Searching Dense Point Correspondences via Permutation Matrix Learning

Zhiyuan Zhang\textsuperscript{1}, Jiadai Sun\textsuperscript{1}, Yuchao Dai\textsuperscript{1,2}, Bin Fan\textsuperscript{1}, and Qi Liu\textsuperscript{1}

Abstract—Although 3D point cloud data has received widespread attentions as a general form of 3D signal expression, applying point clouds to the task of dense correspondence estimation between 3D shapes has not been investigated widely. Furthermore, even in the few existing 3D point cloud-based methods, an important and widely acknowledged principle, i.e., one-to-one matching, is usually ignored. In response, this paper presents a novel end-to-end learning-based method to estimate the dense correspondence of 3D point clouds, in which the problem of point matching is formulated as a zero-one assignment problem to achieve a permutation matching matrix to implement the one-to-one principle fundamentally. Note that the classical solutions of this assignment problem are always non-differentiable, which is fatal for deep learning frameworks. Thus we design a special matching module, which solves a doubly stochastic matrix at first and then projects this obtained approximate solution to the desired permutation matrix. Moreover, to guarantee end-to-end learning and the accuracy of the calculated loss, we calculate the loss from the learned permutation matrix but propagate the gradient to the doubly stochastic matrix directly which bypasses the permutation matrix during the backward propagation. Our method can be applied to both non-rigid and rigid 3D point cloud data and extensive experiments show that our method achieves state-of-the-art performance for dense correspondence learning.

Index Terms—Dense correspondence estimation, One-to-one matching, Point cloud, End-to-end learning.

I. INTRODUCTION

As a new form of 3D signal expression, point cloud has received widespread attention due to the talent for representing 3D shapes efficiently and being obtained directly and easily by scanning. And it has been adopted in many applications, such as segmentation [1], [2], 3D reconstruction [3], [4], compression [5]–[7], autonomous driving [8]. Particularly, although dense correspondence estimation is a fundamental task in computer vision, which has been investigated widely on other signal formats (e.g., 2D image [9], [10]), it remains a challenging problem for point cloud data due to the inherent characteristics of the point cloud (e.g., irregularity, orderless), the serious deformation of object shape, and so on. Note that most existing methods of 3D shape correspondence estimation pay more attention to the 3D mesh input [11]. And although many 3D mesh-based methods have been encouraged [12]–[16], these methods cannot be extended to 3D point cloud input directly since there is no topology information existing in 3D point cloud comparing with the 3D mesh.

To solve the correspondence searching problem for the 3D point clouds directly, scene flow estimation methods e.g., FlowNet3D [17], are usually borrowed by selecting the nearest point as the corresponding point. However, an important and widely acknowledged principle, i.e., one-to-one matching, is ignored. This will result in a large number of one-to-many matches and unsatisfactory performance. CorrNet3D [18] is a representative method advocating solving a permutation matching matrix to ensure the one-to-one matching. However, this intention is not realized because the learned matching matrix is still a soft probability matrix and the final corresponding point is determined by selecting the point with the largest matching probability where the one-to-many matching is still not excluded. Besides, as a special task, learning point correspondence between rigid point clouds has been broadly investigated as a crucial intermediate step in the point cloud registration field [19]. However, most of them [20]–[23] advocate building some reliable correspondences, which is reasonable for registration but do not make sense for dense correspondence estimation task. Some virtual point-based methods [24], [25] advocate estimating dense correspondence, however, the learned correspondence is not accurate since the constructed virtual points degenerate seriously. In summary, as far as we know, the one-to-one matching principle has not been implemented well in the existing 3D point cloud-based dense correspondence learning methods.

In this paper, we focus on the dense correspondence estimation problem of 3D shapes in the form of point clouds and achieve the real one-to-one matching learning in an end-to-end manner for the first time. To this end, we formulate the point matching as a zero-one assignment problem to solve a special binary matrix, permutation matrix (PM) \textsuperscript{1} to implement one-to-one matching principle fundamentally. Although the binary matrix has been investigated widely in [26], [27], the desired PM further constrains the sum of each row and column. Note that the classical solutions of the assignment problem are always not differentiable [28], which is fatal for deep learning frameworks. In response, a special matching module is designed. This module firstly solves a doubly stochastic matrix (DSM) as the approximate solution and then projects the obtained DSM to the desired PM. Moreover, to guarantee the end-to-end learning and the accuracy of the calculated loss, we advocate calculating the loss from the learned PM but propagating the gradient to the DSM directly during the backward

\textsuperscript{1}The formal definitions of PM, DSM are given in the following section.
**C. Permutation Matrix Learning**

As mentioned above, we devote to learning a permutation matrix to implement one-to-one principle fundamentally. To this end, we design a dedicated matching module as follows to guarantee the end-to-end learning.

**Initial similarity matrix solving.** In our matching module, we firstly compute an initial similarity matrix (SM). Based on the obtained deep features $\Phi_X$ and $\Phi_Y$, the SM $S$ is calculated by using the scale dot product attention metric [30], i.e., $S = \Phi_X \Phi_Y^T / \sqrt{d}$, where the entry $s_{ij}$ represents the similarity between point $x_i$ in $X$ and point $y_j$ in $Y$.

**Permutation matrix learning.** Note that the SM $S$ is in continuous space while the desired PM $M$ is in discrete space with the entry of either 0 or 1, solving PM can be defined as a zero-one assignment problem whose existing solutions are always non-differentiable. Therefore, how to optimize $S$ to $M$ in a deep learning pipeline is the main challenge. To solve this problem, we design an ingenious matching method as presented in Alg. 1. And a DSM $P$ is solved as an intermediate variable between $S$ and $M$ as shown in Fig. 1, where $P = [p_{ij}] \in [0, 1]^{N \times N}$, $\sum_i p_{ij} = 1$, $\forall j$, $\sum_j p_{ij} = 1$, $\forall i$. Firstly, this two-stage method applies the Gumbel-Sinkhorn algorithm [31] to solve $P$ as an approximate solution of $M$. Compared to the original Sinkhorn method, the Gumbel-Sinkhorn is more robust by adding a standard i.i.d. Gumbel noise matrix $\epsilon$ to the input $S$. Then, $P$ is projected to the final permutation matching matrix $M$ by formulating a zero-one assignment problem, in which objective function is defined by considering the DSM $P$ as the profit matrix, i.e.,

$$M^* = \arg \max_{M \in \mathcal{M}_N} \min \{M, P > F\},$$

where $\mathcal{M}_N$ denotes the set of permutation matrices and $< M, P >_F = \text{trace}(M^T P)$ denotes the (Frobenius) inner product of matrices. There are many classical solutions to this assignment problem. In our implementation, the representative Hungarian algorithm [28] is applied.

**Algorithm 1: Permutation Matrix Learning.**

**Input:** Initial similarity matrix $S$, Gumbel noise matrix $\epsilon$, Hyper-parameter $\mu$, iteration number $\text{Iter}$.  
**Output:** Optimal Permutation Matrix $M^*$.

- $S = ((S + \epsilon) / \mu)$; // add Gumbel noise to similarity matrix
- $S = \exp(S)$; // $\exp(S)$
- while $\text{Iter} < \text{Iter}$ do
  - $S = T_r(S)$; // $T_r$ indicates row-wise normalization
  - $S = T_c(S)$; // $T_c$ indicates column-wise normalization
- $P = S$; // $P$ is the DSM returned by final $S$.
- $M^* = h(P)$. // $h()$ indicates the Hungarian algorithm.

**D. Loss Function**

Here, we supervise the learned matching matrix directly. Specifically, the loss function is defined as

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} (s_{ij}^{\text{pred}} - s_{ij}^{\text{gt}}),$$

where the superscript “pred” and “gt” indicate the prediction and the ground truth, respectively.
Fig. 2: The illustration of forward and backward propagation in our two-stage matching module (Left) and one-stage matching module (Right). Our method not only guarantees the end-to-end learning but also avoids the local supervision risk.

E. Remarks

Here, we stress the ingenuity of the designed PM learning module in the end-to-end learning pipeline. As mentioned above, this zero-one assignment problem is fatal for the deep learning pipeline due to the non-differentiability. As a response, we propose a ingenious operation as shown in Fig. 2 Left. Specifically, during the forward propagation, the loss is calculated based on the learned PM M. However, during the backward propagation, the gradient is propagated to the learned DSM P directly bypassing the PM solving procedure. This operation guarantees the accuracy of the calculated loss and the end-to-end training synchronously.

Then, a natural question is that, since the assignment procedure is bypassed in backward propagation, why not use the similarity matrix as the profit matrix to solve the permutation matrix but use DSM as the profit matrix. Although this one-stage strategy is more direct as shown in Fig. 2 Right, it exists an obvious local supervision risk. Specifically, if the gradient is propagated to the SM directly, the correlation of entries, which should have been considered in the assignment algorithm, is ignored in backward propagation, i.e. the loss calculated from \( m_{ij} \) can only supervise \( s_{ij} \). Note that, from Eq. (2), when \( m_{ij}^{gt} = 1, m_{ij}^{pred} \) is enforced to 1. When \( m_{ij}^{gt} = 0, m_{ij}^{pred} \) is trained without supervision. Thus many entries \( (N(N-1)) \) will be trained without supervision in this one-stage method, and the training will not converge. Our solution avoids this local supervision risk effectively by propagating the gradient to the intermediate DSM, where the correlation of all entries is considered again in Gumbel-Sinkhorn. Through our method, the similarity of the correct correspondences will be boosted, meanwhile the similarity of the wrong correspondences will be effectively suppressed during the backward propagation.

III. Experimental Results

We conduct extensive experiments on both non-rigid and rigid point cloud. In our application, in the feature extractor, a 4-layer graph neural network is applied where the parameter of K-nearest neighbor is set to 24, the embedding dimension \( d = 512 \), iteration number Iter = 5, hyper-parameter \( \mu = 0.5 \). As for the training, we at first train the feature extractor with an incomplete pipeline, i.e. solving DSM only without assignment for 100 epochs. Then, we insert the Hungarian algorithm, and retrain the entire network for 100 epochs. The initial learning rate in the Adam optimizer is set to 0.001.

Fig. 3: Comparison of the TABLE I: Comparison of per-matching precision for non-point-average geodesic distance rigid 3D point clouds on results (×100). Four dataset set-Surreal\SHREC. And \( K = \) tings are evaluated, which are 0 means the threshold is 0, indicated by \textbf{train set test set}. Datasets. For non-rigid shape correspondence, following [18], we adopted Surreal [32] and SHREC [14] as the training and test sets respectively. Besides, the re-meshed versions of FAUST [33] and SCAPE [34] are also used following [35]. For all mesh data, our method randomly picked vertices as the point cloud. For rigid shape correspondence, ModelNet40 [36] is used following the registration methods [24], [25].

Metrics. We use the matching precision \( (\%) \) metric here to evaluate the dense correspondence estimation performance. Theoretically, it presents the correct correspondence percentage by \( \frac{1}{N} ||M^{pred} \odot M^{gt}||_1 \), where \( \odot \) is the Hadamard product, \( M^{pred} \), \( M^{gt} \) represent the predicted and ground truth matching matrices, respectively. In actual application, this metric is often relaxed by confirming the correspondence as a correct result when the distance between the predicted corresponding point and the real corresponding point is less than a threshold. Here, we propose a self-adaptive threshold as \( \tau_l = \frac{1}{N} \sum_{j=1}^{N} l_{ij}, x_j \in \text{KNN}(x_i) \), \( l_{ij} = ||x_i - x_j|| \), \( x_j \) is the Euclidean distance between the points \( x_i \) and \( x_j \), \( \text{KNN}(\cdot) \) represents the K-nearest neighbor. In other words, for \( j \)-th point \( x_i \), \( \tau_l \) is computed as the mean distance of \( K \) nearest neighbor points. Meanwhile, for a fair comparison, the metric of per-point-average geodesic distance [14] is also used for the methods with 3D mesh input.

Comparison methods: For non-rigid shape correspondence estimation, we first compare with point cloud-based methods including FlowNet3D [17], CorrNet3D [18], in Unsupervised and Supervised training, noted as U-F3D, S-F3D, U-CorrNet3D and S-CorrNet3D respectively. Then, 3D mesh-based methods are also compared, including traditional methods ZoomOut [37], BCICP [38], and learning-based methods 3D-CODED [32], FMNet [12], U-FMNet [13] with and without post-processing (PMF [39]), SURFMNet [40] with and without ICP, DeepGFM [14] with and without ZoomOut (ZO).

For rigid input, ICP [41], DCP [24], PRNet [23] are evaluated. For PRNet, two versions have been evaluated here. For the original version, which is denoted as PRNet, half input points are selected as key points. For the modified version, which is denoted as PRNet*, all points are considered as key points for dense correspondence.

A. Evaluation on Non-rigid Data

Fig. 3 shows the quantitative comparison with 3D point clouds input on Surreal\SHREC. Our method always pro-
Fig. 4: Matching results on SHREC, where green and blue represent the source and target respectively. The points which are not assigned corresponding points (indicated by the red box) show that one-to-many matching exists in baselines.

Fig. 5: Matching precision on rigid data. Three sub-figures indicate three settings, i.e., UPC, UC and ND. Obviously, our method outperforms baselines with a big margin. It produces the best performance under different thresholds, which outperforms all baselines by a large margin. Meanwhile, S-CorrNet3D achieves second-top results, which is better than S-F3D. We speculate that the large deformation of input shapes is difficult for scene flow estimation. Besides, we present a qualitative comparison in Fig. 4. Here, we indicate some points by the red box which are not assigned the corresponding points. Note that the input point clouds are in one-to-one correspondence, thus the problem of one-to-many matching exists in baselines. Our method achieves the real one-to-one matching with high matching precision.

For a fair comparison, the proposed method is compared with those 3D mesh-based methods on re-meshed FAUST and SCAPE. Here, we use the vertices as our input point cloud. And the per-point-average geodesic distance results between the ground truth and predicted corresponding points are reported. Following [14], all results are multiplied by 100 for the sake of readability. From the Table I, our method gives accurate results on all settings, which is comparable with the state-of-the-art method. DeepGFM [14] outperforming other baselines. Moreover, we speculate that when the deformation becomes more severe, e.g., SCAPE dataset, the mesh-based method DeepGFM is more effective, where the topology information plays a more important role.

B. Evaluation on Rigid Data

In this section, we evaluate the dense correspondence estimation on rigid point clouds. For a comprehensive comparison, we also report the rigid transformation estimation results.

Following [24], [25], three different experiment settings are employed here. 1) Unseen point clouds (UPC): all the point clouds in the ModelNet40 are divided into training and test sets with the official setting. 2) Unseen categories (UC): in ModelNet40, we select the first 20 categories for training and the rest for testing. 3) Noisy data (ND): random Gaussian noise (i.e., \( N(0, 0.01) \)) is added to each point of input point clouds, while the sampled noise out of the range of \([-0.05, 0.05]\) will be clipped. The dataset split strategy is the same as UPC.

Table II: Evaluation of transformation estimation.

| Method      | RMSE(R)  | MAE(R)  | RMSE(t) \times 10^{-4} | MAE(t) \times 10^{-4} |
|-------------|----------|---------|------------------------|------------------------|
| ICP         | 12.282   | 12.707  | 11.971                 | 4.613                  |
| FGR         | 20.054   | 21.323  | 18.359                 | 7.146                  |
| PTLK        | 13.751   | 15.901  | 12.691                 | 6.403                  |
| DCP-v2      | 1.094    | 2.568   | 1.475                  | 0.752                  |
| PRNet       | 1.572    | 2.054   | 1.538                  | 0.854                  |
| PRNet*      | 1.449    | 1.543   | 1.449                  | 1.449                  |
| Ours        | 0.864    | 1.962   | 0.066                  | 0.006                  |

Table III: Matching precision on rigid data. The number of KNN for threshold.

| Method      | RMSE(R)  | MAE(R)  | RMSE(t) \times 10^{-4} | MAE(t) \times 10^{-4} |
|-------------|----------|---------|------------------------|------------------------|
| ICP         | 12.282   | 12.707  | 11.971                 | 4.613                  |
| FGR         | 20.054   | 21.323  | 18.359                 | 7.146                  |
| PTLK        | 13.751   | 15.901  | 12.691                 | 6.403                  |
| DCP-v2      | 1.094    | 2.568   | 1.475                  | 0.752                  |
| PRNet       | 1.572    | 2.054   | 1.538                  | 0.854                  |
| PRNet*      | 1.449    | 1.543   | 1.449                  | 1.449                  |
| Ours        | 0.864    | 1.962   | 0.066                  | 0.006                  |

Fig. 6: Comparison of post-processing, end-to-end learning.

Correspondence estimation. We draw the matching precision in Fig. 5. The proposed method outperforms other state-of-the-art methods with a big margin. These obtained results are reasonable because the DCP-v2 is a virtual point-based method that fails to predict the corresponding points and PRNet and PRNet* have serious one-to-many matching problem.

Rigid transformation estimation evaluation. We also provide a comparison of transformation estimation, which is achieved by the Procrustes algorithm [42] based on the learned correspondence. Following [24], [25], we use the metrics of root mean square error (RMSE) and mean absolute error (MAE) between the ground truth and prediction of the Euler angle and translation vector, notated as RMSE(R), MAE(R), RMSE(t), and MAE(t). Among all the experimental settings in Table II, our method achieves the best results on all metrics.

C. Ablation Studies

End-to-end training vs. Post-processing. We zero-out one assignment problem by end-to-end learning. Another more direct idea is to solve this non-convex problem by post-processing, i.e., learning the soft matching matrix and solving the permutation matrix during the test. From the comparison provided by Fig. 6, which is conducted on Surreal/SHREC, the end-to-end learning benefits the final results significantly because more accurate loss is calculated during the training.

IV. CONCLUSION

In this work, we advocate achieving one-to-one matching by learning the permutation matching matrix to tackle the dense correspondence estimation problem for 3D point clouds. Moreover, to guarantee the end-to-end learning and the accuracy of the calculated loss, an ingenious matching module has been designed in our method. Extensive experiments on both non-rigid and rigid 3D point cloud data have validated that our method achieves state-of-the-art performance. However, our method has limitations. Specifically, our method essentially determines the correspondences according to the similarity of the points. Thus, ours cannot deal with objects with strong symmetry, which is a stubborn problem in the community. In the future, we plan to extend our method to other dense correspondence estimation tasks, e.g., 2D-2D, 2D-3D.
