End-to-end Learning the Partial Permutation Matrix for Robust 3D Point Cloud Registration

Zhiyuan Zhang[1], Jiadai Sun[1], Yuchao Dai[1*], Dingfu Zhou[2], Xibin Song[2], Mingyi He[1]

[1] Northwestern Polytechnical University [2] Baidu Research.
{zhangzhiyuan,sunjiadai}@mail.nwpu.edu.cn, daiyuchao@gmail.com,
{zhoudingfu,songxibin}@baidu.com, myhe@nwpu.edu.cn

Abstract

Even though considerable progress has been made in deep learning-based 3D point cloud processing, how to obtain accurate correspondences for robust registration remains a major challenge because existing hard assignment methods cannot deal with outliers naturally. Alternatively, the soft matching-based methods have been proposed to learn the matching probability rather than hard assignment. However, in this paper, we prove that these methods have an inherent ambiguity causing many deceptive correspondences. To address the above challenges, we propose to learn a partial permutation matching matrix, which does not assign corresponding points to outliers, and implements hard assignment to prevent ambiguity. However, this proposal poses two new problems, *i.e*. existing hard assignment algorithms can only solve a full rank permutation matrix rather than a partial permutation matrix, and this desired matrix is defined in the discrete space, which is non-differentiable. In response, we design a dedicated soft-to-hard (S2H) matching procedure within the registration pipeline consisting of two steps: solving the soft matching matrix (S-step) and projecting this soft matrix to the partial permutation matrix (H-step). Specifically, we augment the profit matrix before the hard assignment to solve an augmented permutation matrix, which is cropped to achieve the final partial permutation matrix. Moreover, to guarantee end-to-end learning, we supervise the learned partial permutation matrix but propagate the gradient to the soft matrix instead. Our S2H matching procedure can be easily integrated with existing registration frameworks, which has been verified in representative frameworks including DCP, RPMNet, and DGR. Extensive experiments have validated our method, which creates a new state-of-the-art performance for robust 3D point cloud registration. *The code will be made public.*

Introduction

3D point cloud registration is a well-known task in 3D vision with wide applications including object pose estimation (Wong et al. 2017), 3D reconstruction (Deschaud 2018), simultaneous localization and mapping (Shiratori et al. 2015; Ding and Feng 2019), *etc*. Although the increasingly prosperous deep learning technique has achieved great success in point cloud registration (Lu et al. 2019), how to obtain accurate correspondences for robust registration remains a stubborn problem, which can be formulated as solving a matching matrix to relate the input two point clouds. And each entry indicates the point pair is a correspondence or not.

For ideal consistent point clouds, where the inputs are exactly the same except for the pose, *i.e*. each point can find a corresponding point in the other point cloud, the correspondences are built by searching a permutation matching matrix, which implements the one-to-one matching principle. However, in practical applications, input point clouds are usually not consistent due to the outliers (*i.e*. the points without corresponding points). To handle the outliers, a widely adopted strategy is to select reliable correspondences after the initial matching (Choy, Dong, and Koltun 2020; Probst et al. 2019; Pais et al. 2020; Deng, Birdal, and Ilic 2018b,a; Bai et al. 2021). Nonetheless, this remedial operation is complicated. Alternatively, we propose to handle the outliers in the matching stage synchronously. In this case, the desired matching matrix is turned to a partial permutation matrix (notated as PPM) formulated by a binary matrix, where the sum of row or column corresponding to inlier/outlier is one/zero. PPM embeds two important principles: one-to-one matching and outliers pruning. Unfortunately, existing hard assignment algorithms are not competent to directly solve this PPM since they cannot distinguish inliers and outliers, and they will solve a full rank permutation matrix assigning corresponding points to outliers incorrectly.

Moreover, PPM is defined in the discrete space, and the hard assignment algorithm is non-differentiable, which is fatal for the deep learning pipeline. To address this issue, soft matching-based methods are proposed. They relax the discrete matching matrix into a continuous one, where each entry is either zero or one, and then the virtual points are achieved by performing weighted average to replace the real corresponding points. However, since the geometric constraint is ignored, the network does not learn the underlying physics, which results in an inherent ambiguity making the distribution of the virtual points degenerate seriously. DCP (Wang and Solomon 2019a) is a typical soft matching method, which suffers from this drawback as shown in Table 1. RPMNet (Yew and Lee 2020) applies the Augmented-Sinkhorn algorithm replying to outliers in the soft matching process, where the "trash bin" strategy (Sarlin et al. 2020) is employed. However, the degeneration has not been re-

*Yuchao Dai is the corresponding author.

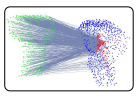
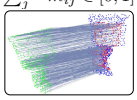
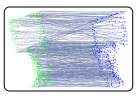
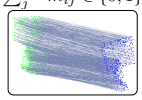
Methods	DCP (Soft matching)	RPMNet (Soft matching)	PRNet (Hard matching)	**Ours** (Hard matching)
Matrix Constraints	$m_{ij} \in [0,1]$ $\sum_j^{N_\mathcal{Y}} m_{ij} = 1$	$m_{ij} \in [0,1]$ $\sum_i^{N_\mathcal{X}} m_{ij} \in [0,1]$ $\sum_j^{N_\mathcal{Y}} m_{ij} \in [0,1]$	$m_{ij} \in \{0,1\}$ $\sum_j^{N_\mathcal{Y}} m_{ij} = 1$	$m_{ij} \in \{0,1\}$ $\sum_i^{N_\mathcal{X}} m_{ij} \in \{0,1\}$ $\sum_j^{N_\mathcal{Y}} m_{ij} \in \{0,1\}$
Matching Results				

Table 1: Comparison of learning-based point cloud registration methods based on "soft" matching and "hard" matching. Points with different colors indicate the source (green), target (blue), and virtual points (red). Connection line indicates the correspondences. m_{ij} is the entry of matching matrix $\mathbf{M} \in \mathbb{R}^{N_\mathcal{X} \times N_\mathcal{Y}}$, where $N_\mathcal{X}$, $N_\mathcal{Y}$ are size of the source and target. (The matching result of ours is returned by SHM_{DCP}.)

mitted. To avoid this ambiguity trap, DeepVCP (Lu et al. 2019) takes an unconvincing assumption, *i.e*. accurate initial motion parameters are provided as prior. PRNet (Wang and Solomon 2019b) and CorrNet3D (Zeng et al. 2021) advocate turning the learned soft matching matrix to a hard matrix by taking the most similar points as the corresponding points of the source points. However, this strategy leads to the drawback of one-to-many matching.

In this paper, we first theoretically analyze the inherent ambiguity in these soft matching-based methods, and then devote to achieving the real PPM matrix to handle the outliers and prevent ambiguity synchronously for robust 3D point cloud registration. However, as mentioned above, two problems block our pace. 1) Although it is well-known that the full rank permutation matrix can be solved by existing hard assignment algorithms, how to solve a PPM has yet not been explored; 2) PPM is defined in the discrete space, which is non-differentiable. To resolve these issues, we design a dedicated soft-to-hard (S2H) matching procedure consists of `S-step`: solving the soft matching matrix, and `H-step`: projecting this soft matrix to the PPM. Specifically, we propose to augment the profit matrix before the hard assignment to solve an augmented permutation matrix, which is cropped to achieve the final real PPM. Moreover, to guarantee end-to-end learning, we supervise the learned final PPM but propagate the gradient to the soft matrix instead. Our matching procedure can be easily integrated with existing 3D registration frameworks, which has been verified in DCP, RPMNet, and DGR (Choy, Dong, and Koltun 2020). Extensive experiments on benchmark datasets show that our method achieves state-of-the-art performance.

Our main contributions can be summarized as follows:

- We theoretically analyze the inherent ambiguity in the soft matching-based methods, which causes serious degeneration of the learned virtual corresponding points.
- We propose a novel S2H matching procedure to learn the PPM, which handles the outliers and prevents ambiguity synchronously. This matching procedure not only solves a real PPM, but also guarantees end-to-end learning.
- Remarkable performance on benchmark datasets verifies the proposed method, which achieves state-of-the-art performance in robust 3D point cloud registration.

Related Works

Herein, we briefly review the learning-based point cloud registration methods. More detailed summaries have been provided in (Rusinkiewicz and Levoy 2001; Pomerleau, rancis Colas, and Siegwart 2015; Zhang, Dai, and Sun 2020).

Correspondences-free methods: These methods estimate the rigid motion by comparing the global representations of input two point clouds, and generally consist of two stages: global feature extraction and rigid motion estimation. PointNetLK (Aoki et al. 2019) utilizes PointNet (Qi et al. 2017a) to extract global features, and then a modified LK algorithm is applied to solve the rigid motion. From the perspective of reconstruction, Huang *et al*. (Huang, Mei, and Zhang 2020) utilize an encoder-decoder structure network to learn a more comprehensive global feature.

Correspondences-based methods: These methods estimate the rigid motion based on correspondences, which generally consists of feature extractor, correspondences building, and rigid motion estimation modules. For *feature extractors*, various well-designed networks are used, such as set abstraction module (Qi et al. 2017b; Yew and Lee 2018), DGCNN (Wang et al. 2019; Wang and Solomon 2019a), FCGF (Choy, Park, and Koltun 2019; Choy, Dong, and Koltun 2020), globally informed 3D local feature (Deng, Birdal, and Ilic 2018b), KPConv (Thomas et al. 2019; Bai et al. 2020), capsule network (Zhao et al. 2020) and various rotation invariant features (Deng, Birdal, and Ilic 2018a; Gojcic et al. 2019). With the rise of deep learning, learning-based feature extractors have approached standard components, which can be integrated easily. For *correspondence building*, both soft matching-based (Lu et al. 2019; Wang and Solomon 2019a) and hard matching-based (Wang and Solomon 2019b) methods are representative. (Yew and Lee 2020; Huang et al. 2021) build correspondences on the identified inliers only. Besides, reliable correspondences selection is the widely adopted subsequent step. It is achieved by learning the reliability weight of each initial correspondence (Pais et al. 2020; Choy, Dong, and Koltun 2020; Probst et al. 2019) or selecting consistent correspondences (Deng, Birdal, and Ilic 2018b; Bai et al. 2021). For *motion estimation*, Procrustes (Gower 1975) is the most widely used algorithm in (Wang and Solomon 2019b; Yew and Lee 2020). Recently, regressing the motion parameters directly has become a new hot spot (Pais et al. 2020).

Preliminaries

Given the source point cloud $\mathcal{X} = [\boldsymbol{x}_i]_{3\times N_\mathcal{X}}$ and the target point cloud $\mathcal{Y} = [\boldsymbol{y}_j]_{3\times N_\mathcal{Y}}$, where $N_\mathcal{X}$ and $N_\mathcal{Y}$ represent the sizes of the two point clouds, 3D point cloud regis-

tration aims at solving a rigid motion to best align $\mathcal{X}$ with $\mathcal{Y}$. Here, we model the rigid motion by the rotation matrix $\mathbf{R} \in SO(3)$ and the translation vector $\mathbf{t} \in \mathbb{R}^3$. Since the Procrustes algorithm (Gower 1975) can optimally solve the rigid motion based on the correspondences, point matching becomes crucial, which is formulated as searching for a matching matrix $\mathbf{M} = [m_{ij}]_{N_\mathcal{X} \times N_\mathcal{Y}}$ to relate the source and target point clouds, and the entry $m_{ij} = 1$ or 0 indicates point $\boldsymbol{x}_i$ and point $\boldsymbol{y}_j$ are correspondence or not.

Ideally, $\mathcal{X}$ and $\mathcal{Y}$ are consistent, *i.e.* points in $\mathcal{X}$ and $\mathcal{Y}$ are exactly one-to-one correspondence. In this case, the matching matrix $\mathbf{M}$ is a permutation matrix, *i.e.* $m_{ij} \in \{0, 1\}$, $\sum_{i=1}^{N} m_{ij} = 1, \forall j$ and $\sum_{j=1}^{N} m_{ij} = 1, \forall i$, where $N_\mathcal{X} = N_\mathcal{Y} = N$. However, in practical applications, outliers always exist without corresponding points. They challenge the matching problem, and turn it into a special assignment problem while the desired matching matrix becomes a PPM for outliers pruning. We reformulate this special PPM $\mathbf{M}$ as,

$$\begin{aligned} &\{\mathbf{M} \mid m_{ij} \in \{0, 1\}; \\ &\quad \sum\nolimits_{i=1}^{N_\mathcal{X}} m_{ij} \in \{0, 1\}, \forall j; \sum\nolimits_{j=1}^{N_\mathcal{Y}} m_{ij} \in \{0, 1\}, \forall i\}. \end{aligned} \tag{1}$$

If the point is an inlier, the sum of the corresponding row/-column equals to 1. Otherwise, the point is an outlier and the sum of the corresponding row/column equals to 0.

Ambiguity in soft matching-based methods

As introduced above, different from the hard matching-based methods that build correspondences on the real points, the soft matching-based methods use the virtual corresponding points instead. Specifically, these soft matching-based methods generate a "soft map" between the source and target, *i.e.* $\boldsymbol{x}_i \in \mathcal{X}$ is assigned to $\mathcal{Y}$ by a probability vector. In this case, the matching matrix $\mathbf{M}$ becomes a soft probability matrix $\mathbf{P}$. In DCP (Wang and Solomon 2019a), $\mathbf{P}$ is a single stochastic matrix, where $p_{ij} \in [0, 1]$, $\sum_{i=1}^{N_\mathcal{X}} p_{ij} = 1, \forall j$. In RPMNet (Yew and Lee 2020), $\mathbf{P}$ is optimized to a partial doubly stochastic matrix (notated as PDSM) by the Augmented-Sinkhorn algorithm, where $p_{ij} \in [0, 1]$, $\sum_{i=1}^{N_\mathcal{X}} p_{ij} \leq 1, \forall j$ and $\sum_{j=1}^{N_\mathcal{Y}} p_{ij} \leq 1, \forall i$. Then, the virtual corresponding points $\mathcal{Y}'$ are obtained by performing weighted average on $\mathcal{Y}$ using $\mathbf{P}$, *i.e.* $\mathcal{Y}' = \mathcal{Y}\mathbf{P}^{\mathrm{T}}$.

However, there is an ambiguity trap here. The geometric constraints and underlying physics are ignored by relaxing the hard matching matrix to a soft probability matrix, hence, the distribution of virtual corresponding points is not unique resulting in serious degeneration. We can conclude as follows. The theoretical proof and more analysis are provided in *supplementary materials*.

Theorem 1 *Considering the consistent subset point clouds* $\mathbf{X}$*,* $\mathbf{Y}$ *with ground truth motion* $\mathbf{R}$*,* $\mathbf{t}$*, there exists more than one soft matching matrix* $\mathbf{P}$ *satisfying* $\mathbf{Y}\mathbf{P}^{\mathrm{T}} = \mathbf{R}\mathbf{X} + \mathbf{t}$.

Since there will be an infinite number of virtual point cloud distributions corresponding to the same rigid motion, this inherent ambiguity will cause serious degeneration of virtual points as shown in Table 1. Essentially, the process of weighted average can also be regarded as a special deformation of the point cloud. Although a seemingly good transformation estimation is obtained (Yew and Lee 2020), this process violates the rigid motion assumption, which cannot be supported by the Procrustes algorithm (Gower 1975).

Soft-to-Hard Matching for Registration

In this paper, we devote to solving a PPM to handle the outliers and prevent the ambiguity synchronously and thus design a meticulous S2H matching procedure. In this section, to clearly present S2H and how to integrate it into existing registration pipeline, we give a pipeline example as shown in Fig. 1, which consists of three core modules: feature extractor and similarity matrix solving, S2H matching, and rigid transformation estimation. Finally, the proposed loss function is also presented herein.

Feature Extractor and Similarity Matrix Solving. To achieve robust point matching, distinguished descriptors are crucial. With the rise of deep learning technique, many standard modules for deep features are proposed. These feature extractors can be easily integrated into our robust point cloud registration pipelines according to different requirements.

We denote the point features of the source and target as $\Phi_\mathcal{X} \in \mathbb{R}^{N_\mathcal{X} \times c}, \Phi_\mathcal{Y} \in \mathbb{R}^{N_\mathcal{Y} \times c}$. c is the feature dimension. Then, based on a certain similarity metric *e.g.* scale dot product attention (Vaswani et al. 2017), the similarity matrix is returned as $\mathbf{S} = [s_{ij}]_{N_\mathcal{X} \times N_\mathcal{Y}}$, where each entry s_{ij} represents the similarity between points $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_i \in \mathcal{Y}$.

S2H Matching. A well-known solution to hard matching is to formulate it as a special assignment problem, *i.e.* zero-one integer programming problem. However, two natural but challenging problems exist for learning-based pipeline. 1) Existing integer programming algorithms usually achieve a row or column full rank permutation matrix ($\mathrm{rank}(\mathbf{M}) = \min(N_\mathcal{X}, N_\mathcal{Y})$) rather than a PPM. It means all points of the source or target will be assigned corresponding points without distinguishing inliers and outliers. 2) PPM is defined in the discrete space, which is non-differentiable. This characteristic is fatal for the deep learning pipeline. To solve these problems, we propose to augment the profit matrix to solve the PPM and design an S2H matching procedure for end-to-end learning as follows.

• **Augmenting the profit matrix to solve PPM:** Conventionally, the matching task is formulated as a zero-one integer programming problem taking the similarity matrix $\mathbf{S}$ as the profit matrix, *i.e.*,

$$\mathbf{M}^* = \arg\max_{\mathbf{M} \in \mathcal{M}_N} < \mathbf{M}, \mathbf{S} >_F, \tag{2}$$

where $\mathcal{M}_N$ denotes the set of partial permutation matrices. $< \mathbf{M}, \mathbf{S} >_F = \mathrm{trace}(\mathbf{M}^\mathbf{T}\mathbf{S})$ denotes the (Frobenius) inner product. Note that the traditional assignment algorithm will achieve a full rank permutation solution. In response to outliers, we propose to augment the profit matrix by adding additional rows and columns to solve an augmented permutation matrix. Then the PPM will be returned by cropping this augmented permutation matrix as shown in Fig. 1. Thus, the following two questions should be addressed properly.

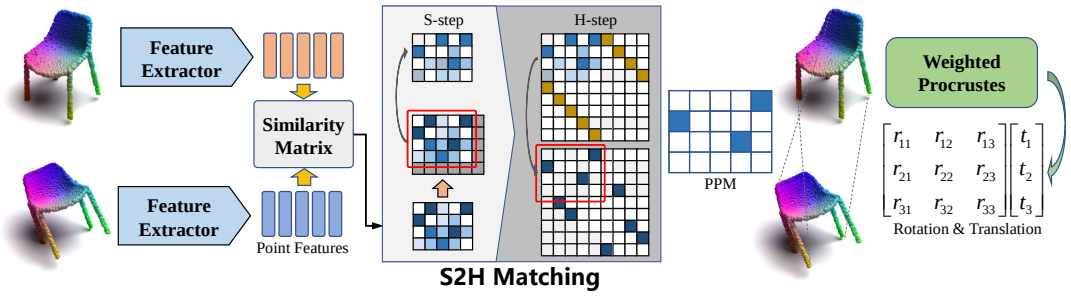


Figure 1: Illustration of the S2H matching procedure in registration pipeline. Given the source and target point clouds, the similarity matrix is obtained based on the point features. Then, S2H is applied for the final PPM. Finally, the rigid motion is estimated by the weighted Procrustes.

1. How many rows and columns should be added? In the Augmented-Sinkhorn algorithm (Yew and Lee 2020), which solves a soft matching matrix, one row and one column are added as "trash bin". However, for partial permutation matching, the number of rows and columns augmented to the profit matrix is more crucial since it implies the upper bound of the number of outliers potentially. Thus, we propose to supplement the original $N_{\mathcal{X}} \times N_{\mathcal{Y}}$ profit matrix to a $(N_{\mathcal{X}} + N_{\mathcal{Y}}) \times (N_{\mathcal{X}} + N_{\mathcal{Y}})$ matrix, which is a maximum redundant operation replying to the case of all points are outliers. Specifically, as shown in Fig. 1, left upper is the original matrix, right upper block is a $N_{\mathcal{X}} \times N_{\mathcal{X}}$ diagonal matrix and left bottom block is a $N_{\mathcal{Y}} \times N_{\mathcal{Y}}$ diagonal matrix, and right bottom block is a $N_{\mathcal{Y}} \times N_{\mathcal{X}}$ zero matrix.

2. What value to set? As aforesaid, the right upper block and the left bottom block are two diagonal matrices, what values should be set to these diagonal positions? Note that these values are roughly used as thresholds to distinguish outliers and inliers potentially. Meanwhile, we observe that in the profit matrix, if the row/column corresponds to an outlier, the entries in this row/column approximate a uniform distribution with low value, which means the outliers have no similar points in another point cloud. Otherwise, if the row/column corresponds to the inlier, in this row/column, the entries are close to a unimodal distribution, which means the inlier has only one similar point in another point cloud ideally. Thus, we propose to self-adaptively fill the diagonal position with σ according to the corresponding row/column of the input profit matrix: $\sigma = 1/\operatorname{var}(\boldsymbol{v})$, $\boldsymbol{v}$ is the corresponding row/column vector, and $\operatorname{var}(\cdot)$ is the variance function. In this case, for outlier, the filled value is large, which enforces the learned augmented permutation matrix to make the value of this position as one to gain more profit. For inlier, the filled value is low, the value of this position in the learned augmented permutation matrix is enforced to zero.

• **S2H matching in end-to-end learning:** Since $\mathbf{M}$ is defined in the discrete space, and the integer programming algorithm is non-differentiable, we design the S2H matching procedure to guarantee end-to-end learning. Specifically, `S-step` learns a soft matrix and `H-step` projects this soft matrix to discrete solution space for final PPM.

`S-step`**:** To deal with the outliers, we use Augmented-Sinkhorn (Yew and Lee 2020) to obtain a soft matrix by adding an additional row and column of ones to the input matrix during the Sinkhorn normalization. In Augmented-Sinkhorn, the additional row and column are regarded as "trash bin", and the matching weights of outliers are expected to "flow" to these additional row and column to distinguish outliers and inliers. Specifically, given the obtained similarity matrix $\mathbf{S} \in \mathbb{R}^{N_{\mathcal{X}} \times N_{\mathcal{Y}}}$, this soft matrix $\mathbf{P} \in \mathbb{R}^{N_{\mathcal{X}} \times N_{\mathcal{Y}}}$ is achieved by cropping the output $(N_{\mathcal{X}} + 1) \times (N_{\mathcal{Y}} + 1)$ matrix as shown in Fig. 1. $\mathbf{P}$ is a PDSM.

`H-step`**:** The resultant PDSM $\mathbf{P}$ is still a soft matrix, and we project it to PPM by applying the proposed profit matrix augmenting strategy to $\mathbf{P}$ and solving this zero-one integer programming problem. In our implementation, we chose the classical Hungarian algorithm (Kuhn 1955) to solve this assignment problem. After cropping the output augmented matrix, the final PPM $\mathbf{M} \in \mathbb{R}^{N_{\mathcal{X}} \times N_{\mathcal{Y}}}$ is obtained.

For a clearer understanding, we stress the ingeniousness of the proposed S2H matching structure from two folds:

1. End-to-end learning. We propose a deceptive operation as shown in the left of Fig. 2 to guarantee end-to-end learning. During the forward propagation, the loss is calculated based on the learned PPM $\mathbf{M}$. However, during the backward propagation, the gradient is not propagated to the PPM, but directly skipped to the learned PDSM $\mathbf{P}$ instead. This ingenious structure guarantees the accuracy of the calculated loss and the backward propagation simultaneously.

2. Hard Matching vs. S2H Matching. To solve a PPM, we give a more straightforward hard matching method in the right of Fig. 2, which can also make sense by learning $\mathbf{M}$ considering the augmented similarity matrix as the profit matrix. Nonetheless, there is an obvious local supervision risk in this case. That is, the gradient will be propagated to the input similarity matrix directly if only use the hard matching. And the correlation of entries, which is considered in the integer programming process, is ignored in backward propagation. In other words, the loss calculated from m_{ij} can only supervise s_{ij}. This will result in only a few sparse points being supervised by the ground truth and the remaining positions will be trained without supervision. For example, assuming $m_{ij} = 1$, the gradient will be propagated to s_{ij} directly, and enforce the feature extractor to enlarge s_{ij}. However, the positions where $m_{ij} = 0$ are not supervised (refer to *loss function section* for more details). Hence, the corresponding point features will not suppress their similarity. An extreme result is that all entries of the

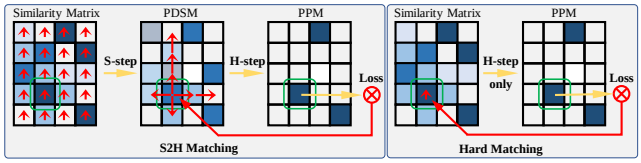


Figure 2: Comparison of our S2H matching (Left) and hard matching (Right). Yellow and red lines represent the forward and backward propagation respectively.

similarity matrix are very large since the point features lose the distinctiveness and each point in $\mathcal{X}$ is very similar to all points in $\mathcal{Y}$. This will result in a divergence of training.

Our S2H solution effectively avoids the local supervision risk by propagating the gradient to all entries in `S-step`, *i.e*. the correlation of all entries is reconsidered. As shown in the left of Fig. 2, during the backward propagation, the similarity of the correct correspondence will be boosted, meanwhile the similarity of the incorrect correspondence will be effectively suppressed.

Weighted Procrustes. When we get a PPM $\mathbf{M}$, the corresponding point set of $\mathcal{X}$ could be achieved as $\mathcal{Y}' = \mathcal{Y}\mathbf{M}^{\mathrm{T}}$. However, the corresponding points of outliers in $\mathcal{Y}'$ are obtained as $[0,0,0]^{\mathrm{T}}$, which should be filtered out when estimating the transformation. To this end, the weighted Procrustes algorithm is used here. Given $\mathcal{X}$, $\mathcal{Y}'$, the weight vector $\boldsymbol{w}$ is obtained as $\boldsymbol{w} = \sum_{j=1}^{N_{\mathcal{Y}}} m_{ij}$, where $\boldsymbol{w} \in \{0,1\}^{N_{\mathcal{X}}}$.

Then, $\boldsymbol{w}$ is normalized to $\bar{\boldsymbol{w}}$. Inspired by DGR (Choy, Dong, and Koltun 2020), the rigid motion is computed as follows: $\mathbf{H} = \mathcal{Y}'\mathbf{K}\mathbf{W}\mathbf{K}\mathcal{X}^{\mathrm{T}}$, where $\mathbf{W} = \mathrm{diag}(\bar{\boldsymbol{w}})$, $\mathbf{K} = \mathbf{I} - \sqrt{\bar{\boldsymbol{w}}}\sqrt{\bar{\boldsymbol{w}}}^{\mathrm{T}}$. $\mathbf{I}$ is an identity matrix. Then, $\mathbf{R} = \mathbf{U}\mathbf{E}\mathbf{V}^{\mathrm{T}}$, $\mathbf{t} = (\mathcal{Y}' - \mathbf{R}\mathcal{X})\mathbf{W}\mathbf{1}$, where $\mathbf{U}\mathbf{D}\mathbf{V}^{\mathrm{T}} = \mathrm{SVD}(\mathbf{H})$, $\mathbf{1} = (1,...,1)^{\mathrm{T}}$ and $\mathbf{E} = \mathrm{diag}(1,...,1,\det(\mathbf{U})\det(\mathbf{V}))$.

Loss function. In this paper, we supervise the matching matrix directly, which is defined as:

$$\mathcal{L}_1 = -\frac{\sum_{i=1}^{N_{\mathcal{X}}}\sum_{j=1}^{N_{\mathcal{Y}}}\left(m_{ij}^{\mathrm{pred}} m_{ij}^{\mathrm{gt}}\right)}{\sum_{i=1}^{N_{\mathcal{X}}}\sum_{j=1}^{N_{\mathcal{Y}}}\left(m_{ij}^{\mathrm{gt}}\right)}, \tag{3}$$

where the superscript "pred" and "gt" represent the prediction and ground truth respectively. It is noteworthy that when $m_{ij}^{\mathrm{gt}} = 1$, m_{ij}^{pred} is enforced to 1. But when $m_{ij}^{\mathrm{gt}} = 0$, m_{ij}^{pred} diverges without supervision. Moreover, due to the introduction of augmentation operation, all points tend to be labeled as outliers, which makes the learned PPM close to a full zero matrix. To avoid this degeneracy, we encourage the number of inliers by:

$$\mathcal{L}_2 = -\frac{\sum_{i=1}^{N_{\mathcal{X}}}\sum_{j=1}^{N_{\mathcal{Y}}}(m_{ij}^{\mathrm{pred}})}{N_{\mathcal{X}} + N_{\mathcal{Y}}}. \tag{4}$$

Besides, we also supervise the final rigid motion, *i.e*.,

$$\mathcal{L}_3 = \|\mathbf{R}^{\mathrm{gt}^{\mathrm{T}}}\mathbf{R}^{\mathrm{pred}} - \mathbf{I}_3\|_2 + \|\mathbf{t}^{\mathrm{gt}} - \mathbf{t}^{\mathrm{pred}}\|_2, \tag{5}$$

where $\mathbf{I}_3$ is a 3×3 identity matrix. Then, our final loss function is reached as $\mathcal{L} = \lambda_1\mathcal{L}_1 + \lambda_2\mathcal{L}_2 + \lambda_3\mathcal{L}_3$, the trade-off parameters are set to $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in this paper.

Experiments and Evaluation

In this section, we evaluate the proposed S2H matching procedure in several representative point cloud registration frameworks including DCP, RPMNet and DGR, on benchmark datasets including ModelNet40 (Wu et al. 2015), 3DMatch (Zeng et al. 2017) and KITTI (Geiger et al. 2013).

Implementation details. To validate the proposed matching method can be generally integrated, we evaluate S2H matching within three typical frameworks, *i.e*. DCP, RPMNet, and DGR, notated as $\mathrm{SHM}_{\mathrm{DCP}}$, $\mathrm{SHM}_{\mathrm{RPMNet}}$, and $\mathrm{SHM}_{\mathrm{DGR}}$ respectively. These three frameworks are typical and representative, where DCP and RPMNet are soft matching-based methods using different feature extractors, and mainly concentrate on the synthetic dataset, ModelNet40. DGR is a hard matching-based method focusing on large-scale real datasets, 3DMatch and KITTI. Note that PRNet (Wang and Solomon 2019b) also inherits the framework of DCP, which is compared with $\mathrm{SHM}_{\mathrm{DCP}}$ herein. The complete loss is used in $\mathrm{SHM}_{\mathrm{DCP}}$, $\mathrm{SHM}_{\mathrm{RPMNet}}$. Only $\mathcal{L}_3$ is used in $\mathrm{SHM}_{\mathrm{DGR}}$. Refer to *supplementary materials* for more details.

Evaluation on synthetic dataset: ModelNet40

In this section, we validate our proposed S2H matching procedure with $\mathrm{SHM}_{\mathrm{DCP}}$, $\mathrm{SHM}_{\mathrm{RPMNet}}$ on a synthetic dataset, ModelNet40. Following DCP and RPNet, we construct a point cloud by randomly sampling 1024 points, and then apply a rigid transformation to this point cloud, where the rotation and translation are uniformly sampled from $[0°, 45°]$ and $[-0.5, 0.5]^3$ respectively along each axis. Next, we randomly sample 768 points from the original point cloud and the transformed point cloud as the source and target to ensure the random distribution of the outliers.

In addition, following DCP and RPMNet, we test ours on two different dataset settings. 1) Unseen categories (*clean*): ModelNet40 will be divided into training and test splits based on the object category, *i.e*. the first 20 categories are selected for training and the rest categories for testing. 2) Noisy data (*noisy*): For robustness testing, random Gaussian noise (*i.e*., $\mathcal{N}(0, 0.01)$) is added to each point, while the sampled noise out of the range of $[-0.05, 0.05]$ will be clipped. The dataset splitting strategy is the same as *clean*.

Matching. Accurate correspondences estimation is crucial for robust point cloud registration. Here, we evaluate the constructed correspondences for a clear comparison.

• **Metric:** We calculate the discrepancy between the predicted and the ground truth corresponding points. The predicted corresponding points $\mathcal{Y}'_{\mathrm{pred}}$ of $\mathcal{X}$ can be obtained by two methods. First, based on the *predicted matching matrix* $\mathbf{M}^{\mathrm{pred}}$, $\mathcal{Y}'_{\mathrm{pred}}$ can be obtained by $\mathcal{Y}'_{\mathrm{pred}} = \mathcal{Y}\mathbf{M}^{\mathrm{pred}^{\mathrm{T}}}$. Second, inspired by the iteration strategy in ICP, $\mathcal{Y}'_{\mathrm{pred}}$ can be obtained based on the *predicted transformation* $\{\mathbf{R}^{\mathrm{pred}}, \mathbf{t}^{\mathrm{pred}}\}$ and *nearest neighbor principle*, *i.e*. $\mathcal{Y}'_{\mathrm{pred}} = \mathrm{NN}_{\mathcal{Y}}(\mathbf{R}^{\mathrm{pred}}\mathcal{X} + \mathbf{t}^{\mathrm{pred}})$, where $\mathrm{NN}_{\mathcal{Y}}(\cdot)$ solves the nearest neighbor point in $\mathcal{Y}$. Given the ground truth corresponding points $\mathcal{Y}'_{\mathrm{gt}}$, the root mean squared error (RMSE) and mean absolute error (MAE) in Euclidean distance between $\mathcal{Y}'_{\mathrm{pred}}$ and $\mathcal{Y}'_{\mathrm{gt}}$, notated as RMSE(dis) and MAE(dis) are presented.

We also report the matching recall (%) based on a proposed self-adaptive threshold, $\tau_i = (1/K)\sum_{j=1}^{K} d_{\boldsymbol{y}'_{\pi(i)},\boldsymbol{y}_j}$, where $\boldsymbol{y}'_{\pi(i)}$ is the correct corresponding point of $\boldsymbol{x}_i$, $\boldsymbol{y}_j \in \text{KNN}_{\mathcal{Y}}(\boldsymbol{y}'_{\pi(i)})$, $\text{KNN}_{\mathcal{Y}}(\cdot)$ solves the K-nearest neighbor points (exclude the self-point) in $\mathcal{Y}$ with pre-defined parameter K, $d_{\boldsymbol{y}'_{\pi(i)},\boldsymbol{y}_j}$ is the distance between $\boldsymbol{y}'_{\pi(i)}$ and $\boldsymbol{y}_j$. To sum up, for the i-th point, τ_i is computed as the average distance of K-nearest points around the correct corresponding point in $\mathcal{Y}$. If the distance between the correct corresponding point and the predicted one is less than the threshold, this pair will be confirmed as a correct matching pair.

Methods	RMSE(dis) ↓		MAE(dis) ↓		RMSE(dis) ↓		MAE(dis) ↓	
	clean	*noisy*	*clean*	*noisy*	*clean*	*noisy*	*clean*	*noisy*
DCP-v2	0.500	0.466	0.705	0.657	0.088	0.095	0.078	0.095
PRNet	0.311	0.303	0.357	0.351	0.046	0.057	0.023	0.036
SHM_{DCP}	**0.106**	**0.188**	**0.027**	**0.082**	**0.033**	**0.053**	**0.008**	**0.019**
RPMNet	0.139	0.137	0.182	0.181	0.019	0.028	0.004	0.013
$\text{SHM}_{\text{RPMNet}}$	**0.057**	**0.121**	**0.009**	**0.041**	**0.007**	**0.027**	**0.001**	**0.010**

Table 2: Discrepancy based on predicted matching matrix (Left) and predicted transformation, nearest neighbor principle (Right).

• **Evaluation:** In Table 2, we report RMSE(dis) and MAE(dis) results in *clean* and *noisy*. For the discrepancy based on the predicted matching matrix, our method improves the performance with a big margin in both DCP and RPMNet frameworks. These obtained results are reasonable because the DCP-v2, RPMNet are virtual point-based methods, where the corresponding points degenerate seriously with a large distance to correct corresponding points. PRNet presents weaker matching performance due to the one-to-many matching. For the discrepancy based on the predicted transformation and nearest neighbor principle, all methods achieve superior performance, whereas SHM_{DCP}, $\text{SHM}_{\text{RPMNet}}$ remain the overall best performance.

Besides, we also draw the matching recall with different thresholds in Fig. 3. For the results based on the predicted matching matrix, the DCP-v2, RPMNet fail to obtain accurate correspondences. And ours obtains the best results. For the results based on the predicted transformation and nearest neighbor principle, all methods achieve better performance, and SHM_{DCP}, $\text{SHM}_{\text{RPMNet}}$ remain the best performance.

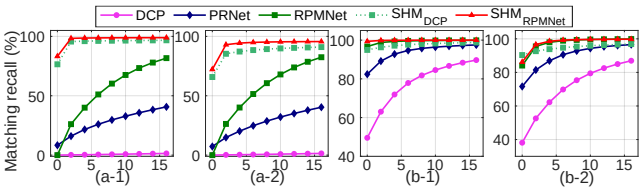


Figure 3: Matching recall with different thresholds based on the predicted matching matrix ((a-1), (a-2)) and the predicted transformation, nearest neighbor principle ((b-1), (b-2)). (a-1), (b-1) are conducted in *clean* and (a-2), (b-2) in *noisy*. The X-axis is the number of K-nearest points for threshold, and $K = 0$ means $\tau_i = 0$.

Registration. In this section, we evaluate the rigid motion estimation performance of SHM_{DCP}, $\text{SHM}_{\text{RPMNet}}$.

• **Metric:** Following DCP, RMSE and MAE between the ground truth and prediction in Euler angle and translation vector are used as the evaluation metrics here, notated as RMSE(R), MAE(R), RMSE(t) and MAE(t) respectively.

• **Evaluation:** The evaluation results are provided in Table 3. Both SHM_{DCP} and $\text{SHM}_{\text{RPMNet}}$ outperform the handcraft registration methods, ICP (Besl and McKay 1992), FGR (Zhou, Park, and Koltun 2016). Furthermore, in *clean*, SHM_{DCP} achieves more accurate performance than DCP-v2 and PRNet in all metrics, $\text{SHM}_{\text{RPMNet}}$ is more accurate than RPMNet in all metrics. In *noisy*, we train and test ours and all baselines with noisy data. And the methods with our S2H matching also achieves the best performance in these two frameworks in all evaluation metrics. These evaluations validate the improvement of our S2H matching, which creates a new state-of-the-art results. More intuitively, we provide some qualitative registration performance in Fig. 4.

Methods	RMSE(R) ↓		MAE(R) ↓		RMSE(t) ↓		MAE(t) ↓	
	clean	*noisy*	*clean*	*noisy*	*clean*	*noisy*	*clean*	*noisy*
ICP	12.545	12.723	5.438	5.298	0.046	0.045	0.024	0.023
FGR	20.042	40.829	7.203	21.065	0.035	0.060	0.019	0.039
DCP-v2	6.265	6.347	3.990	4.294	0.014	0.016	0.011	0.012
PRNet	3.532	4.321	1.760	1.826	0.013	0.013	0.010	0.010
SHM_{DCP}	**2.522**	**3.886**	**0.833**	**1.510**	**0.005**	**0.006**	**0.003**	**0.004**
	+28.60%	+10.07%	+52.67%	+17.31%	+61.54%	+53.85%	+70.00%	+60.00%
RPMNet	0.886	1.631	0.345	0.565	0.006	0.011	0.003	0.004
$\text{SHM}_{\text{RPMNet}}$	**0.514**	**1.456**	**0.247**	**0.378**	**0.004**	**0.008**	**0.002**	**0.003**
	+41.99%	+10.73%	+28.41%	+33.10%	+33.33%	+27.27%	+33.33%	+25.00%

Table 3: Registration perfomance on *clean*, *noisy*. Green notes the improvement comparing with the second-top results.

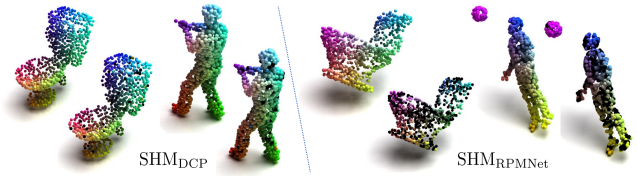


Figure 4: For each pair of point clouds, left upper is the transformed source using the predicted rigid motion and right bottom is the target. For a more clear presentation of the outliers, we only sample the target of the input pair, *i.e.* $N_{\mathcal{X}} = 1024$, $N_{\mathcal{Y}} = 768$. The same color represents the correspondences, and black indicates the abandoned points. Ours achieves advanced correspondences building and robust registration results in such challenging cases.

• **Time-efficiency:** We counted the average inference time of learning-based methods in Table 4. The experiments are conducted in *noisy* on ModelNet40 using a Xeon E5-2640 v4@2.40GHz CPU and a GTX 1080Ti. S2H matching-based methods are slower because the integer programming algorithm is time-consuming. $\text{SHM}_{\text{RPMNet}}$ is slower than SHM_{DCP} due to the iteration in RPMNet framework.

Methods	DCP	PRNet	SHM_{DCP}	RPMNet	$\text{SHM}_{\text{RPMNet}}$
Time(ms)	**10.66**	45.48	98.03	54.75	409.95

Table 4: Inference time comparison of learning-based methods.

Evaluation on real indoor data: 3DMatch

In this part, we evaluate our S2H matching in SHM_{DGR} on real indoor dataset, 3DMatch (Zeng et al. 2017). SHM_{DGR} takes the S2H matching replacing the original matching of DGR (Choy, Dong, and Koltun 2020) (more details can be seen in *supplementary materials*). Following DGR, the input point clouds have been voxelized with the voxel size of 5cm, then each of them contains $\sim$50k points.

• **Metric:** For a fair comparison, we follow the protocols and evaluation metrics of DGR here. The average Rotation Error (RE: $\arccos((trace(\mathbf{R}^{\text{pred}^{-1}}\mathbf{R}^{\text{gt}}) - 1)/2)\frac{180}{\pi}$), average Translation Error (TE: $\|\mathbf{t}^{\text{pred}} - \mathbf{t}^{\text{gt}}\|_2^2$), recall, and time-efficiency are reported. Recall indicates the ratio of successful pairwise registrations. Here, the successful pair is confirmed if the RE and TE are smaller than pre-defined thresholds (*i.e.*, RE$<15^\circ$, TE$<30cm$).

Methods	**TE**(cm) ↓	**RE**(deg) ↓	**Recall**(%) ↑	**Time**(s) ↓
ICP	18.1	8.25	6.04	0.25
FGR	10.6	4.08	42.7	0.31
Go-ICP	14.7	5.38	22.9	771.0
Super4PCS	14.1	5.25	21.6	4.55
RANSAC	9.16	2.95	70.7	2.32
DCP-v2	21.4	8.42	3.22	**0.07**
PointNetLK	21.3	8.04	1.61	0.12
DGR w/o safeguard	7.73	2.58	85.2	0.70
DGR	7.34	2.43	91.3	1.21
PointDSC	6.55	2.06	**93.28**	0.09
SHM_{DGR}	**6.41**	**1.75**	91.7	37.2

Table 5: Evaluation on 3DMatch dataset.

• **Evaluation:** From the Table 5, we find that ICP achieves the weak registration result since the dataset contains challenging sequences with large motion while no reliable prior is provided. Super4PCS (Mellado, Aiger, and Mitra 2014), and Go-ICP (Yang et al. 2015), which are sampling-based algorithm and the variant of ICP with branch-and-bound search respectively, present similar performance here. FGR and RANSAC perform better due to the extracted point features. 3DRegNet (Pais et al. 2020) is also tested here, however, it does not converge, which outputs the error above 30° and 1m. DGR is the state-of-the-art learning-based method, which is designed for scene data specifically. However, DGR also takes the most similar points as the corresponding points ignoring the one-to-one matching principle. SHM_{DGR} achieves better registration performance including transformation estimation and recall comparing with the original DGR. PointDSC (Bai et al. 2021) achieves the best recall but the registration performance is weaker than SHM_{DGR}.

Evaluation on real outdoor data: KITTI

We evaluate SHM_{DGR} on the real outdoor data, KITTI (Geiger et al. 2013). Here, we also follow the protocols of DGR, where the evaluation metrics are identical to the 3DMatch evaluation. The thresholds to confirm the successful pair are set to 0.6m and 5°. From Table 6, the SHM_{DGR} achieves the best performance with respect to rigid transformation estimation and recall, which outperforms the original DGR and the state-of-the-art method, PointDSC.

Methods	**TE**(cm) ↓	**RE**(deg) ↓	**Recall**(%)↑	**Time**(s) ↓
FGR	40.7	1.02	0.2	1.42
RANSAC	25.9	1.39	34.2	1.37
FCGF	10.2	0.33	98.2	6.38
DGR	21.7	0.34	96.9	2.29
PointDSC	20.94	0.33	98.20	**0.31**
SHM_{DGR}	**9.32**	**0.28**	**99.3**	52.4

Table 6: Evaluation on KITTI dataset.

Ablation studies

End-to-end vs. post-processing. In this paper, we advocate learning the one-to-one matching in an end-to-end manner, *i.e*. achieving the PPM in the matching stage. Oppositely, another natural idea is to solve the PPM in post-processing, *i.e*. using only the `S-step` to learn the soft matrix during the training and adding the `H-step` during the test for the final hard matrix, PPM. The comparison of these two ideas is given in Table 7, where the experiments are conducted on ModelNet40. We can see that the end-to-end learning achieves better results in all metrics because more accurate loss are calculated in this pipeline.

Methods	$\mathbf{RMSE}(\mathbf{R})\downarrow$		$\mathbf{MAE}(\mathbf{R})\downarrow$		$\mathbf{RMSE}(\mathbf{t})(\times 10^{-3})\downarrow$		$\mathbf{MAE}(\mathbf{t})(\times 10^{-3})\downarrow$	
	clean	*noisy*	*clean*	*noisy*	*clean*	*noisy*	*clean*	*noisy*
$\text{SHM}^-_{\text{DCP}}$	3.325	4.570	1.039	1.683	5.056	6.624	3.204	4.316
$\text{SHM}^+_{\text{DCP}}$	**2.522**	**3.886**	**0.833**	**1.510**	**4.780**	**5.758**	**3.052**	**3.774**
	+24.15%	+14.97%	+19.83%	+10.28%	+5.46%	+13.07%	+4.74%	+12.56%
$\text{SHM}^-_{\text{RPMNet}}$	0.760	1.537	0.270	0.449	4.927	10.120	2.356	3.602
$\text{SHM}^-_{\text{RPMNet}}$	**0.514**	**1.456**	**0.247**	**0.378**	**3.883**	**8.421**	**2.328**	**3.145**
	+32.37%	+5.27%	+8.52%	+15.81%	+21.19%	+16.79%	+1.19%	+12.69%

Table 7: Comparison between the end-to-end learning and post-processing setting. "-" indicates the post-processing, and "+" indicates the end-to-end learning.

Other important ablation studies. Due to the limitation of space, we provide some other import experiments in *supplementary materials*, including the robustness to different outliers generation strategy, the robustness to different outliers ratio, the influence of different loss function combinations, and more qualitative results of registration, *etc*.

Conclusion

In this paper, we tackle the point matching problem in robust 3D point cloud registration. First, we analyze the inherent ambiguity in soft matching-based methods. Second, to resolve the ambiguity and handle the outliers in the matching stage, we propose to learn the partial permutation matching (PPM) matrix. To address the consequent problem that PPM is defined in non-differentiable space and cannot be solved by existing hard assignment algorithms, we design a soft-to-hard matching method. We have validated the effectiveness by integrating it with various registration frameworks including DCP, RPMNet, and DGR and conducting extensive experiments in both synthetic data (ModelNet40) and real scan data (3DMatch and KITTI), which created a new state-of-the-art performance for robust 3D point cloud registration. In the future, we plan to extend our framework to non-rigid point cloud registration.

Acknowledgments

This research was supported in part by National Key Research and Development Program of China (2018AAA0102803) and National Natural Science Foundation of China (61871325, 62001394, 61901387).

References

Aoki, Y.; Goforth, H.; Srivatsan, R. A.; and Lucey, S. 2019. PointNetLK: Robust & efficient point cloud registration using PointNet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7163–7172.

Bai, X.; Luo, Z.; Zhou, L.; Chen, H.; Li, L.; Hu, Z.; Fu, H.; and Tai, C.-L. 2021. PointDSC: Robust Point Cloud Registration Using Deep Spatial Consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 15859–15869.

Bai, X.; Luo, Z.; Zhou, L.; Fu, H.; Quan, L.; and Tai, C.-L. 2020. D3Feat: Joint Learning of Dense Detection and Description of 3D Local Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6358–6366.

Besl, P. J.; and McKay, N. D. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2): 239–256.

Choy, C.; Dong, W.; and Koltun, V. 2020. Deep Global Registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2511–2520.

Choy, C.; Park, J.; and Koltun, V. 2019. Fully Convolutional Geometric Features. In *Proceedings of the IEEE International Conference on Computer Vision*, 8958–8966.

Deng, H.; Birdal, T.; and Ilic, S. 2018a. PPF-FoldNet: Unsupervised Learning of Rotation Invariant 3D Local Descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 620–638. Springer.

Deng, H.; Birdal, T.; and Ilic, S. 2018b. PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 195–205.

Deschaud, J.-E. 2018. IMLS-SLAM: scan-to-model matching based on 3D data. In *Proceedings of the International Conference on Robotics and Automation*, 2480–2485.

Ding, L.; and Feng, C. 2019. DeepMapping: Unsupervised map estimation from multiple point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8650–8659.

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

Gojcic, Z.; Zhou, C.; Wegner, J. D.; and Wieser, A. 2019. The Perfect Match: 3D Point Cloud Matching With Smoothed Densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5545–5554.

Gower, J. C. 1975. Generalized procrustes analysis. *Psychometrika.*, 40(1): 33–51.

Huang, S.; Gojcic, Z.; Usvyatsov, M.; Wieser, A.; and Schindler, K. 2021. Predator: Registration of 3D Point Clouds With Low Overlap. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4267–4276.

Huang, X.; Mei, G.; and Zhang, J. 2020. Feature-metric Registration: A Fast Semi-supervised Approach for Robust Point Cloud Registration without Correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11363–11371.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2): 83–97.

Lu, W.; Wan, G.; Zhou, Y.; Fu, X.; Yuan, P.; and Song, S. 2019. DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration. In *Proceedings of the IEEE International Conference on Computer Vision*, 12–21.

Mellado, N.; Aiger, D.; and Mitra, N. J. 2014. Super 4PCS Fast Global Pointcloud Registration via Smart Indexing. *Computer Graphics Forum*, 33(5): 205–215.

Pais, G. D.; Ramalingam, S.; Govindu, V. M.; Nascimento, J. C.; Chellappa, R.; and Miraldo, P. 2020. 3DRegNet: A Deep Neural Network for 3D Point Registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7191–7201.

Pomerleau, F.; rancis Colas; and Siegwart, R. 2015. A Review of Point Cloud Registration Algorithms for Mobile Robotics. *Foundations and Trends in Robotics*, 4(1): 1–104.

Probst, T.; Paudel, D. P.; Chhatkuli, A.; and Gool, L. V. 2019. Unsupervised learning of consensus maximization for 3d vision problems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 929–938.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the Advances in Neural Information Processing Systems*, 5099–5108.

Rusinkiewicz, S.; and Levoy, M. 2001. Efficient variants of the ICP algorithm. In *Proceedings of the IEEE International Conference on 3D Digital Imaging and Modeling*, 145–152.

Sarlin, P.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. SuperGlue: Learning Feature Matching With Graph Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4937–4946.

Shiratori, T.; Berclaz, J.; Harville, M.; Shah, C.; Li, T.; Matsushita, Y.; and Shiller, S. 2015. Efficient large-scale point cloud registration using loop closures. In *Proceedings of the International Conference on 3D Vision*, 232–240.

Thomas, H.; Qi, C. R.; Deschaud, J.; Marcotegui, B.; Goulette, F.; and Guibas, L. J. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. In *ICCV*, 6410–6419.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, 5998–6008.

Wang, Y.; and Solomon, J. M. 2019a. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE International Conference on Computer Vision*, 3523–3532.

Wang, Y.; and Solomon, J. M. 2019b. PRNet: Self-supervised learning for partial-to-partial registration. In *Proceedings of the Advances in Neural Information Processing Systems*, 8814–8826.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5): 1–12.

Wong, J. M.; Kee, V.; Le, T.; Wagner, S.; Mariottini, G. L.; Schneider, A.; Hamilton, L.; Chipalkatty, R.; Hebert, M.; Johnson, D. M. S.; Wu, J.; Zhou, B.; and Torralba, A. 2017. SegICP: Integrated deep semantic segmentation and pose estimation. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 5784–5789.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1912–1920.

Yang, J.; Li, H.; Campbell, D.; and Jia, Y. 2015. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11): 2241–2254.

Yew, Z. J.; and Lee, G. H. 2018. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *Proceedings of the European Conference on Computer Vision*, 630–646.

Yew, Z. J.; and Lee, G. H. 2020. RPM-Net: Robust Point Matching Using Learned Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11821–11830.

Zeng, A.; Song, S.; Nießner, M.; Fisher, M.; Xiao, J.; and Funkhouser, T. A. 2017. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 199–208.

Zeng, Y.; Qian, Y.; Zhu, Z.; Hou, J.; Yuan, H.; and He, Y. 2021. CorrNet3D: Unsupervised End-to-end Learning of Dense Correspondence for 3D Point Clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6052–6061.

Zhang, Z.; Dai, Y.; and Sun, J. 2020. Deep learning based point cloud registration: an overview. *Virtual Reality and Intelligent Hardware.*, 2(3): 222–246.

Zhao, Y.; Birdal, T.; Lenssen, J. E.; Menegatti, E.; Guibas, L. J.; and Tombari, F. 2020. Quaternion Equivariant Capsule Networks for 3D Point Clouds. In *Proceedings of the European Conference on Computer Vision*, 1–19.

Zhou, Q.-Y.; Park, J.; and Koltun, V. 2016. Fast global registration. In *Proceedings of the European Conference on Computer Vision*, 766–782.