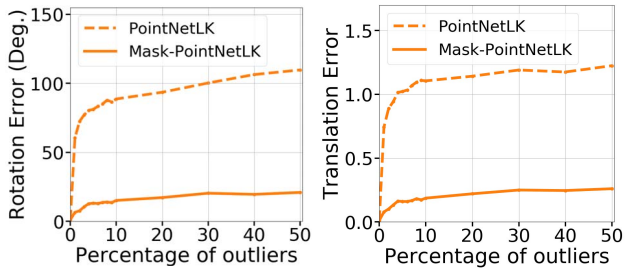


Figure 8: Results for Section 4.2 showing the effect of number of outliers for PointNetLK and its retrofitted version with MaskNet using rotation and translation errors as a metric for comparison. MaskNet is trained with only 10% outliers but shows a great improvement in registration performance of Mask-PointNetLK.

4.3. Partial Point Cloud Registration

We use partial point cloud registration as a downstream task to show the usefulness of MaskNet. The registration pipeline in this experiment is as shown in Fig. 3. The pipeline consists of two stages – (1) the MaskNet estimates the inlier points from given input point clouds, and (2) a standard registration algorithm (classical or deep learning-based) estimates the registration parameters. Training of neural networks in both the stages takes place independently. We train MaskNet with the entire ModelNet40 dataset and conduct experiments using ICP [5], PointNetLK [2], DCP [39] and PRNet [40]. Pre-trained models of PointNetLK, PRNet and DCP are used in this pipeline. As ICP and PointNetLK are iterative methods, we allow maximum of 10 iterations for all the following experiments.

In this experiment, we compare ICP, PointNetLK, PRNet and DCP algorithms with their MaskNet equipped versions hereafter referred to as Mask-ICP, Mask-PointNetLK, Mask-PRNet and Mask-DCP, respectively. Experimental settings to create evaluation dataset are same as described in Sec. 4.1 and the dataset consists of 1000 pairs of point clouds. Mean rotation and translation errors for different initial misalignments are reported in the Fig. 9. PointNetLK and DCP perform poorly with the partial point clouds. We see a notable performance improvement in the error metrics of Mask-PointNetLK and Mask-DCP as compared to PointNetLK and DCP respectively. As ICP is a shape agnostic classical registration method, Mask-ICP shows a marginal performance improvement over ICP. Even though PRNet is specifically designed for partial point cloud registration, we observe an improvement in the performance when using MaskNet. Qualitative results are shown in Fig. 10. Average computational time for ICP, PointNetLK, PRNet and DCP is 6.82 ms, 52.30 ms, 68.52 ms and 23.99 ms. On the other hand, Mask-ICP, Mask-PointNetLK, Mask-PRNet and Mask-DCP requires 9.83 ms, 58.15 ms, 82.18 ms and 24.68 ms showing a negligible increase in time complexity with the addition of MaskNet.

In conclusion, MaskNet proves to be an efficient method to deal with partial point cloud registration when used with any existing registration algorithm. An interesting property of this pipeline is that both the MaskNet and the registration algorithm are independent of each other and can be trained separately. This reduces the computational efforts required to retrain any registration network.

4.4. Generalization

We split ModelNet40 dataset into two parts – models of first 20 categories for training (seen categories) and the last 20 categories for evaluation (unseen categories). Point cloud datasets are created for both the categories using the protocol described in Sec. 4.1. MaskNet is trained only using the seen categories and evaluated using the point clouds

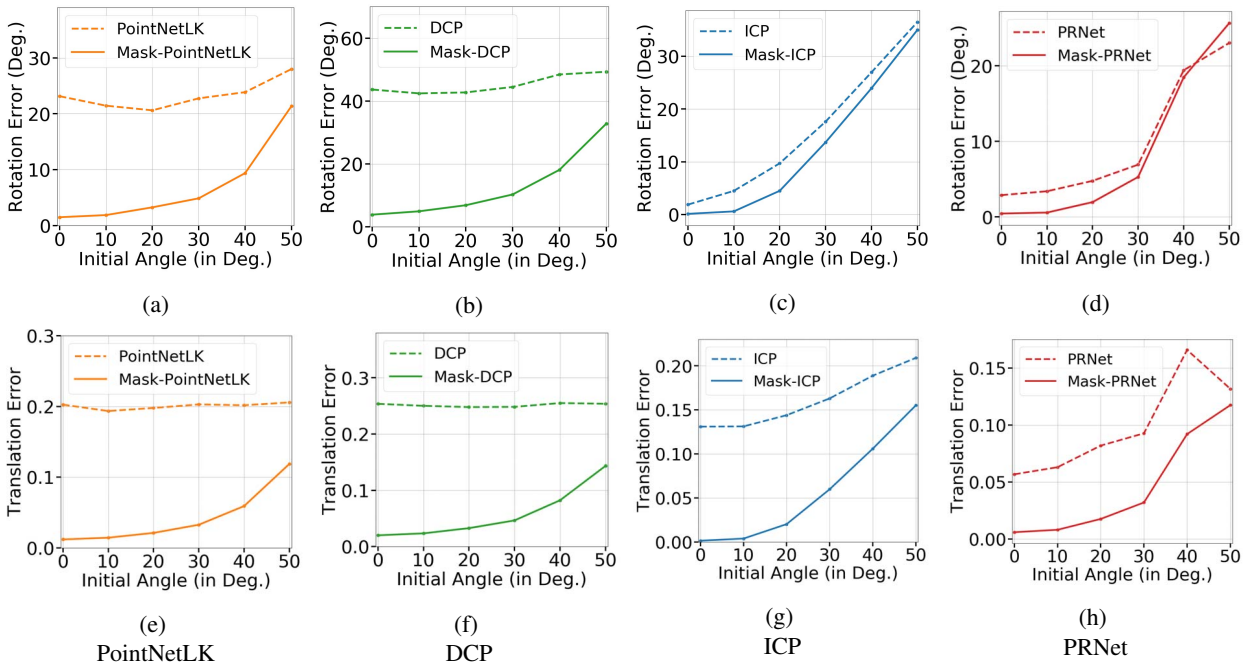


Figure 9: Results for section 4.3. Effect of initial misalignment on rotation error (top row) and translation error (bottom row) for X and Mask-X [X: PointNetLK, DCP, ICP]. Figures (a) and (b) show an improvement of approximately 25° in rotation error due to the use of MaskNet with PointNetLK and DCP. Similarly, improvement in translation errors can be observed in (e) and (f). Figures (c) and (g) show that Mask-ICP has a marginal improvement in registration over ICP. Similarly, figures (d) and (h) show improvement in registration using Mask-PRNet over PRNet.

from unseen categories. Qualitative results of evaluated mask on unseen categories is shown in Fig. 6, where blue points indicate the estimated inlier points and red points show the outliers. We observe that the geometric shape of green and blue point clouds are in good accordance, which indicates the prediction of an accurate mask even in case of unseen categories.

We further evaluate MaskNet by estimating the registration between partial point clouds. We choose a test-set of 1000 point clouds each from seen as well as unseen categories and perturb them with various initial misalignment. We then compute the mean rotation error and mean translation error for both the datasets after registering them with Mask-PointNetLK. The small difference in the performance of Mask-PointNetLK for seen versus the unseen categories in Fig. 11, clearly indicates the generalization ability of MaskNet across different object categories.

4.5. Real-world Data

MaskNet shows good performance for zero shot transfer of learning across the S3DIS [3] and 3DMatch datasets [51]. S3DIS contains a single large point cloud consisting of various offices, conference rooms, kitchen, etc. We divided this large point cloud into smaller point clouds and created a processed-S3DIS dataset. We use this processed-S3DIS

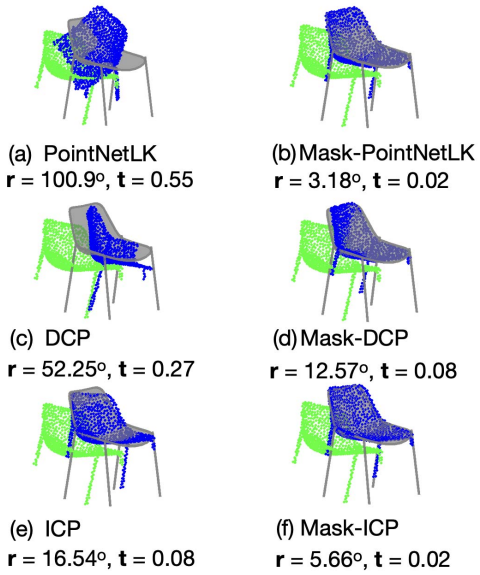
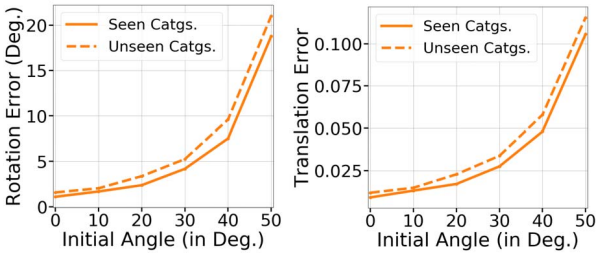


Figure 10: Improvement in registration through MaskNet. The left column shows registration by 3 methods and the right column shows same three methods when augmented by MaskNet. **r**: Rotation Error, **t**: Translation Error



(a) (b)

Figure 11: Results showing the generalization of MaskNet by training the network with 20 object categories (Seen) from ModelNet40 dataset and testing with 20 different object categories (Unseen). The above plots show that there is a minimal difference in the performance of Mask-PointNetLK for seen and unseen categories. This clearly indicates the generalization capability of MaskNet across unseen object categories.

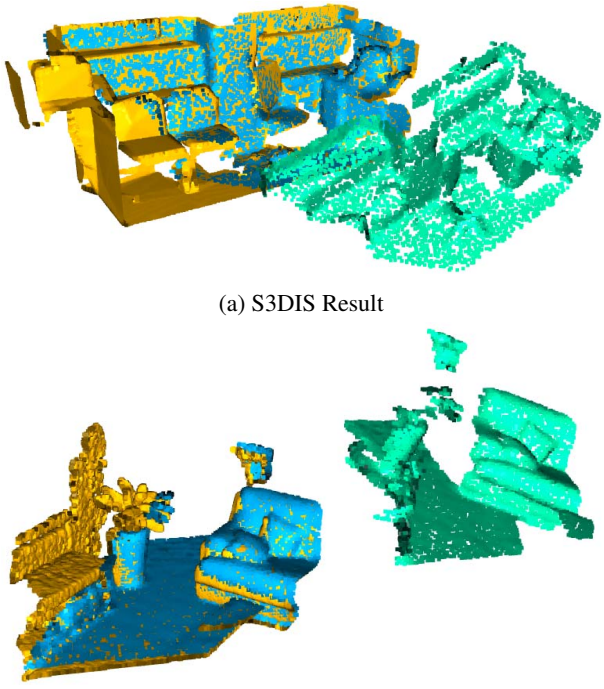
dataset to train MaskNet, and tested using point clouds present in the 3DMatch dataset. We observe that MaskNet trained on S3DIS performs equally well on 3DMatch dataset without the need of any additional fine-tuning. This can be clearly observed in Fig. 12, where yellow and green point clouds are the input to Mask-PointNetLK and blue point cloud shows the registered point cloud.MaskNet was also tested on point clouds of 3D printed objects, obtained from a RealSense sensor. Qualitative results are shown in Fig. 13.

5. Conclusion & Future Work

We offer a new learning-based approach, MaskNet, for determining inlier points in a given pair of point clouds by computing a *mask*. Our approach has a higher accuracy of finding inliers compared to existing learning-based inlier estimation methods which are also computationally expensive. We demonstrate through experiments, that MaskNet – *(a)* augments the ability of existing classical and deep learning-based registration methods to better deal with partial point clouds and outliers, *(b)* can be used to reject noise, and *(c)* generalizes to object categories that it was not trained on.

While we currently use a PointNet encoding in MaskNet, in the future we could replace PointNet with other feature descriptors that are less sensitive to noise, and are invariant to pose transformations. A natural next step, from the perspective of real-world application, is to remove supervision on ground truth masks - as unsupervised networks tend to generalize far better than supervised networks. In addition, this would allow us to train directly on real world datasets without the need to hand label the inlier points.

MaskNet is currently limited to removing points from only one of the input point clouds. A logical extension would be to estimate two sets of inlier points - one for each point cloud - in a given point cloud pair. This can be helpful in tasks involving stitching point clouds for scene reconstruction and SLAM applications. MaskNet offers a starting point for further development of such tasks and applications.



(a) S3DIS Result

(b) 3DMatch Result

Figure 12: Qualitative results for real world 3D scans. Green is the partial point cloud ($\mathcal{Y}$), yellow is the CAD model from which a full point cloud is sampled uniformly ($\mathcal{X}$), and blue point cloud is the result of Mask-PointNetLK registration pipeline.

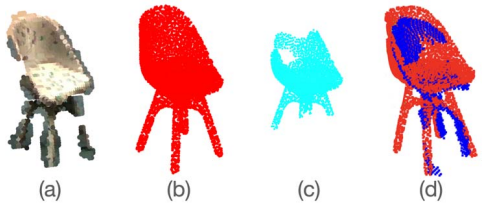


Figure 13: (a) Point cloud obtained from a RealSense sensor. (b) Template point cloud of the chair (red), (c) Result of MaskNet (cyan), (d) blue point cloud is the result of Mask-PointNetLK.

References

[1] D. Aiger, N. J. Mitra, and D. Cohen-Or. 4-points congruent sets for robust surface registration. *ACM Transactions on*

Graphics, 27(3):#85, 1–10, 2008. 1, 2

[2] Y. Aoki, H. Goforth, R. Arun Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1, 2, 3, 4, 6

[3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 1, 4, 7

[4] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. In *ECCV Workshops*, 2016. 1

[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. 1, 2, 4, 6

[6] S. D. Billings, E. M. Boctor, and R. H. Taylor. Iterative most-likely point registration (imlp): A robust algorithm for computing optimal shape alignment. *PloS one*, 10(3):e0117688, 2015. 1

[7] Á. P. Bustos and T.-J. Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2868–2882, 2017. 1, 2

[8] A. P. Bustos, T.-J. Chin, F. Neumann, T. Friedrich, and M. Katzmann. A practical maximum clique algorithm for matching with pairwise constraints. *arXiv preprint arXiv:1902.01534*, 2019. 2

[9] K. N. Chaudhury, Y. Khoo, and A. Singer. Global registration of multiple point clouds using semidefinite programming. *SIAM Journal on Optimization*, 25(1):468–501, 2015. 2

[10] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6526–6534, 2017. 1

[11] C. Choy, W. Dong, and V. Koltun. Deep global registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[12] J. Chu and C. mei Nie. Multi-view point clouds registration and stitching based on sift feature. *2011 3rd International Conference on Computer Research and Development*, 1:274–278, 2011. 1

[13] O. Dovrat, I. Lang, and S. Avidan. Learning to sample. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. 2

[14] N. Dym and S. Z. Kovalsky. Linearly converging quasi branch and bound algorithms for global rigid registration. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1628–1636, 2019. 2

[15] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal. Learning multiview 3d point cloud registration. In *International conference on computer vision and pattern recognition (CVPR)*, 2020. 2

[16] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra. PCPNet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018. 2

[17] G. Izatt, H. Dai, and R. Tedrake. Globally optimal object pose estimation in point clouds with mixed-integer programming. In *ISRR*, 2017. 2

[18] G. Izatt, H. Dai, and R. Tedrake. Globally optimal object pose estimation in point clouds with mixed-integer programming. In *Robotics Research*, pages 695–710. Springer, 2020. 1

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2

[20] I. Lang, A. Manor, and S. Avidan. Samplenet: Differentiable point cloud sampling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7578–7588, 2020. 2

[21] L. Li, F. Yang, H. Zhu, D. Li, Y. Li, and L. Tang. An improved ransac for 3d point cloud plane segmentation based on normal distribution transformation cells. *Remote Sensing*, 9(5):433, 2017. 1

[22] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018. 2

[23] X. Liu, C. R. Qi, and L. Guibas. Flownet3d: Learning scene flow in 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. 2

[24] N. Mellado, D. Aiger, and N. J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Comput. Graph. Forum*, 33:205–215, 2014. 2

[25] A. Myronenko and X. Song. Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, 32(12):2262–2275, 2010. 2

[26] D. S. Pankaj and R. R. Nidamanuri. A robust estimation technique for 3d point cloud registration. *Image Analysis & Stereology*, 35(1):15–28, 2016. 2

[27] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1, 2, 3, 4

[28] 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 5099–5108, 2017. 2

[29] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, pages 185–203. Wiley Online Library, 2020. 2, 6

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 2

[31] B. R. Rusu, C. Z. Marton, N. Blodow, and M. Beetz. Learning informative point classes for the acquisition of object model maps. *ICARCV*, pages 643–650, 2008. 2

[32] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. *2009 IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009. 2

[33] V. Sarode, X. Li, H. Goforth, Y. Aoki, R. Arun Srivatsan, S. Lucey, and H. Choset. Pcrnet: Point cloud registration network using pointnet encoding. Aug 2019. 2, 4

[34] J. Sauder and B. Sievers. Context prediction for unsupervised deep learning on point clouds. *CoRR*, abs/1901.08396, 2019. 2

[35] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. In *Computer graphics forum*, volume 26, pages 214–226. Wiley Online Library, 2007. 2

[36] R. A. Srivatsan, T. Zodage, and H. Choset. Globally optimal registration of noisy point clouds. *arXiv preprint arXiv:1908.08162*, 2019. 1

[37] A. Sveier, A. L. Kleppe, L. Tingelstad, and O. Egeland. Object detection in point clouds using conformal geometric algebra. *Advances in Applied Clifford Algebras*, 27:1961–1976, 2017. 2

[38] F. Tombari, S. Salti, and L. di Stefano. Unique signatures of histograms for local surface description. In *ECCV*, 2010. 1

[39] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 2, 4, 6

[40] Y. Wang and J. M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *33rd Conference on Neural Information Processing Systems (To appear)*, 2019. 2, 4, 6

[41] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2

[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 4

[43] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and certifiable point cloud registration. *arXiv preprint arXiv:2001.07715*, 2020. 2

[44] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2016. 1

[45] J. Yang, H. Li, and Y. Jia. Go-icp: Solving 3d registration efficiently and globally optimally. *2013 IEEE International Conference on Computer Vision*, pages 1457–1464, 2013. 2

[46] Y. Yang, C. Feng, Y. Shen, and D. Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2018. 3

[47] Z. J. Yew and G. H. Lee. Rpm-net: Robust point matching using learned features. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 4

[48] W. Yuan, D. Held, C. Mertz, and M. Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018. 4

[49] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. Pcn: Point completion network. In *2018 International Conference on 3D Vision (3DV)*, pages 728–737, 2018. 3

[50] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017. 2

[51] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *CVPR*, 2017. 1, 4, 7

[52] L. Zhang and Z. Zhu. Unsupervised feature learning for point cloud understanding by contrasting and clustering using graph convolutional neural networks. In *2019 International Conference on 3D Vision (3DV)*, pages 395–404, 2019. 2