# Efficient and Accurate Multi-Instance Point Cloud Registration with Iterative Main Cluster Detection

Zhiyuan Yu[1] , Qin Zheng[2], Chenyang Zhu[1], Kai Xu[1]

[1]National University of Defense Technology, China; [2]Defense Innovation Institute, Academy of Military Sciences, China

**Abstract**

*Multi-instance point cloud registration is the problem of recovering the poses of all instances of a model point cloud in a scene point cloud. A traditional solution first extracts correspondences and then clusters the correspondences into different instances. We propose an efficient and robust method which clusters the correspondences in an iterative manner. In each iteration, our method first computes the spatial compatibility matrix between the correspondences, and detects its main cluster. The main cluster indicates a potential occurrence of an instance, and we estimate the pose of this instance with the correspondences in the main cluster. Afterwards, the correspondences are removed to further register new instances in the following iterations. With this simplistic design, our method can adaptively determine the number of instances, achieving significant improvements on both efficiency and accuracy.*

**CCS Concepts**

• ***Computing methodologies*** $\rightarrow$ ***Matching; Scene understanding;***

## 1. Introduction

Multi-instance point cloud registration aims at estimating the poses of all instances of a model point cloud within a scene point cloud, which plays an important role in augmented reality, robotic manipulation, and autonomous driving. Compared to pairwise registration which recovers the relative pose between two point clouds from different viewpoints, multi-instance registration is more challenging due to the unknown number of instances in the scene.

A traditional solution is to first generate putative correspondences by feature matching [CPK19, QYW*22] and then recover the poses of multiple instances by multi-model fitting algorithms. Multi-model fitting methods usually follow a hypothesize-and-verify paradigm [MF14], which first samples a large number of hypotheses and then selects the valid hypotheses among them. However, this class of methods suffers from low accuracy under high outlier ratio and low efficiency due to numerous hypotheses.

Recent clustering-based methods directly divide the putative correspondences into different clusters, where each cluster is recovered as a model instance. ECC [TZ22] employs a hierarchical clustering algorithm to group the correspondences into multiple subsets, and each subset is used to recover the pose of a model instance. Nevertheless, the complexity of hierarchical clustering reaches $O(N^3)$ in the worst case, where $N$ is the number of correspondences, leading to low efficiency if there are many putative correspondences. PointCLM [YLJ*22] adopts a neural network to embed the correspondences into the feature space and then applies spectral clustering based on feature similarity and spatial consistency [LH05, BLZ*21] to recognize new instances in an iterative fashion. It determines the number of clusters by the position of the maximum drop in the eigenvalues, and RANSAC is used to estimate the pose of each cluster. However, the time complexity of spectral clustering is also $O(N^3)$, which prevents PointCLM from handling a large number of correspondences. To maintain applicable registration speed, these methods usually downsample the input correspondences to a relatively small subset. Nevertheless, in multi-instance cases, the distribution of inliers in the scene point cloud is non-uniform and downsampling may result in losing some instances. As shown in Fig. 1 (right), randomly downsampling the correspondences from 6000 to 1000 significantly reduces the registered instances.

In this work, we propose an efficient and accurate multi-instance registration method which clusters the putative correspondences in an iterative manner. Our key idea is to register one instance at a time until there are no more instances in the scene. Inspired by [LH05], the main cluster of the spatial compatibility matrix between the correspondences indicates the potential occurrence of an instance. To this end, we iteratively detect a main cluster and cluster its correspondences as an instance. In each iteration, we use the second-order spatial consistency [CSYT22] to construct a correspondence compatibility matrix. We then compute the association of each correspondence with the main cluster of the matrix and generate a set of seed correspondences according the association scores. The seeds further generate a set of pose hypotheses, where we select the best pose as a new instance. At last, the correspondences be-

DOI: 10.2312/egs.20241033

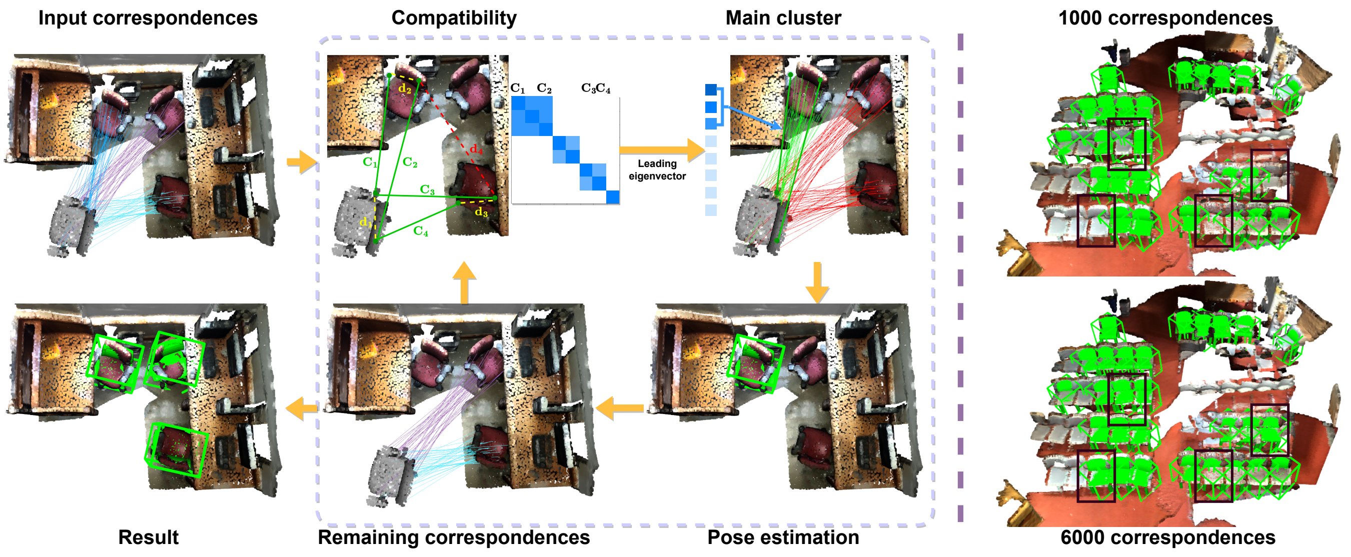

**Figure 1:** *Left: The pipeline of our method. It takes point correspondences as input and constructs their similarity matrix, which is used to extract the main cluster. Then, the pose is estimated on the main cluster and the next iteration takes the remaining correspondences as input after deleting the matched correspondences with the pose. Right: Visualization of registration results under different numbers of correspondences. The black rectangle indicates the difference in registration results between 1000 and 6000 correspondences.*

longing to this instance are removed before the next iteration. The advantages of our method are two-fold. First, instead of determining the number of clusters before clustering, our method adaptively determines the number of instances during registration, enhancing the robustness to the unknown number of instances. Second, our method bypasses the computationally-expensive clustering algorithms with a simplistic iterative pipeline, thus is more efficient and can leverage more correspondences to generate more reliable registrations. Experiments on Scan2CAD benchmark [ADD*19] have demonstrated the effectiveness of our method. Our method surpasses the state-of-the-art methods [TZ22, YLJ*22] by 21.23 points on the F1 score and runs 23× faster with 6000 correspondences.

## 2. Method

In this work, we consider a model point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$ and a scene point cloud $\mathbf{Q} \in \mathbb{R}^{M\times 3}$, where the scene point cloud contains multiple instances of the model point cloud. Given a set of putative correspondences $\mathcal{C} = \{(\mathbf{x}_i, \mathbf{y}_i) | \mathbf{x}_i \in \mathbf{P}, \mathbf{y}_i \in \mathbf{Q}\}$ extracted by a point matching method [CPK19, QYW*22], our goal is to recover the poses of the model instances in the scene. Assuming that there are $K$ instances, the correspondences can be decomposed as $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_2 \cup \cdots \cup \mathcal{C}_K$, where $\mathcal{C}_0$ is the outlier correspondences and $\mathcal{C}_k$ is the inlier correspondences belonging to the $k^{\text{th}}$ instance. The pose of the $k^{\text{th}}$ instance $\mathbf{T}_k = \{\mathbf{R}_k, \mathbf{t}_k\}$ can be solved by:

$$\underset{R_k, t_k}{\operatorname{argmin}} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}_k} ||R_k \cdot \mathbf{x}_i + t_k - \mathbf{y}_i||_2, \tag{1}$$

where $\mathbf{R}_k$ is a 3D rotation and $\mathbf{t}_k$ is a 3D translation.

The key to this problem is to cluster the putative correspondences into separate instances and remove the outliers. This is very challenging due to the unknown instance number and the ambiguity of correspondences from different instances. To address this problem, we leverage spatial consistency between the correspondences to compute a compatibility matrix and recover the instances by iteratively extracting the main cluster of the compatibility matrix. In each iteration, we treat the main cluster as an instance and recover its pose with the correspondences in the main cluster. Afterward, all correspondences that support the recovered pose are removed. The pipeline of our method is shown in Algorithm 1.

**Algorithm 1** Cluster the correspondence set

```
 1: procedure CLUSTER(Correspondence set $C$)
 2:     $Final\_poses \leftarrow [\,]$
 3:     $Max\_inliers \leftarrow 0$
 4:     while LEN($C$) $> 3$ do
 5:         $S \leftarrow$ CAL_COMPATIBILITY($C$)
 6:         $Confidence \leftarrow$ CAL_LEADING_EIGENVECTOR($S$)
 7:         $Seeds \leftarrow$ SEED_SELECTION($Confidence$)
 8:         $\{Pose, Inliers\} \leftarrow$ CAL_BEST_POSE($Seeds, C, S$)
 9:         $Final\_poses$.APPEND($Pose$)
10:         $C \leftarrow$ REMOVEINLIERS($C, Inliers$)
11:         if LEN($Inliers$) $> Max\_inliers$ then
12:             $Max\_inliers \leftarrow$ LEN($Inliers$)
13:         end if
14:         if LEN($Inliers$) $< Max\_inliers \times \tau$ then
15:             break
16:         end if
17:     end while
18:     return $Final\_poses$
19: end procedure
```

### 2.1. Spatial Compatibility Matrix

Spatial consistency [BLZ*21, CSYT22] is an important criterion to retrieve inlier correspondences in point cloud registration, i.e., the respective lengths between two inliers in the two point clouds should be close. In multi-instance registration, spatial consistency holds only for inliers from the same instance, which provides strong geometric clues for correspondence clustering. Based on this motivation, we construct the compatibility matrix between correspondences using the second-order spatial consistency measure [CSYT22] in each iteration. Specifically, given two correspondences $\mathbf{c}_i = (\mathbf{x}_i, \mathbf{y}_i)$ and $\mathbf{c}_j = (\mathbf{x}_j, \mathbf{y}_j)$, the spatial consistency matrix $\mathbf{H}$ is computed as:

$$h_{ij} = \begin{cases} 1, & d_{ij} \leq d_{\text{thr}}, \\ 0, & d_{ij} > d_{\text{thr}}. \end{cases}, \quad d_{ij} = \big| \|\mathbf{x}_i - \mathbf{x}_j\| - \|\mathbf{y}_i - \mathbf{y}_j\| \big|, \tag{2}$$

where $d_{\text{thr}}$ is the distance threshold. Two correspondences are considered as incompatible if their length difference exceeds $d_{\text{thr}}$. Then, we build the final compatibility matrix $\mathbf{S}$ as follows:

$$s_{ij} = h_{ij} \cdot \sum_{k=1}^{N} (h_{ik} \cdot h_{kj}). \tag{3}$$

Here $s_{ij}$ measures the number of correspondences which are compatible with both $\mathbf{c}_i$ and $\mathbf{c}_j$. The inlier correspondences belonging to the same instance tend to have high scores in the $\mathbf{S}$, and the scores between the inliers from different instances or the outliers are expected to be small. Therefore, we can utilize the compatibility matrix to divide the correspondences into different clusters.

## 2.2. Iterative Instance Registration

However, it is still hard to determine the number of clusters due to the unknown instance number and the existence of outliers, which prevents the usage of classic clustering algorithms. Our key insight is that we do not need the number before clustering the correspondences. Instead, we adaptively determine the number of instances by iteratively extracting main clusters of the compatibility matrix. As noted in [LH05, BLZ*21], a main cluster indicates a set of compatible inlier correspondences of a potential instance.

In each iteration, we first compute the compatibility matrix $\mathbf{S}$ between the correspondences as described above. We then compute the leading eigenvector of $\mathbf{S}$ as the association of each correspondence with a main cluster or instance [LH05]. The leading eigenvector also indicates the confidence of each correspondence being an inlier of this instance. We follow [BLZ*21] to use the power iteration algorithm to compute the leading eigenvector.

Next, we devise a seeding method to solve for the pose of this instance. Specifically, we select the top-$k$ correspondences with the highest confidence scores as the initial seeds. Nevertheless, the initial seeds could cluster together, leading to redundant computation. To this end, we further apply Non-Maximum Suppression (NMS) over the confidence score to generate well-distributed seeds. This process removes the seeds which are within the distance of $\sigma_r$ to other seeds with higher confidence scores. For each seed $\mathbf{c}_i$, we find its $k$ most compatible correspondences $\mathcal{C}'_i$ and construct the compatibility matrix as in Eq. 3. We then compute its leading eigenvector $\mathbf{w}_i$ as the weights for the correspondences in $\mathcal{C}'$ and estimate a pose $\mathbf{T}_i = \{\mathbf{R}_i, \mathbf{t}_i\}$ by solving:

$$\mathbf{R}_i, \mathbf{t}_i = \underset{\mathbf{R},\mathbf{t}}{\operatorname{argmin}} \sum\nolimits_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{C}'_i} w_{i,j} \|\mathbf{R} \cdot \mathbf{x}_j + \mathbf{t} - \mathbf{y}_j\|^2. \tag{4}$$

| Model | Samples | Time($s$) | MR(%) | MP(%) | MF(%) |
|---|---|---|---|---|---|
| PointCLM [YLJ*22] | 1000 | **0.04** | 65.59 | **71.24** | 68.30 |
| ECC [TZ22] | 1000 | 0.68 | 83.70 | 55.63 | 66.83 |
| Ours | 1000 | 0.23 | **87.24** | 70.60 | **78.04** |
| PointCLM [YLJ*22] | 2000 | **0.10** | 78.22 | 68.28 | 72.91 |
| ECC [TZ22] | 2000 | 1.36 | 84.78 | 62.64 | 72.05 |
| Ours | 2000 | 0.22 | **89.88** | **73.31** | **80.75** |
| PointCLM [YLJ*22] | 3000 | 0.30 | 82.44 | 52.68 | 64.29 |
| ECC [TZ22] | 3000 | 2.10 | 87.45 | 62.16 | 72.66 |
| Ours | 3000 | **0.23** | **91.37** | **73.95** | **81.74** |
| PointCLM [YLJ*22] | 4000 | 0.91 | 81.67 | 38.70 | 52.52 |
| ECC [TZ22] | 4000 | 3.25 | 88.96 | 55.54 | 68.38 |
| Ours | 4000 | **0.24** | **91.61** | **75.15** | **82.57** |
| PointCLM [YLJ*22] | 5000 | 2.73 | 79.46 | 28.93 | 42.42 |
| ECC [TZ22] | 5000 | 5.06 | 89.84 | 49.84 | 64.11 |
| Ours | 5000 | **0.28** | **92.17** | **74.48** | **82.39** |
| PointCLM [YLJ*22] | 6000 | 9.58 | 77.67 | 24.40 | 37.14 |
| ECC [TZ22] | 6000 | 7.59 | 92.23 | 46.13 | 61.50 |
| Ours | 6000 | **0.32** | **92.55** | **74.79** | **82.73** |

**Table 1:** *Evaluation results on Scan2CAD benchmark.*

This problem can be solved in closed form by weighted SVD. Finally, we select the pose which admits the most inlier correspondences over the entire correspondences $\mathcal{C}$ among all seeds as the pose for this instance.

At last, we remove the inliers of this instance from the correspondence set $C$ and use the remaining correspondences for the next iteration. The process above repeats until there are fewer than three correspondences or the number of inliers in the last iteration falls below $Max\_inliers \cdot \tau$. Here $Max\_inliers$ is the maximum number of inliers over all iterations, and $\tau$ is the ratio threshold. Since the main clusters from the outliers tend to be smaller than those from the inliers, our method could quickly stop if all model instances have been registered.

## 3. Experiments

We implement our algorithm using Pytorch and conduct experiments on the real-world dataset Scan2CAD [ADD*19], comparing our method with two state-of-the-art methods: ECC [TZ22] and PointCLM [YLJ*22]. We employ a fine-tuned FCGF [CPK19] to extract local features of point clouds and generate 20000 correspondences by feature matching. We use a 0.025$m$ voxel grid to downsample the point cloud and $d_{\text{thr}}$ is set to 0.05, $\tau$ is set to 0.35, $\sigma_r$ is set to 0.1. In each iteration, we select ten seeds and sample 40 nearest neighbors as the correspondence subset. Following [YLJ*22, TZ22], we evaluate the performance with mean recall (MR), mean precision (MP), and mean F1 Score (MF). For asymmetric objects, instance is considered successfully registered when *Relative Rotation Error* (RRE) and *Relative Translation Error* (RTE) are smaller than thresholds ($\text{RRE} < 15^\circ, \text{RTE} < 0.1\text{m}$). For symmetric objects, we use the ADD-S metric to evaluate the estimated poses ($\text{ADD-S} < 0.1$).

As shown in Tab. 1, our method achieves the best performance in MR, MP, and MF. Compared to ECC and PointCLM, our method is not sensitive to the size of input and our method runs 30× faster than PointCLM, and green23× faster than ECC under 6000 correspondences input. As the number of inputs increases, all methods detect more instances but the rising inputs also bring more outliers. Nevertheless, our method's mean precision remains stable under more noise and our method rejects more outliers than ECC and

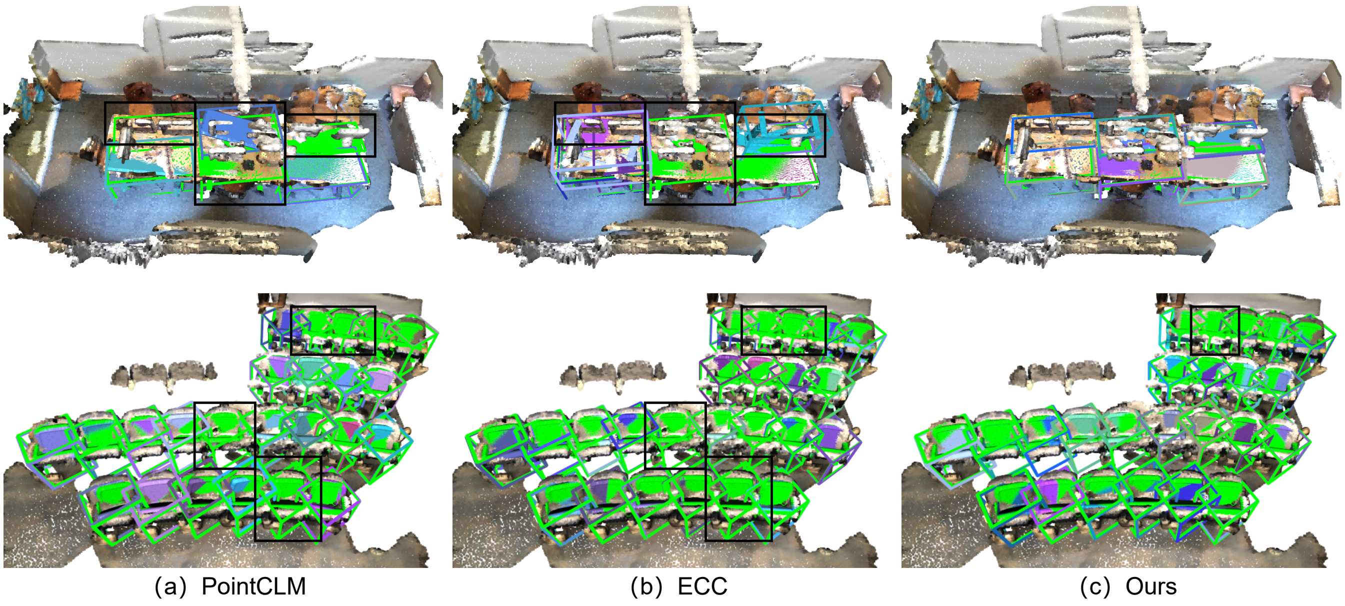

**Figure 2:** *Qualitative registration results on the Scan2CAD benchmark. In (a-c), the green bounding boxes represent the ground truth poses in the scene point cloud, and the bounding boxes in other colors represent the predicted pose. The black rectangles indicate the unregistered instances.*

| Samples | τ | Time(*s*) | MR(%) | MP(%) | MF(%) |
|---|---|---|---|---|---|
| 1000 | 0.15 | 0.41 | 92.85 | 42.07 | 57.91 |
| 3000 | | 0.42 | 95.25 | 45.35 | 61.44 |
| 5000 | | 0.51 | 96.35 | 45.84 | 62.12 |
| 1000 | 0.25 | 0.31 | 89.57 | 61.82 | 73.15 |
| 3000 | | 0.30 | 93.40 | 63.90 | 75.88 |
| 5000 | | 0.35 | 94.44 | 64.65 | 76.75 |
| 1000 | 0.45 | 0.10 | 85.33 | 76.43 | 80.63 |
| 3000 | | 0.11 | 88.64 | 79.63 | 83.89 |
| 5000 | | 0.14 | 88.58 | 80.15 | 84.15 |
| 1000 | **0.35** | 0.23 | 87.24 | 70.60 | 78.04 |
| 3000 | | 0.23 | 91.37 | 73.95 | 81.74 |
| 5000 | | 0.28 | 92.17 | 74.48 | 82.39 |

**Table 2:** *Ablation experiments on the hyperparameters* τ.

PointCLM. Fig. 2 provides a gallery of the registration results of our method, ECC, and PointCLM. On the top row, our method successfully registers all instances without any incorrect predictions. ECC and PointCLM successfully register two instances with some wrong poses. On the bottom row, the number of unregistered instances using our method is fewer than those with ECC and PointCLM.

In Tab. 2, we evaluate the sensitivity to the hyperparameter τ which controls when to stop the algorithm. As shown in the table, when τ is small, the method tends to detect more instances but incorporates more wrong results, and costs more time.

## 4. Conclusion & Limitation

This paper presents an efficient and accurate method for multi-instance registration, employing an iterative clustering approach for putative correspondences. Through estimating one pose based on the main cluster each time, we improve the robustness of unknown cluster numbers. The decreasing input for each iteration improves our efficiency and allows us to handle more correspondences. The experiments conducted on Scan2CAD demonstrate our accuracy and efficiency over existing methods. However, our method, along with others that utilize spatial consistency, encounters a limitation that the outlier set is not merely random noise, it contains some subsets that satisfy spatial consistency. The unknown number of clusters makes it difficult for us to distinguish whether these subsets that satisfy spatial consistency are inlier sets or outlier sets. This problem is caused by local features geometrically less discriminative, resulting in outlier matches. We aim to address this problem in future work.

## References

[ADD*19] Avetisyan A., Dahnert M., Dai A., Savva M., Chang A. X., Niessner M.: Scan2cad: Learning cad model alignment in rgb-d scans. In *CVPR* (2019), pp. 2614–2623. 2, 3

[BLZ*21] Bai X., Luo Z., Zhou L., Chen H., Li L., Hu Z., Fu H., Tai C.-L.: Pointdsc: Robust point cloud registration using deep spatial consistency. In *CVPR* (2021), pp. 15859–15869. 1, 2, 3

[CPK19] Choy C., Park J., Koltun V.: Fully convolutional geometric features. In *CVPR* (2019), pp. 8958–8966. 1, 2, 3

[CSYT22] Chen Z., Sun K., Yang F., Tao W.: Sc2-pcr: A second order spatial compatibility for efficient and robust point cloud registration. In *CVPR* (2022), pp. 13221–13231. 1, 2, 3

[LH05] Leordeanu M., Hebert M.: A spectral technique for correspondence problems using pairwise constraints. In *ICCV* (2005), vol. 2, IEEE, pp. 1482–1489. 1, 3

[MF14] Magri L., Fusiello A.: T-linkage: A continuous relaxation of j-linkage for multi-model fitting. In *CVPR* (2014), pp. 3954–3961. 1

[QYW*22] Qin Z., Yu H., Wang C., Guo Y., Peng Y., Xu K.: Geometric transformer for fast and robust point cloud registration. In *CVPR* (2022), pp. 11143–11152. 1, 2

[TZ22] Tang W., Zou D.: Multi-instance point cloud registration by efficient correspondence clustering. In *CVPR* (2022), pp. 6667–6676. 1, 2, 3

[YLJ*22] Yuan M., Li Z., Jin Q., Chen X., Wang M.: Pointclm: A contrastive learning-based framework for multi-instance point cloud registration. In *ECCV* (2022), Springer, pp. 595–611. 1, 2, 3