Learning 3D Point Cloud Registration as a Single Optimization Problem

Rintaro Yanagi[1], Atsushi Hashimoto[2], Naoya Chiba[3], Shusaku Sone[2], Jiaxin Ma[2], and Yoshitaka Ushiku[2]

[1] National Institute of Advanced Industrial Science and Technology (AIST)
[2] OMRON SINIC X Corp.
[3] Tohoku University

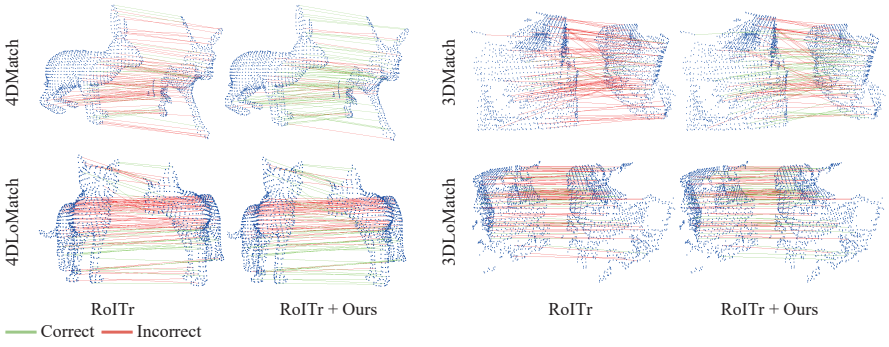


Fig. 1: Qualitative results on rigid/non-rigid whole/partial datasets by RoITr and its variant modified with our method. Samples were randomly selected to avoid cherry-picking. Green lines indicate exact matches, while red lines denote failures. Our method consistently improves performance.

Abstract. We present a simple yet effective technique to boost the performance of 3D point cloud registration. Conventional methods input a distance matrix to a differentiable matching algorithm deterministically, ignoring any uncertainty in upstream distance calculation. Consequently, current methods consider the optimalities of the feature extractor and the matching algorithm independently, leading to sub-optimal performance. We connect them via a non-deterministic information path. To make the algorithm uncertainty-aware, we employ a learning-based matching network module. This modification unifies the estimation process as a single optimization problem, where feature extractors and the matching network are jointly trained, reaching a joint-optimal solution. Experimental results show that our strategy significantly improves the performance of various conventional methods under multiple conditions, including rigid/non-rigid and whole/partial point cloud registration datasets.

1 Introduction

3D point cloud registration lies at the core of many downstream 3D computer vision applications such as motion transfer [39], shape editing [25], and object

localization for industrial robots [7]. The difficulty of 3D point cloud registration arises from the sparseness, sensing challenges, and non-rigid deformation of the 3D point cloud data [4,36,57]. Recent deep learning methods have tackled these issues, showing continuous improvement.

A deep-learning-based 3D point cloud registration method consists of a feature extractor and a differentiable matching algorithm. The feature extractor outputs point-wise features, which are then used to compute a distance matrix. This distance matrix is input into the matching algorithm. The algorithm identifies correspondences between input point clouds typically by solving a linear assignment problem that minimizes the sum of distances. The Hungarian algorithm [19] is known as an optimal solver for the problem; however, it is incompatible with feature extractor because it is not differentiable. Therefore, the Sinkhorn algorithm [5], a differentiable relaxation of the Hungarian algorithm, was initially applied to this task. Recent reports indicate that heuristic operations like DeSmooth [54] and Dual-Softmax [33] outperform the Sinkhorn algorithm despite its theoretical guarantee of the optimality.

We aim to develop a robust matching strategy that consistently works well with various feature extractors. Selecting matching algorithms empirically for each feature extractor is a common practice in recent state-of-the-art (SOTA) methods [22,23,51,54]. This observation suggests that *different feature extractors prefer different matching algorithms*. We attribute this to the different tendencies of uncertainties on the calculated distance matrix. In this context, relying solely on a deterministic distance matrix to link two modules is sub-optimal.

We provide a new 3D point cloud registration paradigm that unifies two modules into a single probabilistic process. Our method generalizes each value in the distance matrix into a vector that represents the distance with its uncertainty. This generalization connects the modules non-deterministically. A hand-crafted algorithm exists to solve a stochastic linear assignment problem [3]. However, it only functions when the distribution shape is known. Instead of using hand-crafted algorithms, we adopt a machine-learning-based matching module. For this purpose, we spotlight a network, WeaveNet [37], originally designed to approximately solve an NP-hard fair stable matching problem. We refine it as a solver of 3D point cloud registration tasks, focusing on memory efficiency and scaling to large-scale data, without requiring additional loss objectives.

The contribution of this paper is threefold.

1. We provide a paradigm that connects a matching module with a feature extractor in a non-deterministic manner, resulting in a unified probabilistic model without requiring additional loss objectives.
2. We develop a practical implementation of the above model by improving memory efficiency of an existing matching-aided neural network.
3. We confirm through exhaustive experiments that the proposed method enables significant performance boosts across various methods and conditions.

2 Related work

2.1 Feature extraction from 3D Point Cloud

There are many tasks on 3D point clouds, such as shape classification, 3D object detection, and point cloud segmentation, in addition to registration [12,13]. Early studies on 3D point cloud data focused on hand-crafted point-wise features (a.k.a., local descriptors) [18, 40, 47, 48] and key point detection [58]. A comprehensive survey [11] has revealed their characteristics but also highlighted limitations in performance due to noise, clutter, fluctuation in resolutions, and occlusions.

The recent growth in deep learning methods has led to significant improvements in feature extraction [12, 13], which has also influenced the registration task. First-generation deep-learning-based feature extractors for 3D matching relied on the network architecture developed for 2D images, where point cloud data are projected onto an image plane as the RGB-D image format [8] or handled 3D shape by voxel-based 3D CNNs [27, 53]. Then, PointNet [29] and DeepSets [52] were proposed, which enabled us to extract point-wise features directly from unstructured point clouds. Subsequently, second-generation studies enhanced the techniques for various tasks on 3D point clouds [14, 26, 30, 42, 44–46, 56]. In this context, feature extractors have been further improved for registration tasks. Huang *et al*. [15] introduced a cross-attention mechanism, which enables extraction of features interactively between two point clouds. This idea is followed by subsequent studies [22, 23, 50].

In addition to the feature extractor, the method for executing registration is also a subject of improvement. There are two main categories of registration approaches: parametric registration (i.e., rigid transform) [1, 35, 43] and non-parametric registration [49,54,55]. The parametric approach formulates the problem as estimating parameters that represent the transform of two point clouds. This approach cannot manage non-rigid deformations. The non-parametric approach captures deviations for each point, allowing for any non-rigid deformations. It is also beneficial for rigid cases; non-parametric registration results can serve as a good starting point for parametric methods. For instance, Li and Harada [22] demonstrated that applying RANSAC and ICP to results from non-parametric registration consistently improves rigid registration performance, albeit with increased computational time. This study specifically focuses on non-parametric methods, aiming to improve traditional methods through reconsidering the connection between feature extractors and matching algorithms.

2.2 Differentiable matching algorithm

Combining feature extractors and differentiable matching algorithms is a common practice not only for 3D point cloud registration, but also for other computer vision registration tasks. SuperGlue [34] is a notable 2D RGB image registration method that first utilized the **Sinkhorn** algorithm in the context of registration. In the following year's CVPR, two papers demonstrated that heuristic matching

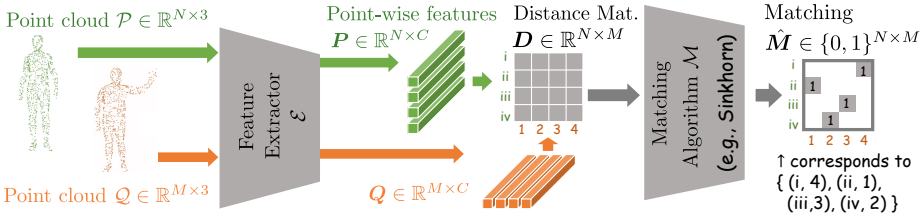


Fig. 2: Conventional architecture: a feature extractor $\mathcal{E}$ and a differentiable matching algorithm $\mathcal{M}$ are connected via a distance matrix calculation. We can regard $\boldsymbol{D}$ as an estimate of the true distance matrix, which divides the inference process into two independent optimization problems.

operations outperformed the Sinkhorn algorithm. The first report by Zeng *et al*. [54] proposed the **DeSmooth** operation, which outperforms Sinkhorn for 3D point cloud registration. The second one by Sun *et al*. [38] reported that the **dual softmax** operation [33] outperformed Sinkhorn for 2D image registration. **Lepard** [22] later introduced dual softmax for 3D point cloud registration. Its subsequent study, **LNDP** [23], also applied dual softmax. The current SOTA method, RoITr [51], incorporates coarse and fine matching modules inspired by CoFiNet [50]. CoFiNet originally used Sinkhorn for coarse matching, which RoITr replaced with dual softmax. Motivated by the superior performance of heuristic matching operations compared to Sinkhorn, we investigate the potential of learning-based matching algorithms.

Despite the popularity of registration tasks in the vision community, learning-based matching algorithms remain underexplored. A matching problem is defined on a bipartite graph, and hence, a network dealing with this problem can be regarded as a kind of graph neural network (GNN). It is known that general GNNs suffer from the over-smoothing problem, specifically when the graph is dense [21, 28]. Unfortunately, bipartite graphs are innately dense, making GNNs ineffective for this type of problem.

In this context, we identified two potential network architectures for our purpose: Deep Bipartite Matching (DBM) [9] and WeaveNet [37], both designed to approximately solve NP-hard fair stable matching problems. The architectures prevent the over-smoothing problem by maintaining edge-wise features. Smoothing occurs through vertex-wise feature aggregation. Hence, forwarding features in an edge-wise manner without aggregation essentially solves the over-smoothing problem. However, this strategy has a notable shortcoming of memory consumption, making it unsuitable for large-scale matching problems like 3D point cloud registration. This study proposes to connect feature extractors with WeaveNet in a memory-efficient way and scale it for 3D point cloud registration.

3 Method

3.1 Preliminaries

Given two point clouds $\mathcal{P} \in \mathbb{R}^{N\times 3}$ and $\mathcal{Q} \in \mathbb{R}^{M\times 3}$, where N and M denote the number of points, respectively. Our goal is to identify a permutation matrix $\boldsymbol{M} \in \{0,1\}^{N\times M}$, which projects $\mathcal{P}$ onto $\mathcal{Q}$ and represents the point-wise correspondence between $\mathcal{P}$ and $\mathcal{Q}$.

Conventional methods solve this problem by training a feature extractor $\mathcal{E}$. Let $\boldsymbol{P} = \mathcal{E}(\mathcal{P}) \in \mathbb{R}^{N\times C}$ be a list of point features extracted from $\mathcal{P}$ by $\mathcal{E}$, where C is the number of channels per feature. We define $\boldsymbol{Q} = \mathcal{E}(\mathcal{Q}) \in \mathbb{R}^{M\times C}$ in the same manner. In conventional methods, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are summarized into a distance matrix $\boldsymbol{D} \in \mathbb{R}^{N\times M}$. Let $\mathcal{D} : (\boldsymbol{P}, \boldsymbol{Q}) \mapsto \boldsymbol{D}$ be a distance function, and $\mathcal{M}$ be a matching algorithm. The entire process of the conventional method is visualized in Fig. 2 and denoted by $\boldsymbol{M} = (\mathcal{M} \circ \mathcal{D} \circ \mathcal{E})(\mathcal{P}, \mathcal{Q})$.

3.2 Constructing a Network as a Single Optimization Problem

$\boldsymbol{P}$ and $\boldsymbol{Q}$ have potential uncertainties caused by noise in the input, unseen shapes and deformations, occlusions, etc. Consider a function $\mathcal{F} : \boldsymbol{x} \mapsto \boldsymbol{y}$ where the input $\boldsymbol{x}$ is a noisy observation of the true value $\boldsymbol{x}^*$ (i.e., $\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\varepsilon_x}$). Then, we can view $\mathcal{F}$'s process as $\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \mathbb{P}_{\mathcal{F}}(\boldsymbol{y}|\boldsymbol{x}^* + \boldsymbol{\varepsilon_x})$, where $\hat{\boldsymbol{y}}$ is an estimate and $\mathbb{P}_{\mathcal{F}}$ is a probabilistic model behind $\mathcal{F}$. From this perspective, we can rewrite the process $\boldsymbol{M} = (\mathcal{M} \circ \mathcal{D} \circ \mathcal{E})(\mathcal{P}, \mathcal{Q})$ as

$$\hat{\boldsymbol{M}} = \arg\max_{\boldsymbol{M}} \mathbb{P}_{\mathcal{M}}(\boldsymbol{M}|\hat{\boldsymbol{D}}),\ \hat{\boldsymbol{D}} = \arg\max_{\boldsymbol{D}} \mathbb{P}_{\mathcal{D}\circ\mathcal{E}}(\boldsymbol{D}|\mathcal{P}, \mathcal{Q}). \tag{1}$$

This equation reveals the two-stage optimization of the conventional methods.

To integrate the two-stage processes into a single optimization step, we extend $\boldsymbol{D}$ to a tensor $\mathbf{Z} \in \mathbb{R}^{N\times M\times D}$. The D-channel vectors in $\mathbf{Z}$ characterize the distribution shape of $\boldsymbol{D}$, denoted as $\phi_{\mathbf{Z}}$ (e.g., $\boldsymbol{D} = (\boldsymbol{D}^* + \boldsymbol{\varepsilon_D}) \sim \phi_{\mathbf{Z}}$). Considering this distribution, we combine the two stages via marginalization as

$$\hat{\boldsymbol{M}} = \arg\max_{\boldsymbol{M}} \int_{\boldsymbol{D}\sim\phi_{\mathbf{Z}}} \mathbb{P}_{\mathcal{M}}(\boldsymbol{M}|\boldsymbol{D})\mathbb{P}_{\mathcal{C}\circ\mathcal{E}}(\boldsymbol{D}|\mathcal{P}, \mathcal{Q}) = \arg\max_{\boldsymbol{M}} \mathbb{P}_{\mathcal{M}\circ\mathcal{C}\circ\mathcal{E}}(\boldsymbol{M}|\mathcal{P}, \mathcal{Q}), \tag{2}$$

where $\mathcal{C}$ is a function to convert $(\boldsymbol{P}, \boldsymbol{Q})$ to $\mathbf{Z}$, which we call a connector (a concrete example will be provided later). Now, $\mathcal{E}$ and $\mathcal{M}$ are combined into a single probabilistic model $\mathbb{P}_{\mathcal{M}\circ\mathcal{C}\circ\mathcal{E}}$ via the connector function $\mathcal{C}$, enabling a joint optimal solution.

The above unification theoretically improves the optimization process, however, it requires an algorithm to identify $\boldsymbol{M}$ from a tensor input $\mathbf{Z}$. As discussed in Sec. 2.2, we employ WeaveNet [37] (see **??** for the network details), a model that is capable of learning how to interpret $\mathbf{Z}$ together with the estimation of $\boldsymbol{M}$. However, its edge-wise feature forwarding poses scalability issues.

Consider implementing Eq. (2) with a trivial $\mathcal{C}$ implementation, where all information from $\mathcal{E}$ is passed to WeaveNet directly, as shown in figure Fig. 3. A

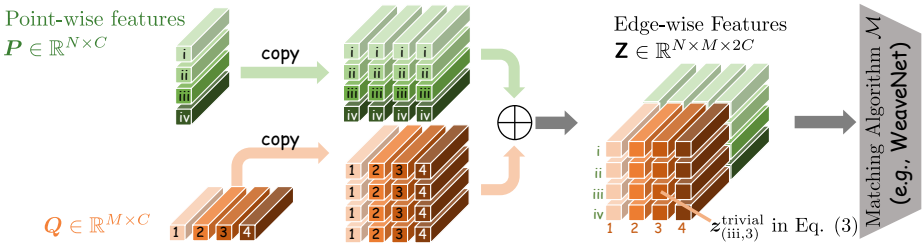


Fig. 3: A trivial implementation of $\mathcal{C}$, which passes through all the information from $\mathcal{E}$ as is. $\mathbf{Z}$ has the full size of $N \times M$ edges, which consumes massive memory. This model is intractable on standard benchmark datasets (e.g., $1,024 \times 1,024$ in 4DLoMatch) or larger practical applications.

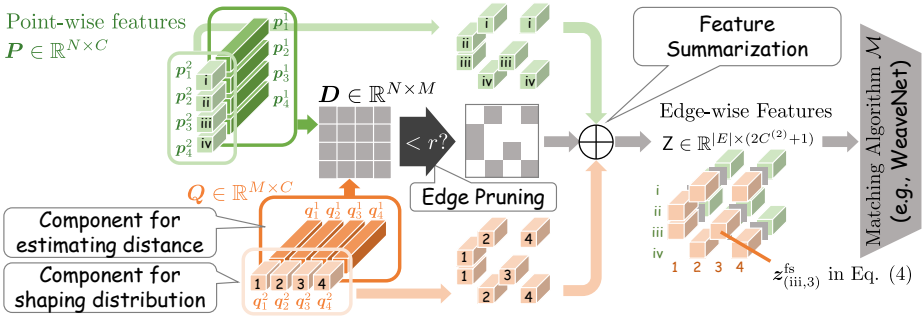


Fig. 4: The proposed implementation of $\mathcal{C}$ separates point-wise features into distance and uncertainty components. The distance component is compressed to $\boldsymbol{D}$. After pruning the edges, $\boldsymbol{D}$ and the uncertainty component are concatenated to form $\mathbf{Z}$ on a sparse bipartite graph.

formal description of this trivial implementation is given as follows. Let $\boldsymbol{p}_n \in \mathbb{R}^C$ and $\boldsymbol{q}_m \in \mathbb{R}^C$ be the features of the n-th point in $\boldsymbol{P}$ and the m-th point in $\boldsymbol{Q}$. Similarly, let $\boldsymbol{z}_{(n,m)}$ be the (n,m)-th vector in $\mathbf{Z}$. The trivial implementation of $\mathcal{C}$ directly pass the distance information as

$$\boldsymbol{z}^{\text{trivial}}_{(n,m)} = \text{cat}(\boldsymbol{p}_n, \boldsymbol{q}_m), \tag{3}$$

where cat is the concatenation operation.

Since WeaveNet (or similar models [9]) forwards edge-wise features, it performs $O(N \times M)$ linear transformations at every layer. Due to this limitation, the original study of WeaveNet [37] could solve the problems up to the size of $N \times M = 100 \times 100 = 10\text{k}$, just 1/100 of the 4D(Lo)Match dataset [22] ($N \times M = 1024 \times 1024 \simeq 1\text{M}$). 1M operations is equivalent to applying a (1×1)-kernel convolution to every pixel on a $1,000 \times 1,000$ pixel image. Thus, processing a standard-size 3D point cloud registration task with this implementation is impractical.

3.3 Details of the WeaveNet architecture

Let $(\mathbf{Z}_\ell^{\mathcal{P}}, \mathbf{Z}_\ell^{\mathcal{Q}})$ be the input to ℓ-th layer. We describe the ℓ-th feature weaving layer as a function $f_\ell : (\mathbf{Z}_\ell^{\mathcal{P}}, \mathbf{Z}_\ell^{\mathcal{Q}}) \mapsto (\mathbf{Z}_{\ell+1}^{\mathcal{P}}, \mathbf{Z}_{\ell+1}^{\mathcal{Q}})$. Hereafter, we only explain the calculation for the $\mathcal{P} \to \mathcal{Q}$ direction for simplicity. Let $\mathcal{N}(p_n)$ be a set of neighbors of node p_n. On a bipartite graph, $\mathcal{N}(p_n) = \mathcal{Q}$ and $\mathcal{N}(q_m) = \mathcal{P}$. A feature weaving layer describes the characteristic of q_m among $\mathcal{Q}$ as

$$\boldsymbol{h}_{\ell,n} = \text{max\_pooling}\{\phi_\ell^1(\boldsymbol{z}_{\ell,(n,m)}) | q_m \in \mathcal{N}(p_n)\}, \tag{4}$$

$$\boldsymbol{g}_{\ell,(n,m)} = \text{pReLU}(\text{BN}(\phi_\ell^2(\text{cat}(\boldsymbol{z}_{\ell,(n,m)}, \boldsymbol{h}_{\ell,n})))), \tag{5}$$

where max_pooling is the max pooling operation, ϕ_ℓ^1 and ϕ_ℓ^2 are linear layers inside of f_ℓ, $\boldsymbol{z}_{(n,m,\ell)}$ is (n,m)-th element in $\mathbf{Z}_\ell^{\mathcal{P}}$, pReLU is the pReLU function, and BN is the batch normalization operation.

Second, the layer mixes features obtained for each side to pass messages each other. Let $\boldsymbol{g}_{\ell,(n,m)}^{\mathcal{P}}$ and $\boldsymbol{g}_{\ell,(m,n)}^{\mathcal{Q}}$ be vectors obtained by Eq. (5) for each direction. f_ℓ concatenate them at the end of calculation, which yields

$$\boldsymbol{z}_{\ell+1,(n,m)}^{\mathcal{P}} = \text{cat}(\boldsymbol{g}_{\ell,(n,m)}^{\mathcal{P}}, \boldsymbol{g}_{\ell,(m,n)}^{\mathcal{Q}}). \tag{6}$$

Finally, the L-th output $\boldsymbol{z}_{L+1,(n,m)}$ is fed to the output layer, which first applies Eq. (4) with $\boldsymbol{z}_{L+1}$ as its input, then, calculate

$$g_{n,m} = \text{softmax}(BN(\phi_{output}(\text{cat}(\boldsymbol{z}_{L+1,(n,m)}, \boldsymbol{h}_{L+1,(n,m)})))), \tag{7}$$

where softmax is the softmax function, and $g_{n,m}$ is (n,m)-the element in $\boldsymbol{M}$.

3.4 Edge pruning and Feature Summarization

We address the memory consumption problem through edge pruning and edge-wise feature summarization (Fig. 4). Note that we extended the original WeaveNet so as to handle edge-pruned sparse bipartite graphs (see **??**).

Edge pruning is a promising approach to reduce the memory consumption in $\mathcal{M}$. This is achieved by finding nearest neighbors for each point $\boldsymbol{p}$ in $\boldsymbol{Q}$ (and for each $\boldsymbol{q}$ in $\boldsymbol{P}$). There are several options for edge selection, such as preserving k-nearest neighbors, k-reciprocal neighbors, and r-radius neighbors. Our preliminary experiments showed that we can train a model with r-radius neighbors more stably than k-nearest neighbors. Thus, we adopted r-radius neighbors, which gives a set of selected edges $\boldsymbol{E} = \{(n,m) \mid d(\boldsymbol{p}_n, \boldsymbol{q}_m) < r\}$ and now $\mathbf{Z} \in \mathbb{R}^{|E| \times D}$.

To prevent memory overflows, we limit the total number of edges $|E|$ with a constant upper-bound parameter K; we discard distant edges when a point p_n (or q_m) has more than K nearest neighbors. Assuming $N > M$ without loss of generality, this operation reduces $\mathcal{M}$'s worst-case memory consumption from $O(N \times M)$ to $O(K \times N) = O(N)$. Note that $K \times N$ typically matches the available GPU memory size since utilizing all available memory for the model is generally the best strategy. Hence, K can be set automatically without requiring

any tuning. In addition, this discarding operation occurs only when an unusually high number of points are within the r-radius, which is rare, and thus hardly affects matching results.

In addition to edge pruning, we further improve memory efficiency by reducing channels D of $\mathbf{Z}$. The trivial connector in Eq. (3) produces $D = 2C$ channels, where $C = 256$ in recent SOTA feature extractors [22, 23, 51, 54]. On the other hand, WeaveNet requires only 32 channels for solving NP-hard problems [37], indicating a large room for memory efficiency improvement.

We aim to reduce this gap with minimal information loss by updating $\mathcal{C}$, referred to as feature summarization. The standardized moment describes arbitrary distribution shapes with one expected value and multiple moment values. Inspired by this system, we update Eq. (3) as

$$\boldsymbol{z}^{\mathrm{fs}}_{(n,m)} = \mathrm{cat}(\boldsymbol{p}^2_n, d(\boldsymbol{p}^1_n, \boldsymbol{q}^1_m), \boldsymbol{q}^2_m), \tag{8}$$

where $(\boldsymbol{p}^1_n, \boldsymbol{p}^2_n)$ is a split of $\boldsymbol{p}_n$ (i.e., $\boldsymbol{p}_n = \mathrm{cat}(\boldsymbol{p}^1_n, \boldsymbol{p}^2_n)$), where $\boldsymbol{p}^1_n \in \mathbb{R}^{C^{(1)}}$ and $\boldsymbol{p}^2_n \in \mathbb{R}^{C^{(2)}}$. Similarly, $\boldsymbol{q}_m = \mathrm{cat}(\boldsymbol{q}^1_m, \boldsymbol{q}^2_m)$. Finally, d denotes a distance function. We use the cosine distance in line with SOTA methods. Now, $d(\boldsymbol{p}^1_n, \boldsymbol{q}^1_m)$ represents the expected value, while $\boldsymbol{p}^2_n$ and $\boldsymbol{q}^2_n$ indicate other distribution parameters. WeaveNet learns how to interpret these parameter automatically through training. Eq. (8) produces $\mathbf{Z}$ with the channel size of $D = 2C^{(2)} + 1$. Here, $C^{(1)}$ and $C^{(2)}$ are hyperparameters, and we restrict them to be $C^{(1)} + C^{(2)} = C$ for a fair comparison with conventional methods. In this condition, $C^{(2)} = 0$ ($C^{(1)} = C$) is equivalent to conventional methods, while $C^{(2)} = C$ ($C^{(1)} = 0$) is equivalent to the trivial implementation.

4 Experiments

We confirmed the effectiveness of our concept by integrating our method with four SOTA methods and testing it on six datasets in total. The differences among datasets are summarized in Tab. 1.

Our main experiment in 4.1 tests our method on the **4DMatch**, **4DLoMatch**, **3DMatch**, and **3DLoMatch** datasets. Here, the SOTA methods include **Lepard** [22], **LNDP** [23], and **RoITr** [51]. These methods were originally implemented using **dual softmax (DS)** or **Sinkhorn (OT)**[4].

Subsequently, we will evaluate the effect of memory reduction in 4.2. 4.3 demonstrates the robustness of our method against uncertainty through a systematic analysis. In 4.4, we explain our hyperparameter tuning. This setting was consistent across all experiments.

Finally, 4.5 presents further evaluation with the **Surreal** and **SHREC** datasets, where the SOTA method is **CorrNet3D** [54]. This experiment includes both supervised and unsupervised settings as described in the CorrNet3D paper. Note that our model was trained using the loss function of each baseline method as is throughout all experiments.

[4] Sinkhorn is also known as optimal transport, and we follow the notation in [51].

Table 1: Diversity in experimental setting.

Dataset	Base model	Shape	partial?	non-rigid?
4DMatch [22]	Lepard [22] LNDP [23] RoITr [51]	Animal		✔
4DLoMatch [22]		Animal	✔	✔
3DMatch [53]		Indoor		
3DLoMatch [15]		Indoor	✔	
Surreal [10]	CorrNet3D [54]	Human		
Surreal (train) / SHREC [6] (test)		Human		✔

Table 2: Quantitative results on 4DMatch, 4DLoMatch, 3DMatch, and 3DLoMatch. Our method consistently boosted the performance of all SOTA base models across all datasets. (*: reported in [22], **: reported in [51], repro.: reproduced by us).

Method	4DMatch		4DLoMatch		3DMatch			3DLoMatch		
	NFMR↑	IR↑	NFMR	IR	FMR↑	IR	RR↑	FMR	IR	RR
D3Feat [2]*	55.5	54.7	27.4	21.5	95.8	39.0	85.8	69.3	13.2	40.2
Predator [16]*	56.4	60.4	32.1	27.5	96.5	57.1	90.6	76.3	28.3	62.4
GeoTrans [31]**	83.2	82.2	65.4	63.6	97.9	76.0	91.8	88.8	46.2	74.2
Lepard-DS [22]*	83.6	82.7	66.9	55.7	98.3	55.5	93.5	84.5	26.0	69.0
RoITr-OT [51]**	83.0	84.4	69.4	<u>67.6</u>	97.9	<u>83.0</u>	91.8	89.5	<u>55.1</u>	74.8
Lepard-DS (repro.)	83.7	82.7	66.9	55.7	98.3	55.5	93.5	84.5	26.0	69.0
Lepard-WN (ours)	86.7	86.1	72.4	62.5	98.4	64.5	<u>95.7</u>	89.6	30.4	74.9
Δ	+3.0	+3.4	+5.5	+6.8	+0.1	+9.0	+2.2	+5.1	+4.4	+5.9
LNDP-DS (repro.)	85.4	84.5	67.6	57.6	98.1	56.5	92.4	83.1	27.4	71.1
LNDP-WN (ours)	**88.7**	**87.9**	<u>73.4</u>	62.8	<u>98.6</u>	65.6	94.1	**91.3**	33.3	<u>76.2</u>
Δ	+3.3	+3.4	+5.8	+5.2	+0.5	+9.1	+1.7	+8.2	+5.9	+5.1
RoITr-OT (repro.)	81.3	81.2	67.2	64.8	98.5	80.3	91.0	89.6	54.3	74.2
RoITr-WN (ours)	<u>87.2</u>	<u>87.3</u>	**75.3**	**73.3**	**98.9**	**86.8**	**96.2**	<u>90.0</u>	**64.4**	**82.4**
Δ	+5.9	+6.1	+8.1	+8.5	+0.4	+6.5	+5.2	+0.4	+10.1	+8.2

4.1 Rigid/Non-rigid and Whole/Partial conditions

Dataset. We conducted experiments on the registration tasks of non-rigid animal point clouds and rigid indoor point clouds following [22,51]. We began with the non-rigid animal point clouds in the 4DMatch and 4DLoMatch datasets [22]. These datasets were obtained from 1,761 animation sequences. They are split into 1,232/176/353 for train/valid/test sets. The test sets are further divided into 4DMatch and 4DLoMatch datasets based on overlap ratios greater than or less than 45%, respectively.

Second, we conducted experiments on indoor point clouds using the 3DMatch and 3DLoMatch datasets [15,53]. 3DMatch contains scan pairs with overlap ratios greater than 30%, while 3DLoMatch contains scan pairs with ratios between 10% and 30%. The 62 indoor scenes are divided into 46/8/8 as train/valid/test sets.

Table 3: Average memory consumption per sample with RoITr on 4DMatch: ES and FS stand for Edge Selection and Feature Summarization, respectively.

Method	EP	FS	train	eval.	NFMR(↑)	IR(↑)
RoITr-WN			135.2 GiB	64.3 GiB	**89.9**	83.3
		✔	124.5 GiB	57.6 GiB	<u>88.6</u>	84.1
	✔		12.3 GiB	6.4 GiB	87.4	<u>85.2</u>
	✔	✔	8.1 GiB	4.5 GiB	87.2	**87.3**
RoITr-OT	-	-	4.6 GiB	2.1 GiB	81.3	81.2

Evaluation metrics. We used the inlier ratio (IR) (also known as accuracy [23]), and non-rigid feature matching recall (NFMR) as the evaluation metrics for the 4DMatch and 4DLoMatch datasets following [22, 23, 51]. The IR, feature matching recall (FMR), and rigid registration recall (RR) are used as the evaluation metrics for the 3DMatch and 3DLoMatch datasets following [22]. The details are described in Appendix. **??**

Implementation details. We used the SGD optimizer with a learning rate of 0.015, a batch size of 8, and 15 epochs. These settings are consistent among the baseline models [22,23,51] and our methods. The loss converged in all models under these settings. Our method adhered to the baselines' original paper for all other settings. Our extension incorporates three additional hyperparameters: $r = 0.5$, $C^{(2)} = 16$, and $L = 10$, where L is the number of layers in WeaveNet. The models were trained with four Tesla V100 GPUs.

Results The experimental results on 4DMatch, 4DLoMatch, 3DMatch, and 3DLoMatch are shown in Tab. 2. In the table, DS, OT, and WN suffixes indicate the matching module used with the baseline model name (e.g., **Lepard-DS** is Lepard's original implementation, while **Lepard-WN** uses our proposed matching module with the connector function). RoITr-WN outperformed other methods on the three datasets, whereas LNDP-WN showed superior performance on the 4DMatch dataset compared to RoITr-WN. Our method did not negatively impact the convergence speed as these results were obtained with an identical schedule across all methods.

The Δ in the table indicates performance improvements from our reproduced results. Our method improved the performance of all baseline models under all tested conditions. This positive effect remained consistent, even when compared with scores reported in the original papers. These results experimentally support our theoretical improvement by Eq. (2).

Fig. 1 shows qualitative results from our experiment. Samples were chosen randomly to prevent cherry-picking. Exact matches are colored in green, while others are red. Our method consistently increased exact matches across these four samples.

4.2 Analysis on Memory Consumption

We improved the memory efficiency to scale WeaveNet [37] for 3D point cloud registration. Tab. 3 shows average memory consumption measured during training and inference on 4DMatch with RoITr and their performance. WeaveNet with and without feature summarization are implemented with $\boldsymbol{z}^{trivial}_{(n,m)}$ in **??** and $\boldsymbol{z}^{\mathrm{fs}}_{(n,m)}$ in Eq. (8), respectively. Due to memory constraint, models without edge pruning were trained with a batch size of 1 on eight Tesla A100 GPUs, while the others were trained with a batch size of 8 on four Tesla V100 GPUs (as in the previous experiment).

From the result, we can see that the edge pruning significantly decreases the memory consumption (par sample) from 135.2 GiB to 12.3 GiB (-90.9%) at training and from 64.3GiB to 6.4 GiB (-90.0%) at inference. Feature summarization further reduces memory from 12.3 GiB to 8.1 GiB (-34.1%) at training and from 6.4 GiB to 4.5 GiB (-29.7%) at inference, maintaining NFMR and improving IR, which we deem comparable. Overall, the memory consumption became about 1/30 while preserving the performance.

We observed similar tendencies with other baseline models and datasets (see Appendix **??**). This experiment also revealed that our method consumes twice as much memory as the original baseline models, which is a limitation.

4.3 Systematic Analysis on Robustness towards Feature Uncertainty

Aiming to confirm the robustness of our method against uncertainty in the upstream process of distance estimation, we conducted a systematic study using the 4DLoMatch and 3DLoMatch datasets. Identical point clouds were used for both the source side input $\mathcal{P}$ and target side input $\mathcal{Q}$ in this experiment. Gaussian noise $\mathcal{N}(0, \sigma)$ was added to the source side input $\mathcal{P}$ to simulate upstream process uncertainties. The parameter σ was varied from 0.0 to 1.0 in steps of 0.2. The models were retrained for each setting.

The results of our experiments are illustrated in Figs. 5 and 6. We observed a clear performance drop for larger σ in both figures, as intended. Among them, methods using our matching module (WN) consistently outperformed their original implementations (DS or OT). The difference was particularly noticeable at $\sigma = 0.6$ with Lepard on 4DLoMatch (Fig. 5). The original Lepard model had a sudden performance drop. However, our method diminished such negative effects caused by this artificial ambiguity. We also confirmed that the positive effect is more pronounced under the non-rigid partial condition of 4DLoMatch. These results support our hypothesis that uncertainty-robust matching was achieved by our method.

4.4 Hyperparameter Validation

Hyperparameter insensitivity is an essential factor for method's utility. A parameter-sensitive method may fail on unseen datasets. We conducted extensive ablation studies to validate the method's hyperparameter insensitivity.

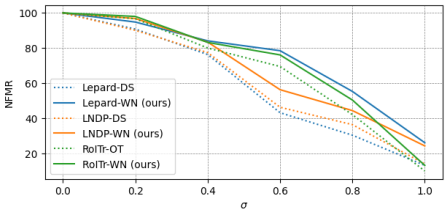


Fig. 5: Noise robustness test on 4DLoMatch. Our method consistently outperforms its base model.

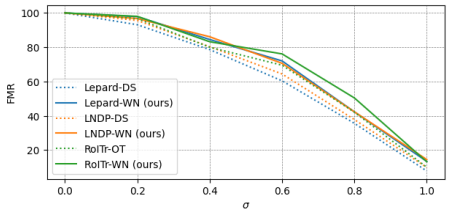


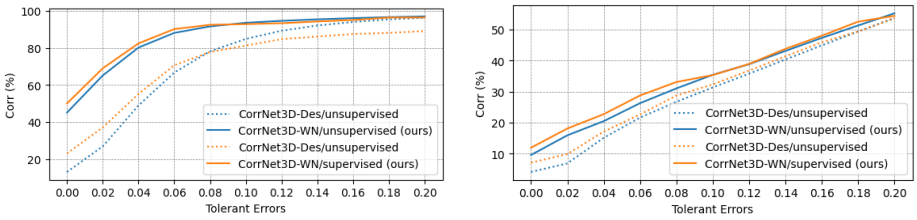
Fig. 6: Noise robustness test on 3DLoMatch. We can see the same tendency with the study on 4DLoMatch in Fig. 5.

Table 4: Study on the impact of hyperparameters. We used the best hyperparameters listed in this table for all other experiments mentioned in this paper, irrespective of the feature extractor and dataset used.

Method		4DLoMatch NFMR(↑)	4DLoMatch IR(↑)	3DLoMatch FMR(↑)	3DLoMatch IR(↑)	3DLoMatch RR(↑)
RoITr-OT		67.2	64.8	89.6	54.3	74.2
RoITr-WN ($L = 10, C^{(2)} = 16$)	$r = 0.1$	69.6	67.1	89.6	60.2	76.0
	$r = 0.5$	**75.3**	**73.3**	**90.0**	**64.4**	**82.4**
	$r = 1.0$	70.1	70.3	89.9	61.3	77.7
RoITr-WN ($r = 0.5, C^{(2)} = 16$)	$L = 6$	68.9	67.9	89.9	58.9	77.8
	$L = 8$	73.4	72.1	90.0	62.1	81.3
	$L = 10$	75.3	73.3	**90.0**	**64.4**	82.4
	$L = 12$	**75.4**	**75.4**	89.2	63.4	**84.3**
RoITr-WN ($r = 0.5, L = 10$)	$C^{(2)} = 4$	69.1	70.5	89.5	55.4	79.5
	$C^{(2)} = 16$	**75.3**	**73.3**	**90.0**	**64.4**	**82.4**
	$C^{(2)} = 64$	70.4	69.5	89.9	60.3	81.3

Our method has three hyperparameters: r, L, and $C^{(2)}$. The values described in Sec. 4.1 were used for them as defaults, adjusting each parameter individually. The results obtained with RoITr on the 4DLoMatch and 3DLoMatch datasets are presented in Tab. 4. In this setup, the default values always outperformed alternatives across all datasets. The same tendency was observed for other methods and datasets (See Appendix **??**).

We discuss the results in detail. First, r is the radius of the nearest neighbor, used as the threshold of edge pruning. A small r might overly prune edges, while a large r might preserve noisy edges. This is observed in the lower performance at $r = 0.1$ and 1.0 compared to $r = 0.5$. Second, L is the number of layers. Generally, deeper networks provide more accurate estimations. Our results confirm this expectation with our method. We did not explore L over 12 as it makes experiments without edge pruning in Sec. 4.2 impossible. Nevertheless, the performance improvements from $L = 6$ to $L = 10$ follow a log-scale trend, and we observe a negative effect when increasing layers from $L = 10$ to 12. Therefore, we



(a) Test on the rigid dataset (surreal) **(b)** Test on the non-rigid dataset (SHREC)

Fig. 7: Accuracy transition of CorrNet3D w/ and w/o the proposed modification on the base model, along with the error tolerances defined in **??**.

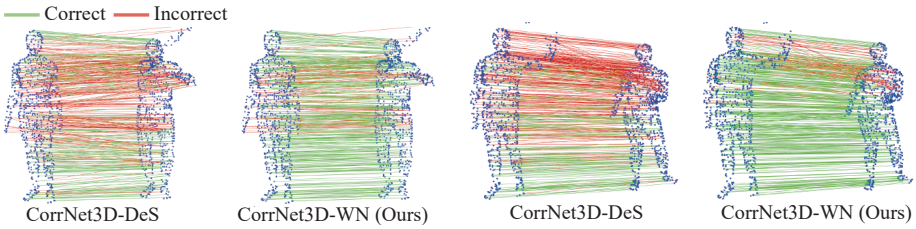


Fig. 8: Qualitative results on SHREC. Samples were chosen randomly to ensure unbiased selection. Green lines indicate exact matches with zero error tolerance, while red lines indicate non-zero errors.

have decided $L = 10$ as the most efficient parameter, balancing performance and memory consumption. Third, $C^{(2)}$ controls the split ratio of point-wise features into components for the distance matrix calculation and uncertainty representation. A $C^{(2)}$ that is too small fails to represent uncertainty, while a $C^{(2)}$ that is too large results in a shortage of dimensions for distance calculation. The result explains this well, as $C^{(2)} = 16$ performs better than $C^{(2)} = 4$ and 64 among the 256 channels.

4.5 Evaluation with CorrNet3D on Human Shape data

Dataset. We conducted additional experiments using human shape data as in [54]. We applied our method to CorrNet3D [54], which has supervised and unsupervised training setups. We tested our method under both conditions.

The Surreal dataset [41] is used for the rigid setting, which contains 230K point cloud samples for training and 100 samples for testing. The 230K point clouds were randomly paired into 115K training samples, and 100 test pairs were created by rotating and translating the test samples. For the non-rigid setting, the Surreal dataset and the SHREC dataset [20] are used, containing 230K for training and 860 samples for testing. The training samples are the same as in the rigid setting, while the 860 test samples are randomly combined into 430 test pairs. Each point cloud contains 1,024 points across the datasets.

Evaluation metrics. Following [54], we used the corresponding percentage (Corr) under controlled error tolerances as the evaluation metric. The details are described in Appendix. **??**.

Implementation details. In this experiment, we replaced the DeSmooth module (**DeS**) proposed for the CorrNet3D model with WeaveNet (WN). The model was optimized according to the loss function in [54]. Following [54], we used the Adam optimizer with a learning rate of 1e-4, a batch size of 10, and trained for 100 epochs.

Results on the rigid condition. The experimental results on the rigid dataset are shown in Fig. 7 (a). The rigid lines illustrate the Corr metric scores (↑) of CorrNet3D-WN in supervised (orange) and unsupervised (blue) setups. The dotted lines illustrate the reproduced scores of the original CorrNet3D. The results show that CorrNet3D-WN consistently outperforms CorrNet3D-DeS in both setups, further supporting our method's versatility.

A closer look at the figure indicates that the performance gain is substantial for small tolerance error regions. It reveals that our method effectively distinguishes and identifies the corresponding point from similar points.

Results on the non-rigid condition. Fig. 7 (b) shows the experimental results on the non-rigid dataset, and Fig. 8 shows the samples of the experimental results. The CorrNet3D-WN consistently outperforms the CorrNet3D-DeS in both setups again. The absolute Corr scores are lower than those in Fig. 7 (a) due to the gap between training and test data. The original CorrNet3D paper suggests training a model with rigid Surreal and testing it on non-rigid SHREC. Even in this challenging scenario, our method shows improved performances.

5 Limitation

One limitation of our approach is sensitivity to the hyperparameters. Although our method works well across datasets with an appropriate hyperparameter setup, specific hyperparameter settings can lead to negative effects, similar to conventional methods [17, 24, 32]. Another limitation is memory usage. Our experiments revealed that our modification, including edge pruning and feature summarization, scaled the method to a consumer GPU level. However, it still doubles memory consumption. We must further improve the efficiency to apply our method for mobile edges.

6 Conclusion

This paper proposed unifying a feature extractor and a matching module into a single probabilistic model through a non-deterministic connection and a learning-based matching module. Our method demonstrated consistent and significant improvement across four SOTA architectures on six datasets. Our systematic analysis with controlled noise also supported that our implementation performed as intended. These results, achieved using a single hyper-parameter setting, demonstrate the method's reliability.

References

1. Aoki, Y., Goforth, H., Srivatsan, R.A., Lucey, S.: PointNetLK: Robust & efficient point cloud registration using pointnet. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7163–7172 (2019)
2. Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., Tai, C.L.: D3feat: Joint learning of dense detection and description of 3d local features. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 6359–6367 (2020)
3. Clay, R., Steenkiste, P.: Distributing a chemical process optimization application over a gigabit network. In: Supercomputing '95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing. pp. 45–45 (1995). https://doi.org/10.1145/224170.224310
4. Cortinhal, T., Tzelepis, G., Erdal Aksoy, E.: SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds for autonomous driving. In: International Symposium on Advances in Visual Computing. pp. 207–222. Springer (2020)
5. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 26 (2013)
6. Donati, N., Sharma, A., Ovsjanikov, M.: Deep geometric functional maps: Robust feature learning for shape correspondence. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8592–8601 (2020)
7. Du, G., Wang, K., Lian, S., Zhao, K.: Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. Artificial Intelligence Review **54**(3), 1677–1734 (2021)
8. Elbaz, G., Avraham, T., Fischer, A.: 3D point cloud registration for localization using a deep neural network auto-encoder. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4631–4640 (2017)
9. Gibbons, D., Lim, C.C., Shi, P.: Deep learning for bipartite assignment problems. In: IEEE International Conference on Systems, Man and Cybernetics. pp. 2318–2325 (2019)
10. Groueix, T., Fisher, M., Kim, V.G., Russell, B., Aubry, M.: 3D-CODED: 3D correspondences by deep deformation. In: European Conference on Computer Vision. pp. 230–246 (2018)
11. Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., Kwok, N.M.: A comprehensive performance evaluation of 3D local feature descriptors. International Journal of Computer Vision **116**(1), 66–89 (2016)
12. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3D point clouds: A survey. IEEE Transaction on Pattern Analysis and Machine Intelligence (2020)
13. Hana, X.F., Jin, J.S., Xie, J., Wang, M.J., Jiang, W.: A comprehensive review of 3D point cloud descriptors. arXiv preprint arXiv:1802.02297 **2** (2018)
14. Hua, B.S., Tran, M.K., Yeung, S.K.: Pointwise convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 984–993 (2018)
15. Huang, S., Gojcic, Z., Usvyatsov, M., Wieser, A., Schindler, K.: Predator: Registration of 3D point clouds with low overlap. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4267–4276 (June 2021)

16. Huang, X., Mei, G., Zhang, J., Abbas, R.: A comprehensive survey on point cloud registration. arXiv preprint arXiv:2103.02690 (2021)
17. Jiang, H., Yan, F., Cai, J., Zheng, J., Xiao, J.: End-to-end 3D point cloud instance segmentation without detection. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12796–12805 (2020)
18. Johnson, A.E., Hebert, M.: Using spin images for efficient object recognition in cluttered 3D scenes. IEEE Transaction on Pattern Analysis and Machine Intelligence **21**(5), 433–449 (1999)
19. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2**, 83–97 (1955)
20. Li, B., Lu, Y., Li, C., Godil, A., Schreck, T., Aono, M., Burtscher, M., Chen, Q., Chowdhury, N.K., Fang, B., et al.: A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. Computer Vision and Image Understanding **131**, 1–27 (2015)
21. Li, Q., Han, Z., Wu, X.M.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI Conference on Artificial Intelligence. pp. 3538–3545 (2018)
22. Li, Y., Harada, T.: Lepard: Learning partial point cloud matching in rigid and deformable scenes. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5554–5564 (2022)
23. Li, Y., Harada, T.: Non-rigid point cloud registration with neural deformation pyramid. Advances in Neural Information Processing Systems **35**, 27757–27768 (2022)
24. Liu, H., Guo, Y., Ma, Y., Lei, Y., Wen, G.: Semantic context encoding for accurate 3D point cloud segmentation. IEEE Transactions on Multimedia **23**, 2045–2055 (2021)
25. Liu, W., Sun, J., Li, W., Hu, T., Wang, P.: Deep learning on point clouds and its application: A survey. Sensors **19**(19), 4188 (2019)
26. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: International Conference on Learning Representation (2022), https://openreview.net/forum?id=3Pbra-_u76D
27. Maturana, D., Scherer, S.: VoxNet: A 3D convolutional neural network for real-time object recognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 922–928 (2015)
28. Oono, K., Suzuki, T.: Graph neural networks exponentially lose expressive power for node classification. In: International Conference on Learning Representation (2020), https://openreview.net/forum?id=S1ldO2EFPr
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (2017)
30. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (2017)
31. Qin, Z., Yu, H., Wang, C., Guo, Y., Peng, Y., Xu, K.: Geometric transformer for fast and robust point cloud registration. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11143–11152 (June 2022)
32. Ravanbakhsh, S., Schneider, J., Poczos, B.: Deep learning with sets and point clouds. In: International Conference on Learning Representation Workshop (2017)

33. Rocco, I., Cimpoi, M., Arandjelović, R., Torii, A., Pajdla, T., Sivic, J.: Neighbourhood consensus networks. In: Advances in Neural Information Processing Systems. vol. 31 (2018)
34. Sarlin, P.E., DeTone, D., Malisiewicz, T., Rabinovich, A.: SuperGlue: Learning feature matching with graph neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (June 2020)
35. Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R.A., Lucey, S., Choset, H.: PCRNet: Point cloud registration network using pointnet encoding. arXiv preprint arXiv:1908.07906 (2019)
36. Senin, N., Catalucci, S., Moretti, M., Leach, R.: Statistical point cloud model to investigate measurement uncertainty in coordinate metrology. Precision Engineering **70**, 44–62 (2021)
37. Sone, S., Ma, J., Hashimoto, A., Chiba, N., Ushiku, Y.: Weavenet for approximating two-sided matching problems. arXiv preprint arXiv:2310.12515 (2023)
38. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: LoFTR: Detector-free local feature matching with transformers. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8922–8931 (June 2021)
39. Sun, Y.T., Fu, Q.C., Jiang, Y.R., Liu, Z., Lai, Y.K., Fu, H., Gao, L.: Human motion transfer with 3D constraints and detail enhancement. IEEE Transaction on Pattern Analysis and Machine Intelligence **45**(4), 4682–4693 (2023)
40. Tombari, F., Salti, S., Di Stefano, L.: Unique signatures of histograms for local surface description. In: European Conference on Computer Vision. pp. 356–369. Springer (2010)
41. Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M.J., Laptev, I., Schmid, C.: Learning from synthetic humans. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
42. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2589–2597 (2018)
43. Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3523–3532 (2019)
44. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. ACM Transactions on Graphics **38**(5), 1–12 (2019)
45. Xie, S., Liu, S., Chen, Z., Tu, Z.: Attentional shapecontextnet for point cloud recognition. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4606–4615 (2018)
46. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: SpiderCNN: Deep learning on point sets with parameterized convolutional filters. In: European Conference on Computer Vision. pp. 87–102 (2018)
47. Yang, J., Zhang, Q., Xian, K., Xiao, Y., Cao, Z.: Rotational contour signatures for both real-valued and binary feature representations of 3D local shape. Computer Vision and Image Understanding **160**, 133–147 (2017)
48. Yang, J., Zhang, Q., Xiao, Y., Cao, Z.: TOLDI: An effective and robust approach for 3D local shape description. Pattern Recognition **65**, 175–187 (2017)
49. Yew, Z.J., Lee, G.H.: RPM-Net: Robust point matching using learned features. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11824–11833 (2020)

50. Yu, H., Li, F., Saleh, M., Busam, B., Ilic, S.: CoFiNet: Reliable coarse-to-fine correspondences for robust pointcloud registration. Advances in Neural Information Processing Systems **34** (2021)
51. Yu, H., Qin, Z., Hou, J., Saleh, M., Li, D., Busam, B., Ilic, S.: Rotation-invariant transformer for point cloud matching. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5384–5393 (2023)
52. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Advances in Neural Information Processing Systems. vol. 30 (2017)
53. Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3DMatch: Learning local geometric descriptors from rgb-d reconstructions. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1802–1811 (2017)
54. Zeng, Y., Qian, Y., Zhu, Z., Hou, J., Yuan, H., He, Y.: CorrNet3D: Unsupervised end-to-end learning of dense correspondence for 3D point clouds. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6052–6061 (2021)
55. Zhang, Z., Dai, Y., Sun, J.: Deep learning based point cloud registration: an overview. Virtual Reality & Intelligent Hardware **2**(3), 222–246 (2020)
56. Zhang, Z., Hua, B.S., Yeung, S.K.: Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: International Conference on Computer Vision. pp. 1607–1616 (2019)
57. Zheng, Y., Xu, X., Zhou, J., Lu, J.: PointRas: Uncertainty-aware multi-resolution learning for point cloud segmentation. IEEE Transactions on Image Processing **31**, 6002–6016 (2022)
58. Zhong, Y.: Intrinsic shape signatures: A shape descriptor for 3D object recognition. In: International Conference on Computer Vision Workshop. pp. 689–696. IEEE (2009)