Neural Intrinsic Embedding for Non-rigid Point Cloud Matching

Puhua Jiang\textsuperscript{1,2}, Mingze Sun\textsuperscript{1}, Ruqi Huang\textsuperscript{1}

\textsuperscript{1}Tsinghua Shenzhen International Graduate School
\textsuperscript{2}Peng Cheng Laboratory

\{jph21, smz22\}@mails.tsinghua.edu.cn ruqihuang@sz.tsinghua.edu.cn

Figure 1. Given a point cloud, we select 5 landmarks (see red points on the right-most one) and assign each of the rest points to the cluster represented by its nearest neighbor among the landmarks in the respective embedded space. We compare our method to Euclidean coordinates, LIE [28], and GPS [40]. Our method takes in only the point cloud and produces segmentation that is intrinsic geometry-aware.

Abstract

As a primitive 3D data representation, point clouds are prevailing in 3D sensing, yet short of intrinsic structural information of the underlying objects. Such discrepancy poses great challenges in directly establishing correspondences between point clouds sampled from deformable shapes. In light of this, we propose Neural Intrinsic Embedding (NIE) to embed each vertex into a high-dimensional space in a way that respects the intrinsic structure. Based upon NIE, we further present a weakly-supervised learning framework for non-rigid point cloud registration. Unlike the prior works, we do not require expansive and sensitive off-line basis construction (e.g., eigen-decomposition of Laplacians), nor do we require ground-truth correspondence labels for supervision. We empirically show that our framework performs on par with or even better than the state-of-the-art baselines, which generally require more supervision and/or more structural geometric input.

1. Introduction

Estimating correspondences between non-rigidly aligned point clouds serves as a critical building block in many computer vision and graphics applications, including animation [21, 34], robotics [15, 44], autonomous driving [10, 54], to name a few. In contrast to the well-known rigid case, more sophisticated deformation models are in demand to characterize the non-rigid motions, for instance, articulation movements of human shapes.

To address this challenge, extrinsic methods in principle approximate a complex global non-rigid deformation with a set of local rigid and/or affine transformations, e.g., point-wise affine transformation [24, 50, 53], deformation graph [6, 7, 25], and patch-based deformation [23, 52]. Being intuitive and straightforward, the extrinsic deformation models are in general redundant and lack global structures. On the other hand, intrinsic methods [2, 8, 19, 29, 30, 33] first transform extrinsic coordinates into an alternative representation, in which shape alignment is performed. For instance, the seminal functional maps framework [32] utilizes eigenbasis of the Laplace-Beltrami operator as spectral embeddings and turns non-rigid 3D shapes matching into rigid alignment of high-dimensional spectral embeddings, under the isometric deformation assumption. However, spectral embeddings are generally obtained by an inefficient, non-differentiable off-line eigen-decomposition of the Laplacian operator defined on shapes, either represented as polygonal meshes [36] or point clouds [42]. Moreover, spectral em-
beddings are sensitive to various practical artifacts such as noise, partiality, and disconnectedness, to name a few.

To this end, we follow the isometric assumption and first propose a learning-based framework, Neural Intrinsic Embedding (NIE), to embed point clouds into a high-dimensional space. In particular, we expect our embedding to satisfy the following desiderata: (1) It is aware of the intrinsic geometry of the underlying surface; (2) It is computationally efficient; (3) It is robust to typical artifacts manifested in point clouds. Our key insight is that geodesics on a deformable surface, which are inherently related to the Riemannian metric, contain rich information of the intrinsic geometry. Therefore, NIE is trained such that the Euclidean distance between embeddings approximates the geodesic distance between the corresponding points on the underlying surface. In particular, considering the local tracing manner of geodesic computation, we choose DGCNN [49] as our backbone, which efficiently gathers local features at different abstraction levels. We also carefully formulate a set of losses and design network modifications to overcome practical learning issues including rank deficiency, and sensitivity to point sampling density. As a consequence, NIE manages to learn an intrinsic-aware embedding from merely unstructured point clouds. Fig. 1 demonstrates that we obtain the segmentation result closest to the ground truth based on geodesic distances.

Furthermore, based on NIE, we propose a Neural Intrinsic Mapping (NIM) network, a weakly supervised learning framework for non-rigid point cloud matching. Though closely related to the Deep Functional Maps (DFM) frameworks, our method replaces the spectral embedding with the trained NIE and further learns to extract the optimal features based on a self-supervised loss borrowed from [14]. In the end, we establish a pipeline for weakly supervised non-rigid point cloud matching, which only requires all the point clouds to be rigidly aligned and, for training point clouds, access to the geodesic distance matrices of them.

Our overall pipeline is simple and geometrically informative. We conduct a set of experiments to demonstrate the effectiveness of our pipeline. In particular, we highlight that (1) our method performs on par with or even better than the competing baselines which generally require more supervision and/or more structural geometric input on near-isometric point cloud matching; (2) our method achieves sensible generalization performance, thanks to our tailored design to reduce the bias of point sampling density; (3) our method is robust regarding several artifacts, including noise and various partiality.

2. Related Work

Non-rigid point cloud matching This is a challenging task due to the complexity of modeling non-rigid deformations. Extrinsic methods [6, 7, 23–25, 50, 52, 53] approximate a complex global non-rigid deformation with a set of local rigid and/or affine transformations.

On the other hand, intrinsic approaches, especially the spectral-based techniques, leverage the geometric information encoded in the eigenbasis of Laplace-Beltrami operators, which lift the matching problem into a high-dimension space, where a family of isometric non-rigid deformations is well characterized. The structural benefits are attained at the cost of a significantly larger search space for the optimal transformation. We conclude some typical spectral embeddings in the following.

Geometric Embeddings Pioneered by the work [38], the eigenbasis of the Laplace-Beltrami operator plays a dominant role in geometry processing for decades. Especially, several early approaches [9, 26, 40] attempt to establish the connection between eigenbasis and surface geodesics, which encode essentially the intrinsic geometry. However, due to the computational burden and the noise-prone nature of high-frequency eigenfunctions, this line of work usually uses relatively low-frequency eigenbasis, yielding only rough approximation in recovering geodesic distances.

Related to this topic, there are also approaches directly optimizing for embeddings that best recover the underlying geodesics. For instance, MDS [47] is a classical dimension reduction method, which can achieve reasonably accurate embedding by minimizing certain stress. More recently, by exploiting the structural properties of the geodesics on the surface, GeodesicEmbedding [51] is proposed to build a hierarchical embedding, which in turn helps to reduce computing time of inferring geodesic distance on high-resolution meshes. While these approaches achieve relatively high recovery accuracy, we point out that they both require the ground-truth geodesic distances as input, thus not suitable for our target.

(Deep) Functional Maps Another line of work that is closely related to ours is the functional maps framework [32]. In the functional space, a correspondence can be represented by a small matrix encoded in a reduced eigenbasis and computed as the optimal transformation that aligns a given set of probe functions possibly with other regularization. Early works along this line take mostly an axiomatic approach [18, 20, 22, 31], while in recent years a trend of integrating functional maps mechanism into a learning pipeline is attracting extensive attention [13, 17, 27, 39]. While most of the deep functional maps frameworks follow the utility of spectral embeddings and refine features upon some hand-crafted descriptors, e.g., HKS [43], WKS [3], SHOT [46]. By leveraging full [12] or weak [41] supervision, networks are capable of extracting features directly from point clouds. Furthermore, exploration on how to establish embeddings to take over the spectral ones is also taken in [28], which again relies heavily on the supervision over shape correspondences.
3. Background

For the sake of completeness, we briefly review the basic notions of functional map [32], deep functional maps framework, and the framework of Linear invariant embedding [28] (LIE), which are closely related to our framework.

**Functional Maps** Functional maps [32] is an alternative representation of point-wise maps, which is formulated primarily upon the eigenbasis of the Laplace-Beltrami operator. Given a pair of shapes $S_1, S_2$, one first computes the first $k$ eigenfunctions and store them as matrices $\Phi_i \in \mathbb{R}^{n_i \times k}$, $i = 1, 2$. Now, given a point-wise map encoded as a permutation matrix $\Pi_{21} \in \mathbb{R}^{n_2 \times n_1}$, the functional representation is

$$C_{12} = \Phi_1^\dagger \Pi_{21} \Phi_1 \in \mathbb{R}^{k \times k},$$

where $\dagger$ denotes the Moore Penrose pseudo-inverse. Regarding the inverse conversion, one can compute via nearest neighbor search between the rows of $\Phi_i C_{12}$ and that of $\Phi_1$.

One of the key properties of functional maps is that, by introducing the spectral embeddings, i.e., $\Phi_1, \Phi_2$, one can express global map priors in simple algebraic forms in terms of $C_{12}$. For instance, area-preserving maps are supposed to correspond to orthogonal functional maps. In other words, one can add $\|C_{12}^T C_{12} - I\|_2$ as regularization to promote such property.

**Deep Functional Maps** The above insight in turn gives rise to Deep Functional Maps (DFM) framework, which was first proposed in [27]. In a nutshell, DFM is designed as a Siamese network, which aims to learn a universal feature extractor $G$ over a set of training pairs, and output the optimal functional maps from the trained model, which can be converted to point-wise maps in the end. In fact, this is the first stage of training, with no need of any further structural information, e.g., triangulation.

Therefore, one can formulate the following optimization problem:

$$C_{12} = \arg \min_{C \in \mathbb{R}^{k \times k}} \left\| C_{12} \Phi_1 G_1 - \Phi_2 G_2 \right\|_2 + E_{\text{Reg}}(C_{12})$$

Equipped with the pre-computed spectral embeddings and some proper initial features (e.g., WKS [3]), one can optimize $G$ over a set of training pairs, and output the optimal functional maps from the trained model, which can be converted to point-wise maps in the end. In fact, this is the basic design shared by several recent unsupervised DFM frameworks [13, 17, 39].

**Linearly Invariant Embedding** [28] It is evident that the key ingredient of functional maps representation is the spectral embeddings, which allow to encode point-wise maps into compact transformation matrices, but also integrate and optimize map priors efficiently. LIE is the first work aiming to learn a basis in place of spectral embedding.

The key insight of LIE is that, given a collection of shapes and ground-truth correspondences among them, one can learn a basis generator that consumes a point cloud $X \in \mathbb{R}^{n \times 3}$ as input and return $k$-dimensional basis, i.e.,

$$F(X) = \Phi_X \in \mathbb{R}^{n \times k}$$

Similar to Eqn. 1, given a ground-truth map $\Pi_{YX}$ from $Y$ to $X$, one can write the corresponding “functional map” as

$$C_{XY} = \Phi_1^\dagger \Pi_{YX} \Phi_X.$$
It seems then plausible to train a network with the following loss
\[
L(\Theta_B) = \sum_{i} \sum_{(p,q) \in S_i} \frac{|d^i_E(v_p, v_q) - d_S(v_p, v_q)|^2}{d_S(v_p, v_q)^2},
\]

**Relative Geodesic Loss** However, the above naive loss, using absolute geodesic error, is prone to favoring long geodesic distance preservation within the embedding. This would in turn hamper the local distance preservation, due to the limited capacity of the network and the finite embedding dimension. Thus, we instead use the loss penalizing the relative geodesic error:
\[
L_G(\Theta_B) = \sum_{i} \sum_{(p,q) \in S_i} \frac{|d^i_E(v_p, v_q) - d_S(v_p, v_q)|^2}{d_S(v_p, v_q)^2},
\]

**KL Loss** Furthermore, since preservation of local geometry is critical for obtaining fine-grained correspondences, we strengthen short-distance recovery from a statistical point of view as follows. Given a vertex \(v_p \in S_i\), we compute the two distances from it to all the other vertices \([d_S(v_p, v_1), d_S(v_p, v_2), \ldots, d_S(v_p, v_n)]\) and \([d^i_E(v_p, v_1), d^i_E(v_p, v_2), \ldots, d^i_E(v_p, v_n)]\). We then define a distribution by:
\[
P_S^p(v_q) = \frac{\exp(-\alpha d_S(v_p, v_q))}{\sum_q \exp(-\alpha d_S(v_p, v_q))}, \quad \forall v_q \in S_i
\]
Similarly, we can define another distribution \(P_E^p\) with respect to the embedded distance \(d^i_E\). Then we define a loss based on KL-divergence between distributions:
\[
L_{KL}(\Theta_B) = \sum_i \sum_p KL(P_E^p, P_S^p)
\]

**Bijectivity Loss** Training with the two losses above, we observe that the relative geodesic error of the network saturates at \(k = 8\). Interestingly, this finding also agrees with [51], where the authors find that their MDS-like embedding method also saturates at the same dimension. One consequence of this saturated performance is that further increasing embedding dimension leads to rank deficiency, which results in irreversible transforms with respect to the rank-deficient embeddings.

To address this issue, we take a self-supervised approach. Namely, we apply furthest point sampling to sample 2000 vertices from the \(X_i\), and then we let \(X^a_i, X^b_i\) be the first and second 1000 vertices from the sampled vertices. Note that by construction they are evenly distributed on the surface and well separated. We then compute the point-wise maps \(T_{ab}, T_{ba}\) between \(X^a_i\) and \(X^b_i\) via simply nearest neighborhood searching, as the two point sets are on the same surface. Finally, we write the point-wise into the form of permutation matrices \(\Pi_{ab}, \Pi_{ba}\), and according to Eqn. 1, we have
\[
C_{ab} = \phi_i^b \Pi_{ba} \phi_i^a, C_{ba} = \phi_i^a \Pi_{ab} \phi_i^b.
\]
Finally, we formulate the bijectivity loss as follows:
\[
L_B(\Theta_B) = \sum_{a,b,i} ||C_{ab}C_{ba} - I||_F + ||C_{ba}C_{ab} - I||_F^2.
\]

Putting every piece together, the total loss is written as:
\[
L_{total} = \lambda_1 L_G + \lambda_2 L_{KL} + \lambda_3 L_B.
\]

**Amelioration of Sampling Density Bias:** Apart from the aforementioned issues, we also encounter another problem hindering training – point clouds may manifest varying sampling density across the underlying surface. Especially, the vanilla DGCNN implements local feature aggregation via \(k\)-nearest neighbor search, which is unaware of density distribution. This issue can significantly impact our generalization capacity, as each dataset owns its specific sampling pattern. To this end, we propose a simple yet effective modification on DGCNN as follows.

Given a point cloud \(X\), we first conduct the furthest point sampling on \(X\) to obtain an evenly distributed subset \(X_s\). Now, given a point, \(v_p\), instead of searching directly its \(k\)-NN within \(X\), we first find its nearest neighbor, \(v_p\), in \(X_s\), and then assign the \(k\)-NN of \(v_p\) within \(X_s\) to \(v_p\). The above description is well illustrated in Fig. 2, where \(X_s\) are colored purple.

In the end, we remark that sampling density bias is not a new issue – several prior works [12, 28, 41] that aim to learn feature/basis directly from non-rigid point clouds may have encountered the same problem. As a typical solution, the prior works also apply FPS sampling to ensure a relatively even distribution. In the case where spectral embeddings are available [12, 41], the authors simply leverage the fact that eigenbasis is insensitive to point distribution and estimate only functional maps. On the other hand, LIE [28] circumvents this problem by heavily downsampling in both train and test point clouds (to \(1k\) vertices), resulting in a dataset of low resolution. We highlight in Table 4, by utilizing our modified DGCNN, the generalization capacity of our method is largely enhanced.
4.2. Neural Intrinsic Mapping

In this section, we formulate our NIM network. In essence, the network belongs to the family of the deep functional maps reviewed in Section 3, though bears two main modifications as shown in Fig. 3: (1) we replace the pre-computed eigenbasis with NIE proposed in Section 6.1 (denoted by \( \Phi_X, \Phi_Y \) in the figure); (2) we remove the original structural losses on functional maps and instead use a self-supervised loss introduced in \([14]\), which is defined in terms of geodesic information to guide feature learning.

In a nutshell, NIM learns to predict a set of optimal descriptors \( G_X = \mathcal{G}_{\Theta_D}(X) \) and \( G_Y = \mathcal{G}_{\Theta_D}(Y) \) from input point clouds \( X \) and \( Y \), here \( \Theta_D \) is the set of learnable parameters. Once learned, the map from \( Y \) to \( X \) encoded in our NIE is given by:

\[
C = A_X A_Y^\dagger = \left( \Phi_Y^\dagger G_Y \right) \left( \Phi_X^\dagger G_X \right)^\dagger \tag{8}
\]

Similarly, we have \( \tilde{C} \) form \( X \) to \( Y \):

\[
\tilde{C} = A_Y A_X^\dagger = \left( \Phi_X^\dagger G_X \right) \left( \Phi_Y^\dagger G_Y \right)^\dagger \tag{9}
\]

Since we do not need any correspondence label, in order to make full use of the geodesic distance information, we convert the functional map \( C \) into a soft correspondence map and follow deep cyclic mapping \([14]\) to design our unsupervised loss. Given \( C, \Phi_X, \Phi_Y \) the soft correspondence matrix mapping between input point clouds \( X \) and \( Y \) can be computed as:

\[
P = \text{softmax}(-\alpha \| \Phi_X C - \Phi_Y \|_2) \tag{10}
\]

where each entry \( P_{ij} \) is the probability the \( j \)-th point in \( X \) corresponds to the \( i \)-th point in \( Y \), and \( \alpha \) is a hyper-parameter controlling the entropy of the probability distribution.

Similarly, we can compute the inverse map:

\[
\tilde{P} = \text{softmax}(-\alpha \| \Phi_Y \tilde{C} - \Phi_X \|_2) \tag{11}
\]

Then the cyclic distortion \([14]\) is

\[
L_{\text{cyclic}}(X, Y) = \frac{1}{|X|^2} \left\| (D_X - (\tilde{P} P) D_X (\tilde{P} P)^T) \right\|^2_F + \frac{1}{|Y|^2} \left\| (D_Y - (P \tilde{P}) D_Y (P \tilde{P})^T) \right\|^2_F \tag{12}
\]

where \( D_X, D_Y \) are the geodesic distance matrix regarding \( X \) and \( Y \), respectively.

Thus the total loss for descriptor learning is:

\[
L_{\text{desc}}(X, Y) = L_{\text{isometric}}(X, Y) + L_{\text{cyclic}}(X, Y) \tag{14}
\]

Map Inference via NIE and NIM Once we have trained the NIE and the NIM network, \( \mathcal{F}_{\Theta_D} (\cdot) \), \( \mathcal{F}_{\Theta_D} (\cdot) \), we can estimate the correspondence between a pair of rigidly aligned point clouds \( X \) and \( Y \) as follows: (1) Compute neural intrinsic embeddings \( \Phi_X, \Phi_Y \). (2) Compute the set of learned features, \( G_X, G_Y \). (3) Compute \( C_{XY} \) according to Eqn. 8. (4) Compute the point-wise correspondences as described in Section 3.

5. Implementation

We implemented our pipeline in PyTorch \([35]\) by adapting the implementation of DGCNN \([49]\) released by the authors. Our network contains three EdgeConv layers mapping the input dimension from 3 to 64 and then to 512, followed by three convolutions layers reducing the dimension from 512 to output. We use the same backbone for both the basis and descriptor generator networks, only the output feature dimension differs. We always train with basis dimension of 20 and descriptor dimension of 40. We refer readers to the supplementary material for more details. During inference, given a point cloud/mesh of around 5000 points, NIE takes 4.9 ms to generate basis, which is comparable to LIE (3.0 ms) and faster than computing LBO basis (10 ms).

6. Experimental Results

In this section, we demonstrate a set of experiments, comprised of three main parts as follows. First of all,
in Section 6.1, we evaluate our learned embeddings and provide ablation studies to justify our proposed design. Secondly, in Section 6.2, we demonstrate the matching results of our proposed NIM network and compare it to several competitive baselines. Finally, in Section 6.3, we demonstrate the robustness of our NIE and NIM network with respect to artifacts including noise and various partialities. We report all matching results in terms of mean geodesic error on shapes normalized to the unit area, even in the case that only point clouds are fed in inference time.

Datasets We provide details on the involved datasets: FAUST,r: The remeshed version [37] of FAUST dataset [5] contains 100 human shapes. We split the shapes as 80/20 for training and testing. SCAPE,r: The remeshed version [37] of SCAPE dataset [1] contains 71 human shapes. We split the shapes into 51/20 for training and testing. SURREAL,r: We randomly sampled 120 human shapes from SURREAL dataset [48], and perform remeshing so that each shape has around 5000 points. We split the shapes into 100/20 for training and testing.

6.1. Embedding Evaluation

Embedding Quality We compare our NIE with several embeddings including the Euclidean coordinates (properly centered and normalized), MDS [47], eigenbasis of the Laplace-Beltrami operator [40] defined on meshes, eigenbasis of the Laplacian operator [4] defined on point clouds, and LIE [28]. For a fair comparison, we set all embeddings including the Euclidean coordinates (properly center- and normalized), MDS [47], eigenbasis of the Laplacian operator [4] defined on point clouds, and LIE [28]. For a fair comparison, we set all embeddings to be of dimension 20, with an exception of Euclidean coordinates. Regarding LIE and our method, we train the basis generator network on the 51 training shapes from SCAPE,r dataset, and evaluate all the basis, either constructed or learned, on the rest 20 test shapes.

We evaluate all the embeddings via two metrics proposed before: (1) the relative geodesic error (Eqn. 4); (2) the metric OPT introduced in [28]: Given a pair of point clouds \(X, Y\), together with the ground-truth correspondence \(\Pi_{YX}\), we first use Eqn. 1 to encode \(\Pi_{YX}\) into a matrix regarding an embedding, then we recover the point-wise map from the matrix, and evaluate the geodesic error of the recovered map regarding \(\Pi_{YX}\).

As shown in Table 1, it is indeed expected that our method performs the best regarding the first metric, since we train our network using exactly the same loss. While there is no related constraint on LBO, PC-LBO, and LIE, leading to significant relative geodesic errors. It is worth noting, though, our method outperforms MDS as well, which takes as input the ground-truth geodesic matrices. This is because MDS regresses embeddings with respect to the absolute geodesic error, which naturally favors long-distance preservation. And interestingly, in terms of OPT, MDS20 is also outperformed by our method, suggesting the rationality of training with relative geodesic error.

On the other hand, it is remarkable that our method performs the best in \(\text{OPT}\). Especially, LIE enforces the encoded ground-truth maps to be orthogonal during training, which imposes strong structural prior on the \(\text{OPT}\) metric, while our pipeline is trained without any related prior.

Ablation on NIE Design In Table 2 we report ablation studies on the training loss terms and our modified DGCNN. When only the relative geodesic loss \(L_G\) is used, though we can get the lowest error, NIE suffers from a rank deficiency problem, which in turn leads to the worst \(\text{OPT}\) score. Adding the bijectivity loss \(L_B\) effectively retains full rank and improves the \(\text{OPT}\) score by 20%. Combining the KL loss \(L_{KL}\), we further improve the \(\text{OPT}\) score as well as the relative geodesic error. Finally, integrated with our modified version of DGCNN, our full model performs the best in the ablation study. We also report the resulting matching error of the NIM regarding each variant, it is evident that our loss design effectively improves the matching performance.

6.2. Near-isometric point cloud matching

Baselines We compare our method with a set of baselines, which are categorized depending on if mesh information is required during inference time: (1) BCICP [37], SURFMNet [39], UnsupFMNet [17], NeuroMorph [13], FMNet [27], WSupFMNet [41] in which meshes are required for computing eigenbasis; (2) 3D-CODED [16], CorrNet-3D [56], LIE [28], on the other hand, can directly predict point-wise maps based on point clouds as test input. The used supervision is indicated next to each method in the table: Unsupervised, Supervised, Weakly-supervised.

First, we train models on FAUST,r and SCAPE,r datasets respectively. In particular, we train our NIE and NIM network both with ground-truth geodesic information.
Table 3. Mean geodesic errors (×100) of the different methods on near-isometric point cloud matching. The best results are highlighted separately for methods with and without mesh during inference.

<table>
<thead>
<tr>
<th>Method</th>
<th>F</th>
<th>S</th>
<th>F on S</th>
<th>S on F</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCICP [37]</td>
<td>15.</td>
<td>16.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SURF-MNet(U) [39]</td>
<td>15.</td>
<td>12.</td>
<td>32.</td>
<td>32.</td>
</tr>
<tr>
<td>UnsupFMNet(U) [17]</td>
<td>10.</td>
<td>16.</td>
<td>29.</td>
<td>22.</td>
</tr>
<tr>
<td>NeuroMorph(U) [13]</td>
<td>8.5</td>
<td>30.</td>
<td>29.</td>
<td>18.</td>
</tr>
<tr>
<td>FMNet(S) [27]</td>
<td>11.</td>
<td>12.</td>
<td>30.</td>
<td>33.</td>
</tr>
<tr>
<td>WSupFMNet(W) [41]</td>
<td>3.3</td>
<td>7.3</td>
<td>12.</td>
<td>6.2</td>
</tr>
<tr>
<td>NIM w/ LBO basis 20</td>
<td>5.8</td>
<td>13.</td>
<td>22.</td>
<td>16.</td>
</tr>
<tr>
<td>3D-CODED(S) [16]</td>
<td>2.5</td>
<td>31.</td>
<td>31.</td>
<td>33.</td>
</tr>
<tr>
<td>CorrNet-3D(U) [56]</td>
<td>63.</td>
<td>58.</td>
<td>58.</td>
<td>63.</td>
</tr>
<tr>
<td>LIE(S) [28]</td>
<td>3.6</td>
<td>12.</td>
<td>19.</td>
<td>12.</td>
</tr>
<tr>
<td>Ours(W)</td>
<td>5.5</td>
<td>11.</td>
<td>15.</td>
<td>8.7</td>
</tr>
</tbody>
</table>

Table 4. Mean geodesic errors (×100) when trained on Surreal and tested on re-meshed Faust and Scape.

<table>
<thead>
<tr>
<th>Method</th>
<th>S</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>CorrNet-3D [56]</td>
<td>52.</td>
<td>54.</td>
</tr>
<tr>
<td>LIE [28]</td>
<td>20.</td>
<td>15.</td>
</tr>
<tr>
<td>Ours</td>
<td>10.</td>
<td>6.5</td>
</tr>
<tr>
<td>CorrNet-3D Noise</td>
<td>58.</td>
<td>62.</td>
</tr>
<tr>
<td>LIE Noise</td>
<td>20.</td>
<td>15.</td>
</tr>
<tr>
<td>Ours Noise</td>
<td>11.</td>
<td>7.2</td>
</tr>
</tbody>
</table>

computed on the meshes from the training set. In Table 3, we report the normal matching errors as well as generalized matching errors. For instance, the column \( \text{F on S} \) reads that training on FAUST\(_r\) but test on SCAPE\(_r\). In Table 3, the best score from each category are highlighted in bold. Our method performs the best in 3 out of 4 terms among the competing methods of the same category. Indeed, our score is also the second best of all methods in the table with respect to the 3 terms, only being outperformed by WSupFMNet [41] by a reasonable margin, given the fact that the latter uses eigenbasis of the Laplace-Beltrami operator(LBO). Finally, we replace our NIE with 20 LBO basis obtained from the meshes in training NIM (see NIM w/ LBO basis 20 in Tab. 3) and obtain deteriorated results. Though we follow the same losses with [14], the latter uses SHOT [45] descriptor as input, which is absent in NIM.

We report further the generalization capacity in Table 4. In this case, we train our NIE and NIM network, as well as the baseline methods, on SURREAL\(_r\), and then infer on datasets SCAPE\(_r\) and FAUST\(_r\). In this case, we mainly compare CorrNet-3D [56] and LIE [28]. It is evident from the top half of Table 4 that our method generalizes the best, with 50% and 56.7% matching error reduction upon LIE. We also provide qualitative illustrations on the computed maps from different approaches in Fig. 4.

Finally, we demonstrate that, given a trained NIE, one can even train a NIM network on a different training set, where geodesic information is absent. More specifically, we first train the NIE module on SURREAL\(_r\) dataset. Then given a set of point clouds from other datasets, e.g., the training set of FAUST\(_r\), we can use the trained NIE to embed the unseen point clouds, and to approximate the geodesic distances with Euclidean distances among the embeddings. In the end, we train NIM with the point clouds from FAUST\(_r\) and the respective approximated geodesics.

Fig. 6 shows the results of the above learning protocol. As a strong baseline, we train two NIM networks on FAUST\(_r\) and on SCAPE\(_r\), which exploit the full information from the respective dataset. As shown in Fig. 6, our method, without any ground-truth geodesic information from the dataset of interest, achieves decent performance even compared to the models trained with full information.

6.3. Robustness

In this section, we show that our NIM network is robust with respect to typical artifacts including noise, various partialities, and even disconnectedness. We start our ex-
Figure 6. Comparison between directly inferring on datasets and fine-tuning on datasets. Our method achieves decent performance compared to the models trained with full information.

<table>
<thead>
<tr>
<th>Method</th>
<th>half</th>
<th>hole</th>
<th>cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIE [28]</td>
<td>15.</td>
<td>15.</td>
<td>16.</td>
</tr>
<tr>
<td>Ours</td>
<td>10.</td>
<td>7.0</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Table 5. Mean geodesic errors (×100) for partial point cloud matching.

experiments following the setting presented in Table 4, however this time we perturb the input point clouds by Gaussian noise. As shown in Table 4, our accuracy still significantly outperforms the competing baselines by a large margin. We also provide a qualitative evaluation in Fig. 5.

Then we further test our method together with the baselines on point clouds undergoing three types of partiality, namely, half, hole, and cut. For half, we simulate a camera in front of the point clouds and therefore capture half of the data. For hole, we randomly choose 10 points on the surface and remove 100 nearest points around. For cut, we randomly cut a part of the legs or arms. Table 5 shows the quantitative results for partial shape matching, in which we estimate point-wise maps from a partial shape to full shapes (see Fig. 7 for illustration). For hole and cut, the matching performance only decreases a little. As for half, though nearly half of the data are removed, our method still returns reasonable results. In particular, in Fig. 7, we compare qualitatively our results with LIE [28], where we find a noticeable discrepancy of the latter. Overall, even trained without any ground-truth correspondence, our NIM network is capable of retrieving intrinsic information from corrupted data that are completely unseen during training.

7. Conclusion, Limitations and Future Work

To conclude, in this paper we first propose NIE, a learning-based framework that embeds unstructured point clouds into high-dimensional space in a way that respects the intrinsic geometry of the underlying surfaces. Then, based on NIE, we present NIM, a weakly supervised non-rigid point cloud matching network. NIM only assumes the training point clouds to be approximately rigidly aligned, and require nothing more than geodesic distances among the training point clouds, which can even be approximated by a trained NIE. We demonstrate in a set of comprehensive experiments that: (1) NIE effectively learns intrinsic information and therefore allows for structured map encoding; (2) NIM enjoys decent matching performance and excellent generalization capacity; (3) Both NIE and NIM are robust to common artifacts, including noise and various partiality.

The main limitation of our framework is its sensitivity regarding the extrinsic pose of point clouds. As shown in Fig. 8, when shapes are reasonably aligned, our NIM can estimate high-quality maps even in the presence of significant pose differences. However, when the rigid alignment is inaccurate due to uncommon poses, the estimated maps are hampered, either by severe symmetric flip(bottom middle), or erroneous intrinsic embedding (bottom right). It would be an interesting future work to incorporate the recent advances in $SO(3)$-invariant and -equivariant [11, 55] networks to enhance our pipeline.

Acknowledgement This work was supported in part by the National Natural Science Foundation of China under contract No. 62171256, in part by Shenzhen Key Laboratory of next-generation interactive media innovative technology (No. ZDSYS20210623092001004).
References


