2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 978-1-6654-6946-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CVPR52688.2022.00655

Multi-instance Point Cloud Registration by Efficient Correspondence Clustering

Weixuan Tang and Danping Zou*
Shanghai Key Laboratory of Navigation and Location Based Services
Shanghai Jiao Tong University, Shanghai,China
{weixuantang,dpzou}@sjtu.edu.cn

Abstract

We address the problem of estimating the poses of multiple instances of the source point cloud within a target point cloud. Existing solutions require sampling a lot of hypotheses to detect possible instances and reject the outliers, whose robustness and efficiency degrade notably when the number of instances and outliers increase. We propose to directly group the set of noisy correspondences into different clusters based on a distance invariance matrix. The instances and outliers are automatically identified through clustering. Our method is robust and fast. We evaluated our method on both synthetic and real-world datasets. The results show that our approach can correctly register up to 20 *instances with an F1 score of* 90.46% *in the presence of* 70% *outliers, which performs significantly better and at least* 10× *faster than existing methods. (Source code : https://github.com/SJTU-ViSYS/multi-instant-reg)*

1. Introduction

Three-dimensional point cloud registration [12] [48] [45] mainly focuses on estimating one single transformation between the source point cloud and the target point cloud. However, we may sometimes want to estimate multiple transformations between point clouds. For instance, we have a 3D scan of an object and may want to find the poses of the same objects on the table within the target point cloud as shown in Figure 1. This problem, named multi-instance point cloud registration here, has been less investigated in the literature. It is non-trivial to extend existing point cloud registration methods to solve this problem.

The major challenge is to identify different clusters of corresponding points belonging to different instances within the set of noisy correspondences. One solution is to adopt a 3D object detector or apply instance segmentation to the target point cloud. After that, the pose of each instance can be estimated by a conventional point cloud registration method. However, this approach needs to train a detector or a segmentation network [39] [25] for specific objects or classes, which does not apply to unknown objects or arbitrary 3D scans. Another solution is via multi-model fitting [32] [33] [34] or [35] [29] [11]. Existing multi-model fitting methods rely on sampling valid hypotheses, which involves a large number of sampling steps when the number of models or the outlier ratio becomes high, making the efficiency and robustness of those algorithms drop drastically.

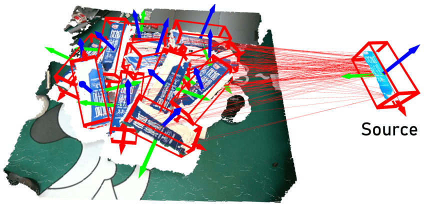


Figure 1. Multi-instance point cloud registration: Given a source point cloud of an object, multi-instance registration needs to estimate the pose of each object within the target point cloud.

In this paper, we propose a robust and efficient solution to the multi-instance 3D registration problem. The key idea is to directly group the corresponding points into different clusters according to a distance invariance matrix. Specifically, the matrix is constructed by checking the distance consistency between each pair of correspondences after the point correspondences have been obtained by feature matching using descriptors like D3Feat [7], PREDATOR [26], or SIFT [31]. We find that the row or column vector of this matrix has a powerful representation capability that can be used for identifying the set of correspondences from a particular instance. We hence apply a simple and efficient clustering algorithm to divide those correspondences into cliques. The clustering is further refined by a few recursive steps involving merging similar clusters and re-assigning cluster ids to each correspondence. Finally, both the outliers and the inliers of each instance are automatically identified by a simple ranking strategy.

*Corresponding author. This work was supported by National Science Foundation of China (62073214).

978-1-6654-6946-3/22/$31.00 ©2022 IEEE
DOI 10.1109/CVPR52688.2022.00655

Our method is highly efficient since no time-consuming hypothesis sampling is required. We have conducted extensive experiments on both synthetic and real-world datasets. The results show that our method is at least ten times faster than existing methods while performing significantly better in terms of accuracy and robustness. To summary, our contributions include:

- We propose an efficient and robust solution to the multi-instance point cloud registration problem, which achieves superior performance in terms of accuracy, robustness, and speed.
- We propose to use three metrics (Mean Hit Recall, Mean Hit Precision, and Mean Hit F1) to fully evaluate the performance of multi-instance point cloud registration.
- Our solution can be potentially used for zero-shot detection of 3D objects as our real-world tests demonstrate.

2. Related Work

Point cloud registration can be divided into three stages: point matching, outlier rejection, and pose estimation. Most works focus on the first two stages since acquiring correct point correspondences is the key to successful registration. Point matching usually relies on features, either hand-crafted features [41] [22] or learning-based features [51] [21] [24] [18] [7]. Though recent results show that the latter is superior to the hand-crafted ones in some benchmarks, those features are still far from producing perfect matching and a powerful outlier rejection mechanism is still required. RANSAC [23] and its variants([8] [13] [19]) follow the hypothesis-and-verification process to reject outliers. This kind of method requires a lot of sampling steps when many outliers exist, becoming highly time-consuming, while could still fail to obtain the correct model. GORE [15] and PMC [37] seek to reduce the outliers by geometric consistency checks. Other methods such as FGR [52] and TEASER [48] adopt robust estimators to solve the transformation directly from the noisy correspondences. By carefully tackling each subproblem, TEASER [48] achieved impressive performance in terms of robustness and efficiency. There are also learning-based outlier rejection methods. DGR [17] and 3DRegNet [36] treat outlier rejection as binary classification and predict the inlier probability for each correspondence. PointDSC [6] takes a step further to embed the spatial consistency into feature learning for better training the inlier classifier. Recently, a stream of work (e.g.PointNetLK [1], FMR [27], DCP [45], PRNet [46], RPMNet [50]) tries to apply end-to-end learning to solve the registration problem. They also exhibit impressive performance, especially in low-overlap cases [26].

Existing point cloud registration methods mostly focus on the one-to-one registration problem which estimates a single transformation between two point clouds. The multi-instance registration that aligns a source point cloud to its multiple instances in the target point cloud is however less investigated. This task is different from the multi-way registration [16] whose goal is to produce a globally consistent reconstruction from multiple fragments via pair-wise registration [52] [17]. The multi-instance registration requires not only rejecting outliers from the noisy correspondences but also identifying the set of inliers for individual instances, making it even more challenging than the classic registration problem.

3D object detection and instance segmentation are closely related to multi-instance 3D registration. Given a single point cloud, 3D object detection [39] is to obtain the bounding box of each object of interest, while 3D instance segmentation [44] [25] produces the instance labels for each point. Though they produce results [4] [5] similar to that of multi-instance registration, they need to train the prior of specific objects or categories into the network. By contrast, multi-instance registration processes two point clouds by directly aligning the source one to multiple instances in the target one, without using any priors about the contents of input 3D scans.

Multi-model fitting Multi-instance registration can be approached by multi-model fitting, which aims to estimate the model parameters from the data points generated from multiple models. Existing multi-model fitting methods can be categorized into clustering-based methods and RANSAC-based ones. The clustering-based methods(e.g. [32] [33] [34]) initialize a huge hypothesis set by sampling points and then calculate the preference vector about those hypotheses for each point. Those data points are clustered according to their preference vectors. Finally, the model parameters are computed from different clusters. RANSAC-based methods(e.g. [28] [9] [10] [35] [29] [11]) run revised RANSAC sequentially to obtain multiple model parameters. They change the sampling weight of each point in each iteration to get different model parameters. CONSAC [29] is a learning-based method that learns to weigh each point for sampling. Both clustering-based and RANSAC-based methods rely on sampling valid hypotheses. When the number of models or the outlier ratio increases, a lot of hypotheses are required to be sampled, making those algorithms highly inefficient.

3D spatial consistency, defined between every pair of points by a rigid transformation, is an important property for outlier rejection in 3D registration. Spectral matching [30] constructs a graph using the length consistency between each pair of correspondences and extracts the maxi-

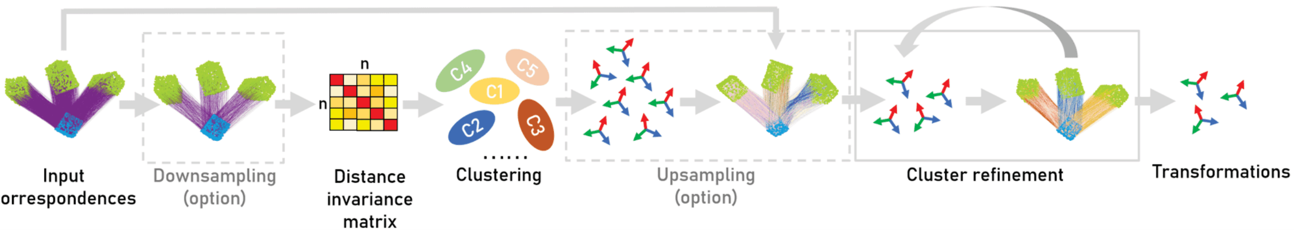


Figure 2. The pipeline of the proposed method for multi-instance point cloud registration. A distance invariance matrix is constructed from the input correspondences, which is used to cluster the correspondences into different clusters (**Clustering**) and being refined (**Cluster refinement**). Finally, the rigid transformation (**Transformations**) related to each instance is estimated from each cluster of correspondences. To handle a large number of correspondences, two addition processes (**Downsampling** and **Upsampling**) are adopted.

mum clique from the graph to reject outliers. Existing methods such as TEASER [48], GORE [15], and PMC [37] also incorporate the spatial consistency in their algorithms. Recently, ROBIN [43] generalizes the concept of spatial consistency to high orders. PointDSC [6] integrates the spatial consistency into an end-to-end learning pipeline to better regress the inlier probability.

Motivated by those works, we also adopt spatial consistency in our solution. Different from existing methods that apply spectral clustering [30] or approximated solutions [43] in the spatial consistency graph which is slow and has trouble dealing with multiple instances, we employ an efficient algorithm to find multiple instances within the correspondences. Specifically, we take the row vector or the column vector of the distance invariance matrix as the 'feature vector' of correspondence and run bottom-up clustering to get the inlier correspondences from different instances. Our method avoids hypothesis sampling which is the key weakness of existing multi-model fitting methods. It also does not rely on any particular features to obtain point correspondences, hence the performance can be further improved if better features (either 3D or image features) are adopted.

3. Problem Statement

In multi-instance point registration problem, the source point cloud $\mathbf{X}$ provides an instance of a 3D model and the target point cloud $\mathbf{Y}$ contains K instances of this model, where those instances are the sets of points that may sample only a part of the 3D model. If we write the k^{th} instance as $\mathbf{Y}_k$, the target point cloud $\mathbf{Y}$ can be decomposed as $\mathbf{Y} = \mathbf{Y}_0 \cup \mathbf{Y}_1 \cup \ldots \mathbf{Y}_k \ldots \cup \mathbf{Y}_K$. Here we use $\mathbf{Y}_0$ to represent the part of the point cloud that does not belong to any instances. The goal of multi-instance 3D registration is to find the rigid transformation $(\mathbf{R}_k, \mathbf{t}_k)$ that aligns the source instance $\mathbf{X}$ to each target instance $\mathbf{Y}_k$. If we manage to obtain the correspondences between the source instance and each target instance $\mathbf{X} \leftrightarrow \mathbf{Y}_k$, the pose of the k^{th} instance in the target point cloud, $(\mathbf{R}_k, \mathbf{t}_k)$, can be solved from the set of correspondences $\mathbf{X} \leftrightarrow \mathbf{Y}_k$ by minimizing the sum of alignment errors (1) [2]:

$$\min_{\mathbf{R}_k, \mathbf{t}_k} \sum_i \| \mathbf{y}_{ki} - (\mathbf{R}_k \mathbf{x}_i + \mathbf{t}_k) \|^2 \,. \tag{1}$$

Consider we have obtained a set of correspondences $\mathcal{C}$ between the source and target point clouds. The key of multi-instance registration task is to classify those correspondences into separate sets related to different instances, namely,

$$\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cdots \cup \mathcal{C}_K. \tag{2}$$

Here $\mathcal{C}_0$ is used to represent the set of outliers. As we can see, multi-instance registration needs to not only reject outlier correspondences but also resolve the ambiguity of correspondences from different instances. This task is not easy because all instances look the same and a lot of outlier correspondences usually exist.

4. Method

The overview of the proposed method is shown in Fig. 2. Our method takes the point correspondences as the input. An invariance consistency matrix is then constructed by checking the distance consistency between correspondences. Next, those correspondences are quickly clustered into different groups by treating the column or row vectors as 'features' of those correspondences. The clustering is done efficiently via agglomerative clustering, which is further refined by alternatively merging similar transformations and re-assigning the cluster labels for several iterations. Optionally, we apply downsampling and upsampling processes to handle the case when the number of correspondences is large. The details are presented in the next sections.

4.1. Invariance matrix & compatibility vector

The distance invariance property has been already explored in 3D registration for many years [48] [43] [30], which describes that the distance between two points keeps unchanged after a rigid transformation. Namely, if $c_i : \mathbf{x}_i \leftrightarrow \mathbf{y}_i$ and $c_j : \mathbf{x}_j \leftrightarrow \mathbf{y}_j$ are two real correspondences, they should have

$$G_{ij} = |d_{ij} - d'_{ij}| < \delta \tag{3}$$

where $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$, $d'_{ij} = \|\mathbf{y}_i - \mathbf{y}_j\|$ and δ is a threshold accounting for the noise. Hence the difference between d_{ij} and d'_{ij} can be used as a metric to test whether an outlier exists, or whether the two correspondences are from different rigid transformations. Instead of using the absolute difference defined in (3), we follow [14] to assign a score measuring the relative difference between c_i and c_j by defining

$$G_{ij} = s_{ij}^2, s_{ij} = \min(\frac{d_{ij}}{d'_{ij}}, \frac{d'_{ij}}{d_{ij}}) \in (0,1). \tag{4}$$

A *distance invariance matrix* G (where we let $G_{ii} = 1$) can be obtained by computing the scores between all the correspondence pairs. The distance invariance matrix is symmetric, where each column or row is a vector describing the compatibility between a given correspondence and other correspondences [49].

We name a column vector $G_i = (G_{i1}, \ldots, G_{ij}, \ldots)^T$ as a *compatibility vector* of the correspondence c_i. We observe that if two correspondences belong to the same instance, their *compatibility vectors* have similar patterns. Consider two correspondences $c_i, c_j \in \mathcal{C}_s$. For any correspondence $c_k \in \mathcal{C}_s$, we have $G_{ik} \to 1, G_{jk} \to 1$ because of distance invariance. For other correspondences $c_k \in \mathcal{C}/\mathcal{C}_s$, we are likely to have $G_{ik} \to 0, G_{jk} \to 0$. In other words, G_i, G_j have similar $0-1$ patterns. By contrast, if the two correspondences belong to different instances, their compatibility vectors are very different. To better understand this observation, we illustrate a simple example in Figure 3.

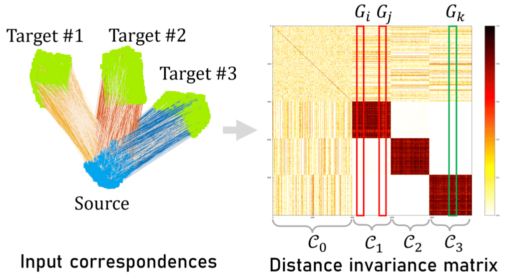


Figure 3. The column vectors (*compatibility vectors*) in the distance invariance matrix contain rich information related to the instances. Here G_i, G_j represent the compatibility vectors of i^{th} and j^{th} correspondences, which are both in the instance $\mathcal{C}_1$. We observe that G_i is similar to G_j. By constrast, G_i differs significantly from G_k since the k^{th} correspondence is within the different instance $\mathcal{C}_3$. Here $\mathcal{C}_0$ represents the set of outliers. Please refer to Section 4.1 for details.

The compatibility vector of a correspondence can be regarded as a characteristic representation or 'feature' of this correspondence. Correspondences belonging to the same rigid transformation have similar features. Therefore, based on these compatibility vectors, we can cluster the correspondences into different groups related to inliers from different instances.

4.2. Fast correspondence clustering

We cluster the correspondences in a bottom-up manner which is much faster than spectral clustering adopted by existing methods [37] [43]. In the beginning, each correspondence is treated as an individual group. We then repeatedly merge the two groups with the smallest distance until the smallest distance between two groups is larger than a given value (min_dist_thresh). The way the distance between groups being defined yields different flavors of algorithms. We follow [32] to define the distance. Let $\mathbf{p}_i, \mathbf{p}_j$ be the representation vectors of two groups i and j, the group distance is defined as

$$d(\mathbf{p}_i, \mathbf{p}_j) = 1 - \frac{\langle \mathbf{p}_i, \mathbf{p}_j \rangle}{\| \mathbf{p}_i \|^2 + \| \mathbf{p}_j \|^2 - \langle \mathbf{p}_i, \mathbf{p}_j \rangle}. \tag{5}$$

If the two groups are merged, the representation vector of the new group is updated by $\mathbf{p}_i \leftarrow \min(\mathbf{p}_i, \mathbf{p}_j)$, where $\min(\cdot)$ denotes taking the minimum value for each dimension of the two vectors. At the beginning of clustering, the representation vector of a group (containing only one correspondence) is set as the compatibility vector of that correspondence.

4.3. Recursive cluster refinement

After agglomerative clustering, we further refine the result by repeating the following steps until no change happens.

Step 1. Estimate the rigid transformations from the clusters where the number of correspondences is larger than a threshold α.

Step 2. Merge similar transformations. This step will be explained in the next section.

Step 3. Re-assign the cluster label to each correspondence. Each correspondence is assigned to the transformation where its alignment error is the smallest. If the smallest alignment error over all the transformations is larger than $inlier\_thresh$, the correspondence is marked as an outlier.

During iteration, the correspondences become more and more gathered, so we can adjust α in Step 1 to increase the strength of outlier rejection. We use the following strategy to update the α in each iteration:

$$\alpha \leftarrow \min(\alpha_0 \times \theta^{n-1}, [N/100]), \tag{6}$$

where n denotes the n^{th} iteration, N is the number of correspondences, and $[\cdot]$ is a rounding operation. We set $\alpha_0 = 3$ and $\theta = 3$ in our experiments. The refinement process usually converges within three iterations in our experiments, hence it is also highly efficient.

4.4. Merge duplicated transformations

Sometimes similar transformations are generated from different clusters, which means they probably belong to the same instance. We need to merge them in this case. Given two estimated transformations $(\mathbf{R}_1, \mathbf{t}_1)$ and $(\mathbf{R}_2, \mathbf{t}_2)$, we compute the alignment error for each correspondence, namely, $e_{ki} = \|\mathbf{y}_i - (\mathbf{R}_k\mathbf{x}_i + \mathbf{t}_k)\|^2, (k = 1, 2)$. Next, we set $p_{ki} = 1$ if $e_{ki} < inlier\_thresh$, $p_{ki} = 0$ otherwise. Thus, we obtain two binary sets P_1, P_2 for the two transformations. The criterion for merging the two transformations is

$$IOU = |P_1 \cap P_2| / |P_1 \cup P_2| \geq 80\%. \tag{7}$$

If this criterion is satisfied, we drop one of the two transformations with more outliers ($p_{ki} = 0$). Then we re-assign the cluster label to each correspondence according to the one with the smallest alignment error among all the transformations.

4.5. Extract transformations from clusters

After clustering, we need to extract the rigid transformations from those correspondence clusters. Since we do not know about the true number of instances in the target point clouds, we need to choose those inlier clusters automatically. We first select the inlier clusters whose element number is larger than a threshold (10 in our experiments) and estimate transformations from those clusters. Next, we sort the transformations by their inlier numbers in descending order. The more inliers a transformation has, the higher chance it is associated with a true instance. Finally, we check the dropping ratio of the inlier number between the transformations and the first transformation (with the most inliers) by

$$\gamma_k = \#I_k / \#I_0, \ k = 1, 2, \ldots \tag{8}$$

where $\#I_k$ denotes the number of inliers of k^{th} transformation. We neglect all the transforms after k if $\gamma_k <= \gamma\_thresh$. γ_thresh can be changed for the trade-off between recall and precision.

4.6. Handle a large number of correspondences

When the number of input correspondences is large, both calculating the distance invariance matrix and clustering the correspondences may become expensive. We add downsampling and upsampling processes to address this issue. The downsampling process is run before constructing the distance invariance matrix, which is done by randomly sampling a fixed number of correspondences (1024 in our implementation) for further processing. The upsampling process is run after clustering on the selected correspondences, which assigns all the correspondences to existing clusters. The assignment is done by selecting the transformation with the smallest alignment error as described in Section 4.3 (Step 3).

5. Experiment

We conduct experiments on both synthetic and real-world datasets by comparing our method with three state-of-the-art multi-model fitting methods: T-linkage(2014) [32], Progressive-X(2019) [10], and CONSAC(2020) [29]. Other multi-model fitting methods: RPA [33] and RansaCov [34] are extremely slow (need months) to run our experiments, hence we do not include them. We also present the results of the state-of-the-art one-to-one registration method TEASER(2020) [48] for comparison. We carefully tune all the methods to achieve the best performance on the evaluation datasets within a reasonable time and memory consumption. For a fair comparison, all the methods take the same set of point correspondences as the input.

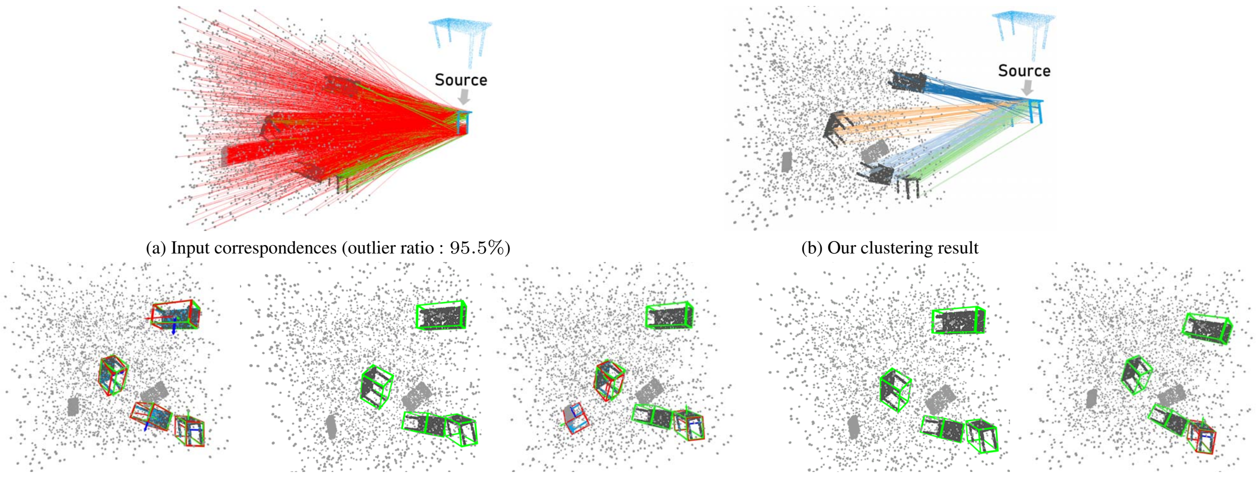
We implement our algorithm in Pytorch [38]. T-linkage and Progressive-X are pure-CPU algorithms, while CONSAC is a learned-based method that runs on GPU. We run our algorithm on the same CPU (Intel Core i7-8700K) with T-linkage and Progressive-X, and the same GPU (GTX 1080Ti) with CONSAC. Our method has three parameters, among which are set as $min\_dist\_thresh = 0.2$, $inlier\_thresh = 0.3$ and $\gamma\_thresh = 0.5$ for our experiments. All the point clouds were downsampled in $0.05m$ voxel size. Our method is not sensitive to the parameter change as the ablation study shown in the supplementary material.

As the metrics used one-to-one registration can not be used for the multi-instance setting, we adopt three metrics from the retrieval task for evaluation: MHR (Mean Hit Recall), MHP (Mean Hit Precision), MHF1 (Mean Hit F1). Their definitions are described in the supplementary material.

5.1. Synthetic datasets

We generate a synthetic dataset from a pre-sampled Modelnet40 dataset [47] from PointNet++ [40]. We downsample each point cloud to 256 points and randomly generate K (up to 20 in our tests) transformations to form a target point cloud. The target point cloud is also mixed with other objects and random points to better mimic real-world cases.

Synthetic correspondences In this test, we directly generate the input correspondences by mixing the ground truth and outlier ones. Different outlier ratios were tested, $10\% \sim 50\%$, $50\% \sim 70\%$ and $70\% \sim 90\%$. Note that the outliers were randomly sampled within a given range for each test sample. The results are shown in Table 1. As the outlier ratio increases, the performance of almost all the methods decreases, but ours drops slowly and is still significantly better than other methods. Our algorithm is 10x faster than existing methods either on CPU or on GPU. We also plot



(a) Input correspondences (outlier ratio : 95.5%)

(b) Our clustering result

(c) Ours

(d) T-Linkage(2014) [32]

(e) Progressive-X(2019) [10]

(f) CONSAC(2020) [29]

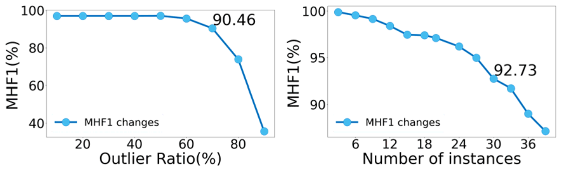
(g) TEASER(2020) [48]

Figure 4. **Results on the synthetic dataset.** (a) Input correspondences by matching PREDATOR [26] features. The inlier and outliers are visualized in green and red respectively. (b) Our clustering result is visualized by different colors (only inliers are shown). In (c-g), we visualize estimated poses in red boxes and ground truth poses in green boxes. Our method (c) registers all instances. T-linkage (d) and CONSAC (f) fail to register any instances. Progressive-X (e) registers 2 instances but produces a wrong registration. TEASER (g) registers one instance.

the MHF1(Mean Hit F1) curve of our methods with different outlier ratios in 20 instances in Figure 5(a). Though the performance degrades quickly when the outlier ratio is very large, our method still achieves 90.46% MHF1 with 70% outlier ratio. Figure 5(b) shows the MHF1 curve with different instance numbers with a fixed outlier ratio 50%. Our method's MHF1 is about 92.73% even when 30 instances are present.

Metric	MHR(%) ↑	MHP(%) ↑	MHF1(%) ↑	Time(s) ↓
	Outlier ratio : $10\% \sim 50\%$			
T-Linkage	3.05	14.80	4.65	57.27
Progressive-X	27.91	80.28	41.04	87.25
CONSAC	0.47	0.47	0.47	9.23
Ours	**96.08**	**99.73**	**97.03**	**0.62/0.30**
	Outlier ratio : $50\% \sim 70\%$			
T-Linkage	1.33	7.00	2.05	56.90
Progressive-X	20.60	75.10	31.70	85.54
CONSAC	0.49	0.49	0.49	9.55
Ours	**93.99**	**99.49**	**95.51**	**0.55/0.28**
	Outlier ratio : $70\% \sim 90\%$			
T-Linkage	0.81	4.42	1.25	56.89
Progressive-X	12.88	62.60	20.73	84.5
CONSAC	0.51	0.51	0.51	7.70
Ours	**60.39**	**94.42**	**69.36**	**0.50/0.24**
	Outlier ratio : $90\% \sim 99\%$			
T-Linkage	0.28	1.30	0.42	56.69
Progressive-X	7.13	39.19	11.67	84.43
CONSAC	0.51	0.51	0.51	9.57
Ours	**14.70**	**65.20**	**22.75**	**0.47/0.21**

Table 1. Results on synthetic correspondences with different outlier ratios. ↑ means the larger the better, while ↓ indicates the contrary. The running time on CPU/GPU of our method is presented.



(a)

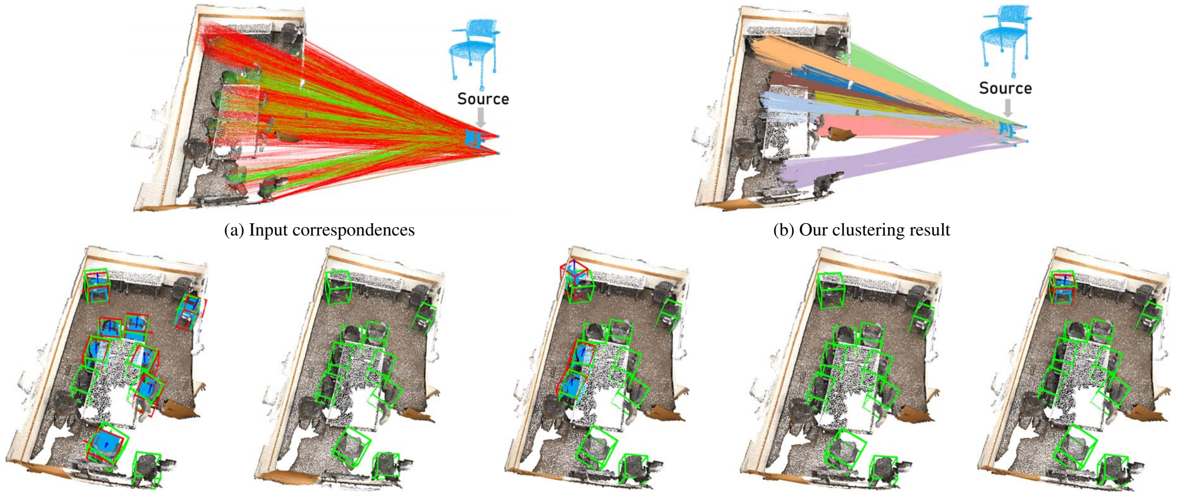
(b)

Figure 5. (a) Mean Hit F1 vs Outlier Ratio. (b) Mean Hit F1 vs Number of Instances (with a fixed outlier ratio 50%).

Correspondences by feature matching In this test, we apply feature matching to obtain the point correspondences by the PREDATOR [26] and D3Feat [7]. Both feature models were trained on synthetic data. The results are shown in Table 2. Note that both features produce correspondences with a high outlier ratio greater than 90%. In such a challenging case, our method still performs well and much better than existing methods in terms of both robustness and efficiency. The results of using D3Feat are much worse than those of using PREDATOR. The reason is that not only because of the presence of more outliers but also missing inliers as we examine the results. We visualize some results in Figure 4.

5.2. Benchmark dataset

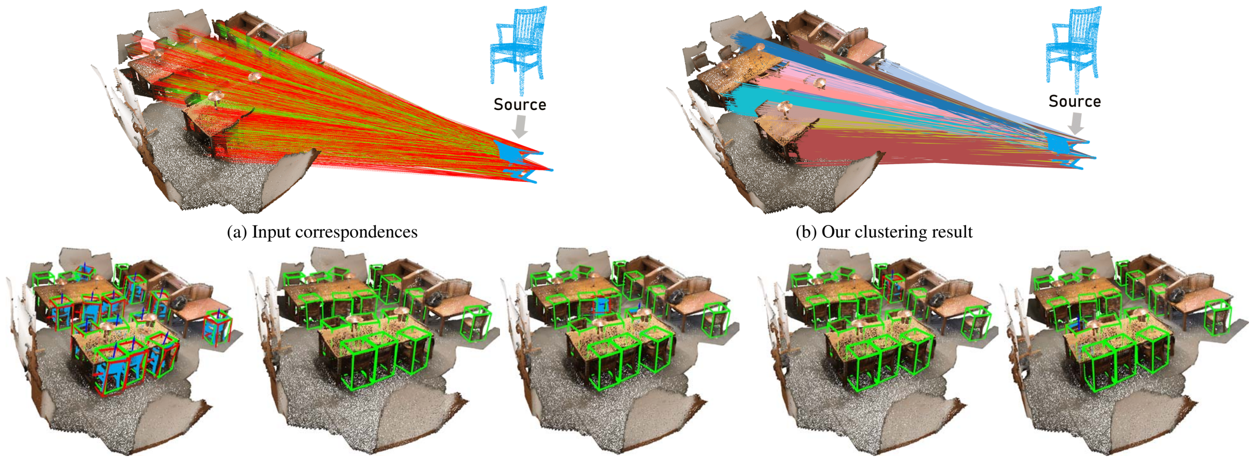
Scan2CAD [3] is a benchmark dataset that aligns the ShapeNet [42] CAD models with the object instances in ScanNet [20] point clouds. Some scans have several aligned CAD models with annotated poses. We choose those scans containing multiple CAD models as the target point cloud



(a) Input correspondences (b) Our clustering result

(c) Ours (d) T-linkage(2014) [32] (e) Progressive-X(2019) [10] (f) CONSAC(2020) [29] (g) TEASER(2020) [48]

Figure 6. **Scan2CAD results.** (a) Input correspondences by matching PREDATOR [26] features. The inlier and outliers are visualized in green and red respectively. (b) Our clustering result is visualized by different colors (only inliers are shown). In (c-g), we visualize estimated poses in red boxes and ground truth poses in green boxes. Our method (c) correctly aligns 8 instances. T-Linkage (d) and CONSAC (f) fail to register any instances. Progressive-X (e) register 3 instances. TEASER (g) registers one instance.



(a) Input correspondences (b) Our clustering result

(c) Ours (d) T-linkage(2014) [32] (e) Progressive-X(2019) [10] (f) CONSAC(2020) [29] (g) TEASER(2020) [48]

Figure 7. **Scan2CAD results.** Our method (c) registers 13 instances among 16 chairs. Progressive-X (e) registers 2 instances, but one of them has a large pose error. CONSAC (f) and TEASER (g) align one instance. T-Linkage (d) fails to register any instances.

and sample the source point cloud from the CAD model for tests. We generate 173 samples for the registration test, where most samples contain $2 \sim 5$ instances. Note that in each point cloud, only parts of instances were annotated in Scan2CAD. It means that we cannot correctly evaluate the performance such as the precision and recall using the partially annotated poses. To address this issue, we match points only within the ground-truth bounding boxes of the annotated objects in the target point clouds to generate the correspondences. Similarly, we use PREDATOR [26] and D3Feat [7] for point matching, where both are fine-tuned with 1028 training and 187 validation samples from the Scan2CAD dataset. The results are shown in Table 3. Our method performs significantly better than existing methods when using PREDATOR. Note that when using D3Feat, all the methods perform poorly. After we carefully checked the results, we found that the reason is not only the high outlier ratio (about 97.25%) but also the lack of sufficient inliers when using D3Feat, even though the feature matching is restricted within the ground truth bounding boxes in the target point clouds. Some results are visualized in Figure 6 and Figure 7. We also evaluate the performance of our method by enlarging the bounding box by$1.5\times, 2.0\times$, and $4.0\times$. When the box size is adjusted to $4\times$, the target

Metric	MHR(%) ↑	MHP(%) ↑	MHF1(%) ↑	Time(s) ↓
	PREDATOR (estimated outlier ratio : 94.32%)			
T-Linkage	0.19	0.54	0.27	43.46
Progressive-X	15.90	31.01	18.98	86.39
CONSAC	0.1	0.07	0.08	7.65
Ours	**53.39**	**61.44**	**51.80**	**1.28/0.48**
	D3Feat (estimated outlier ratio : 99.30%)			
T-Linkage	0.07	0.29	0.1	56.37
Progressive-X	4.29	15.28	5.94	87.22
CONSAC	0.13	0.04	0.05	9.53
Ours	**16.98**	**27.05**	**17.91**	**0.68/0.30**

Table 2. Results on synthetic data using feature matching to generate correspondences. Some results are visualized in Figure 4.

Metric	MHR(%) ↑	MHP(%) ↑	MHF1(%) ↑	Time(s) ↓
	PREDATOR(estimated outlier ratio : 76.44%)			
T-Linkage	2.46	3.79	2.71	1655.0
Progressive-X	11.58	6.86	7.87	26.32
CONSAC	2.66	0.35	0.62	21.35
Ours	**31.63**	**29.23**	**27.04**	**1.46/0.51**
	D3Feat (estimated outlier ratio : 97.25%)			
T-Linkage	0.04	0.22	0.06	2178.43
Progressive-X	**0.67**	**0.30**	**0.4**	28.48
CONSAC	0	0	0	21.88
Ours	0.29	0.04	0.07	**2.13/0.89**

Table 3. Results on Scan2CAD benchmark dataset.

point cloud is almost the original scan. The results based on PREDATOR features are shown in Table 4. When more background points are included, feature matching becomes more challenging, producing highly noisy correspondences, which makes the MHF1(Mean Hit F1) of our method decrease notably.

Box Size	MHR(%) ↑	MHP (%) ↑	MHF1(%) ↑	Time(s) ↓
1.5 (94.50%)	40.58	12.61	16.64	0.63
2 (95.97%)	37.43	8.85	12.35	1.37
4 (98.73%)	7.25	6.28	4.92	3.86

Table 4. Results of using different sizes of bounding boxes for feature matching on Scan2CAD dataset. The estimated outlier ratio is listed in the bracket after the box size. GPU time is listed in the last column.

5.3. Real-world tests

We use an RGB-D camera (Intel D455) to capture a sequence of point clouds a pile of objects on the table and apply our algorithm to align a 3D scan of a particular object to its multiple instances in the target RGB-D scan. Since the color information is available, we use SIFT feature to generate 3D point correspondences. Then we apply our algorithm to extract the pose of each object. Some results are shown in Figure 8. Though the table is cluttered with different objects, our methods can correctly align the source 3D scan up to more than ten instances almost in real-time (about 0.2s per frame). More results can be found in the supplementary material.

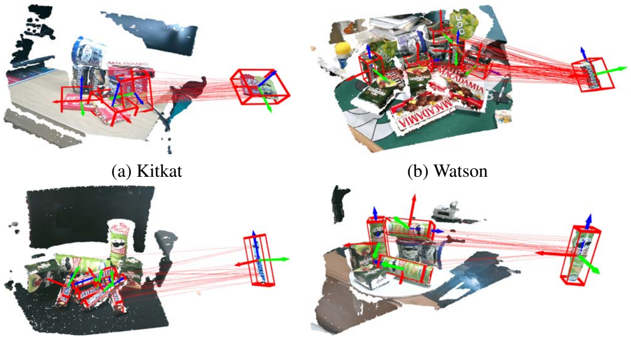

(a) Kitkat
(b) Watson
(c) Snickers
(d) Crisp

Figure 8. Real-world tests on RGB-D scans. The source point cloud is extracted from the depth scan of a single object. The target point cloud is constructed from the depth scan captured from the camera viewpoint.

6. Limitation

The performance of our method relies on the quality of point correspondences. Unfortunately, we found that using state-of-the-art 3D features such as D3Feat and PREDATOR produces unsatisfactory correspondences, although they have exhibited good performances on some benchmark tests. To improve the correspondence quality, we have to train those features for each dataset in our experiments. Even by doing so, the outlier ratios are still very high and sometimes the inliers are missing for some instances (especially when using D3Feat) which significantly reduces the recall as the experiments show. Therefore, the 3D feature is a bottleneck that requires to be significantly improved. Another limitation is that distance invariance is a weak rule that does not hold well for noisy point clouds and sometimes is insufficient to reject outliers that are close to inliers, which may also degrade the performance of our method. One possible solution is to seek a better invariance 'feature' representing each correspondence within an end-to-end learning pipeline as [6].

7. Conclusion

We address the novel task of multi-instance 3D registration in this paper. We found that the column vectors of the distance invariance matrix encode rich information about the instance to which the correspondences are related. Based on this observation, we cluster the correspondences into different groups efficiently by an agglomerative algorithm and refine the result by several iterations. The results on synthetic, benchmark, and real-world datasets show that our method outperforms existing methods significantly in terms of robustness, accuracy, and efficiency. Though our solution is still far from perfect as discussed, we hope our work could inspire future research on this topic.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *CVPR*, pages 7163–7172, 2019. 2

[2] K.S. Arun, T.S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE TPAMI*, PAMI-9(5):698–700, 1987. 3

[3] Armen Avetisyan, Manuel Dahnert, Angela Dai, Manolis Savva, Angel X Chang, and Matthias Niessner. Scan2cad: Learning cad model alignment in rgb-d scans. In *CVPR*, pages 2614–2623, 2019. 6

[4] Armen Avetisyan, Angela Dai, and Matthias Niessner. End-to-end cad model retrieval and 9dof alignment in 3d scans. In *ICCV*, pages 2551–2560, 2019. 2

[5] Armen Avetisyan, Tatiana Khanova, Christopher Choy, Denver Dash, Angela Dai, and Matthias Niessner. Scenecad: Predicting object alignments and layouts in rgb-d scans. In *ECCV*, pages 596–612. Springer, 2020. 2

[6] Xuyang Bai, Zixin Luo, Lei Zhou, Hongkai Chen, Lei Li, Zeyu Hu, Hongbo Fu, and Chiew-Lan Tai. Pointdsc: Robust point cloud registration using deep spatial consistency. *CVPR*, 2021. 2, 3, 8

[7] Xuyang Bai, Zixin Luo, Lei Zhou, Hongbo Fu, Long Quan, and Chiew-Lan Tai. D3feat: Joint learning of dense detection and description of 3d local features. *CVPR*, 2020. 1, 2, 6, 7

[8] Daniel Barath and Jiri Matas. Graph-cut ransac. In *CVPR*, 2018. 2

[9] Daniel Barath and Jiri Matas. Multi-class model fitting by energy minimization and mode-seeking. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *ECCV*, pages 229–245, Cham, 2018. Springer International Publishing. 2

[10] Daniel Barath and Jiri Matas. Progressive-x: Efficient, anytime, multi-model fitting algorithm. In *ICCV*, pages 3780–3788, 2019. 2, 5, 6, 7

[11] Daniel Barath, Denys Rozumny, Ivan Eichhardt, Levente Hajder, and Jiri Matas. Progressive-x+: Clustering in the consensus space. *arXiv preprint arXiv:2103.13875*, 2021. 1, 2

[12] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE TPAMI*, 14(2):239–256, 1992. 1

[13] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In *ICCV*, 2019. 2

[14] Anders Glent Buch, Yang Yang, Norbert Krüger, and Henrik Gordon Petersen. In search of inliers: 3d correspondence by local and global voting. In *CVPR*, pages 2075–2082, 2014. 4

[15] Alvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE TPAMI*, 40(12):2868–2882, 2017. 2, 3

[16] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, pages 5556–5565, 2015. 2

[17] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. *CVPR*, pages 2511–2520, 2020. 2

[18] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In *ICCV*, pages 8958–8966, 2019. 2

[19] Ondrej Chum, Jiri Matas, and Josef Kittler. Locally optimized ransac. In *DAGM-Symposium*, 2003. 2

[20] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. 6

[21] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. *CVPR*, pages 195–205, 2018. 2

[22] Bertram Drost, Markus Ulrich, Nassir Navab, and Slobodan Ilic. Model globally, match locally: Efficient and robust 3d object recognition. In *CVPR*, pages 998–1005. Ieee, 2010. 2

[23] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 2

[24] Zan Gojcic, Caifa Zhou, Jan D. Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. *CVPR*, 2019-June:5540–5549, 2019. 2

[25] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. Occuseg: Occupancy-aware 3d instance segmentation. In *CVPR*, pages 2940–2949, 2020. 1, 2

[26] Shengyu Huang, Zan Gojcic, Mikhail Usvyatsov, Andreas Wieser, and Konrad Schindler. Predator: Registration of 3d point clouds with low overlap. In *CVPR*, pages 4267–4276, 2021. 1, 2, 6, 7

[27] Xiaoshui Huang, Guofeng Mei, and Jian Zhang. Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In *CVPR*, June 2020. 2

[28] Hossam Isack and Yuri Boykov. Energy-based geometric multi-model fitting. *IJCV*, 97(2):123–147, 2012. 2

[29] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. Consac: Robust multi-model fitting by conditional sample consensus. In *CVPR*, 2020. 1, 2, 5, 6, 7

[30] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. 2005. 2, 3

[31] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[32] Luca Magri and Andrea Fusiello. T-linkage: A continuous relaxation of j-linkage for multi-model fitting. *CVPR*, pages 3954–3961, 2014. 1, 2, 4, 5, 6, 7

[33] Luca Magri and Andrea Fusiello. Robust multiple model fitting with preference analysis and low-rank approximation. *BMVC*, pages 20.1–20.12, 2015. 1, 2, 5

[34] Luca Magri and Andrea Fusiello. Multiple model fitting as a set coverage problem. In *CVPR*, pages 3318–3326, 2016. 1, 2, 5

[35] Luca Magri and Andrea Fusiello. Fitting multiple heterogeneous models by multi-class cascaded t-linkage. In *CVPR*, pages 7460–7468, 2019. 1, 2

[36] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *CVPR*, pages 7193–7203, 2020. 2

[37] Álvaro Parra, Tat-Jun Chin, Frank Neumann, Tobias Friedrich, and Maximilian Katzmann. A practical maximum clique algorithm for matching with pairwise constraints. *arXiv preprint arXiv:1902.01534*, 2019. 2, 3, 4

[38] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[39] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019. 1, 2

[40] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 2017-Decem:5100–5109, 2017. 5

[41] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, pages 3212–3217. IEEE, 2009. 2

[42] Manolis Savva, Fisher Yu, Hao Su, M Aono, B Chen, D Cohen-Or, W Deng, Hang Su, Song Bai, Xiang Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *3DOR*, volume 10, 2016. 6

[43] Jingnan Shi, Heng Yang, and Luca Carlone. Robin: a graph-theoretic approach to reject outliers in robust estimation using invariants. In *ICRA*, pages 13820–13827. IEEE, 2021. 3, 4

[44] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, pages 2569–2578, 2018. 2

[45] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *ICCV*, pages 3523–3532, 2019. 1, 2

[46] Yue Wang and Justin M. Solomon. Prnet: Self-supervised learning for partial-to-partial registration. In *NeurIPS*, 2019. 2

[47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 5

[48] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *TRO*, 37(2):314–333, 2020. 1, 2, 3, 5, 6, 7

[49] Jiaqi Yang, Ke Xian, Peng Wang, and Yanning Zhang. A performance evaluation of correspondence grouping methods for 3d rigid data matching. *IEEE TPAMI*, 14(8):1–1, 2019. 4

[50] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. *CVPR*, 2020. 2

[51] Andy Zeng, Shuran Song, Matthias Niessner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. *CVPR*, 2017-Janua:199–208, 2017. 2

[52] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. *ECCV*, 9906(August):694–711, 2016. 2