Supplementary Materials for:
Generalizable Local Feature Pre-training for Deformable Shape Analysis

Souhaib Attaiki  Lei Li  Maks Ovsjanikov
LIX, École Polytechnique, IP Paris

In this document, we collect all the results and discussions, which, due to the page limit, could not find space in the main manuscript. This supplementary material consists of two parts. First, in Appendix A, we describe more implementation details mainly regarding our pilot study, local feature pre-training, and experiments on downstream deformable shape data. Next, in Appendix B, we present additional experimental results and analysis of our local feature-pretraining strategy and its generalization in downstream tasks, including deformable shape matching and segmentation.

A. Implementation Details

A.1. Feature Locality vs. Transferability

In Sec. 3 of the main text, we conducted a pilot study on feature locality vs. transferability on deformable shapes. We tested three different architectures for pre-training a local feature extractor, and their details are as follows.

**SparseConv.** We used the ResNet14 architecture introduced in [2]. During pre-training, given a 3D point cloud \( P \), a fixed-size local patch with a radius of 0.15 is cropped at point \( p \in P \) and then reoriented with a local reference frame (LRF) computed by the method in [4] for rotation invariance. The resulting local patch is fed to the sparse convolution network, which extracts a 32-dimensional feature vector for point \( p \).

**PCPNet.** It is a variant of PointNet [17] endowed with a quaternion spatial transformer. We used the single-scale architecture proposed by [5]. PCPNet is designed to be a local network requiring input patches to have a fixed number of points. Thus during pre-training, a fixed-size local patch (radius = 0.15) is cropped at point \( p \) and reoriented by an LRF. The local patch is then resampled to 1,024 points and fed to the network, resulting in a 32-dimensional feature vector for point \( p \).

**3DCNN.** We used the architecture from [9] with a learnable receptive field size and differentiable voxelization, the same as our VADER in Sec. 4.1 of the main text. More details can be found in Appendix A.2.

**Dataset.** We pre-trained the above local networks on the 3DMatch dataset, which is a collection of RGB-D scan datasets with 62 indoor scenes and 4,142 point cloud fragments. There are 13K points on average in a fragment after downsampling.

**Loss.** We used the PointInfoNCE loss, in which 300 point correspondences were randomly sampled for a pair of point clouds for faster training and the temperature parameter \( \tau \) was set to 0.07.

We also used the cycle consistency loss \( \mathcal{L}_c \). During pre-training, we use the extracted features to build correspondences for rigid alignment between shapes \( P \) and \( Q \). The intuition for \( \mathcal{L}_c \) is that the estimated transformation \( (R, t) \) aligning \( P \) to \( Q \) should be the inverse of the transformation \( (R', t') \) aligning \( Q \) to \( P \). Mathematically, this can be expressed as:

\[
\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R' & t' \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} RR' & Rt' + t \\ 0 & 1 \end{bmatrix} = I \quad (4)
\]

**Application to deformable shape matching.** In Fig. 3 of the main text, we have shown the results of shape matching on the Faust Remeshed dataset, directly using the pre-trained feature extractors. Given two shapes \( S_1 \) and \( S_2 \), we compute their respective point-wise features \( F_1 \) and \( F_2 \) using a specific pre-trained model. We first produce an estimate of the point-to-point maps \( T_{21}^{nn} \) and \( T_{12}^{nn} \) using nearest neighbor search between \( F_1 \) and \( F_2 \). We then filter the correspondences by mutual check: a pair of points \( x \in S_1, y \in S_2 \) is considered to be in correspondence, if and only if in the feature space, \( x \) is the nearest neighbor of \( y \), and \( y \) is the nearest neighbor of \( x \). This results in two filtered maps \( T_{21}^{nn} \) and \( T_{12}^{nn} \). Finally, we further refine these two maps using the ZoomOut method [13], which is based on navigating between the spectral and spatial domains while progressively increasing the number of spectral basis functions. We emphasize that if the initial point-to-point map is noisy or contains strong ambiguities like symmetry ambiguities, ZoomOut is not able to remedy these errors, thus leading to final correspondences of bad qual-
ity. We perform 10 iterations of ZoomOut, starting from 30 eigenfunctions up to 100 eigenfunctions.

A.2. Local Feature Pre-training

In Sec. 4.1 of the main text, we introduced our local feature pre-training strategy.

Feature extraction. We use $r_{\text{LRF}} = 0.3$ and $\sigma = 10^{-3}$ for differentiable voxelization [9], and the voxel grid resolution is set to $16^3$. We pre-trained on the 3DMatch dataset introduced in Appendix A.1.

Pre-training loss. For the PointInfoNCE loss $L_{\text{nce}}$, its settings are described in Appendix A.1. For the cycle consistency loss $L_{\text{c}}$, 300 points were randomly sampled on each point cloud for feature extraction and alignment estimation. A relaxation-based solver is used in $L_{\text{c}}$ for estimating a 3D transformation between two point clouds, and its details can be found in [9].

In the main text, we investigated the performance difference between the cycle consistency loss and PointInfoNCE loss w.r.t learned feature smoothness. Suppose that $F \in \mathbb{R}^{m \times n}$ is the matrix of extracted $n$-dimensional point-wise features for a shape of $m$ vertices, we measure the Dirichlet energy as follows:

$$E_{\text{Dirichlet}}(F) = \frac{1}{n} \sum_{i=1}^{n} F_i^T W F_i,$$

(5)

where $F_i$ is the $i^{th}$ column of $F$, and $W$ is the standard stiffness matrix computed using the classical cotangent discretization scheme of the Laplace-Beltrami operator [15].

A.3. Baselines

In Sec. 5 of the main text, we tested our proposed VADER features against a wide spectrum of competitors, including both hand-crafted and learned features.

Specifically, the Heat Kernel Signature (HKS) and Wave Kernel Signature (WKS) features are both sampled at 100 values of energy $t$, logarithmically spaced in the range proposed in their respective original papers. SHOT descriptors are 352-dimensional, and we used the implementation from the PCL library [18]. PointContrast features are 32-dimensional, and we used the publicly available implementation and the pre-trained weights released by the authors.

A.4. Downstream Shape Analysis Training

In Sec. 5 of the main text, we used DiffusionNet on top of the baselines features and our VADER respectively, in both the shape matching and segmentation tasks. We employed the publicly available implementation of DiffusionNet released by the authors. Unless specified otherwise,

1https://github.com/facebookresearch/PointContrast
2https://github.com/nmwsharp/diffusion-net

in our experiments, we used four DiffusionNet blocks of width = 128. The DiffusionNet is trained by an ADAM optimizer [7] with an initial learning rate of $10^{-3}$.

In Sec. 5.1 of the main text, we also used a point-wise MLP network on top of the baselines features and our VADER respectively for supervised shape matching. For this, we use the same MLP architecture as in FMNet [10]. After computing the point features with the MLP, we use them to compute the predicted functional map $C_{\text{pred}}$ as in [3] and penalize its deviation from the ground-truth map $C_{\text{gt}}$ using the L2 loss: $L = \|C_{\text{pred}} - C_{\text{gt}}\|_2^2$.

A.5. Computational Specifications

All our experiments were executed using Pytorch [14], on a 64-bit machine, equipped with an Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and an RTX 2080 Ti Graphics Card.

In terms of computational time, pertaining our method takes about 12 hours on a single RTX 2080 Ti Graphics Card, in contrast to the 64 hours required for PointContrast. The receptive field optimization takes about 20 minutes per dataset. For feature extraction, our method takes 3 seconds to extract local features for a 5000-vertex shape, which is on par with other local features like SHOT [19], but slower than PointContrast (0.1s). Finally, the forward pass using VADER takes the same time as for all baseline features, e.g., 0.2 seconds per iteration for the unsupervised shape-matching experiment in Sec 5.1 of the main text.

B. Additional Results and Analysis

B.1. Size of the learned receptive field

Fig. 5 of our paper provides an illustration of the optimized receptive field in downstream tasks. In Fig. 10, we include more visualizations for shape pairs for both humans and animals. Observe that the optimized receptive field indeed corresponds to interpretable concepts, such as the head or foot of a human, and is consistent across shape pairs.
Table 6. Accuracy of various features for unsupervised shape matching on un-aligned data. X-Y means train on X and test on Y. Values are mean geodesic error $\times 100$ on unit-area shapes.

<table>
<thead>
<tr>
<th>Method / Dataset</th>
<th>FR-SR</th>
<th>SR-FR</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURFMNET</td>
<td>15.2</td>
<td>9.5</td>
</tr>
<tr>
<td>Cyclic FMaps</td>
<td>23</td>
<td>23.2</td>
</tr>
<tr>
<td>WSUpFMNet</td>
<td>27.1</td>
<td>14.2</td>
</tr>
<tr>
<td>Deep Shells</td>
<td>6.0</td>
<td>3.4</td>
</tr>
<tr>
<td>DiffusionNet - XYZ</td>
<td>25.7</td>
<td>8.4</td>
</tr>
<tr>
<td>DiffusionNet - HKS</td>
<td>7.9</td>
<td>23</td>
</tr>
<tr>
<td>DiffusionNet - WKS</td>
<td>4.2</td>
<td>24.1</td>
</tr>
<tr>
<td>DiffusionNet - SHOT</td>
<td>7.2</td>
<td>4.1</td>
</tr>
<tr>
<td>DiffusionNet - PCH</td>
<td>11.4</td>
<td>8.7</td>
</tr>
<tr>
<td>DiffusionNet - PCN</td>
<td>20.4</td>
<td>9.1</td>
</tr>
<tr>
<td>DiffusionNet - VADER (ours)</td>
<td><strong>4.1</strong></td>
<td><strong>3.9</strong></td>
</tr>
</tbody>
</table>

Table 7. Human shape segmentation on the dataset of [11]. Our VADER achieves the state-of-the-art performance among methods that do not perform post-processing and evaluate on the full shape resolution. The reported numbers are the mean and standard deviation of the accuracy over five runs initialized randomly.

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy ± s.d</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCNN [12]</td>
<td>86.4%</td>
</tr>
<tr>
<td>ACNN [1]</td>
<td>83.7%</td>
</tr>
<tr>
<td>Toric Cover [11]</td>
<td>88.0%</td>
</tr>
<tr>
<td>PointNet++ [17]</td>
<td>90.8%</td>
</tr>
<tr>
<td>MDGCNN [16]</td>
<td>88.6%</td>
</tr>
<tr>
<td>DGCNN [20]</td>
<td>89.7%</td>
</tr>
<tr>
<td>SNGC [6]</td>
<td>91.0%</td>
</tr>
<tr>
<td>CGConv [21]</td>
<td>89.9%</td>
</tr>
<tr>
<td>DiffusionNet - XYZ</td>
<td>91.9 ± 0.27%</td>
</tr>
<tr>
<td>DiffusionNet - HKS</td>
<td>91.5 ± 0.21%</td>
</tr>
<tr>
<td>DiffusionNet - WKS</td>
<td>91.8 ± 0.33%</td>
</tr>
<tr>
<td>DiffusionNet - SHOT</td>
<td>91.5 ± 0.77%</td>
</tr>
<tr>
<td>DiffusionNet - PCH</td>
<td>85.6 ± 0.75%</td>
</tr>
<tr>
<td>DiffusionNet - PCN</td>
<td>87.3 ± 0.57%</td>
</tr>
<tr>
<td>DiffusionNet - VADER (ours)</td>
<td><strong>92.4 ± 0.25%</strong> (+0.9)</td>
</tr>
</tbody>
</table>

Figure 11. Qualitative evaluation of RNA segmentation on the dataset of [16]. Left: ground truth. Right: prediction by DiffusionNet + VADER.

B.2. Human Shape Matching

In Sec. 5.1 of the main text, we performed unsupervised shape matching on the FAUST-Remeshed (FR), SCAPE-Remeshed (SR), and SHREC’19 datasets (SH) and reported the matching performance in Tab. 1. We provide additional quantitative results of the FR-SR and SR-FR settings in Tab. 6. Compared with the baseline features, our VADER has the best and most consistent performance in both settings.

B.3. Molecular Surface Segmentation

In Fig. 11, we show qualitative results of RNA segmentation using DiffusionNet + VADER. It can be seen that the challenging RNA molecules can be robustly segmented into functional components with our pre-trained features.

B.4. Human Shape Segmentation

We performed an additional experiment on the human shape segmentation task. We used the dataset introduced in [11], which combines segmented human models taken from a variety of existing datasets. We used the same train/test split of 380 training and 18 test shapes as in prior works. We compared our VADER only with methods that used the original evaluation protocol as in [11], i.e., without using post-processing and evaluating the results on the full shape resolution (techniques such as Mesh Walker [8] are thus excluded).

We ran each experiment five times and report the mean and standard deviation of the accuracy in Tab. 7. Our VADER features achieve an accuracy of $92.4 \pm 0.25\%$, the state-of-the-art result on this dataset. In Fig. 12, we present qualitative results of human segmentation using DiffusionNet + VADER. Note that the segmentation results are simply the network predictions, and we do not perform any complex post-processing to the segmentation.

Figure 12. Qualitative evaluation of human shape segmentation on the dataset of [11]. Left: ground truth. Right: prediction by DiffusionNet + VADER.
B.5. Robustness to Noise

We performed an additional experiment to evaluate the robustness of our features to noise. For this, we followed the same setup as in Sec. 5.1 of the main text and in Appendix B.2, by performing unsupervised learning on FR and testing on SR with an increasing amount of noise as input. We compared our method to the best three competing features. The results are shown in Fig. 13 - left. It can be seen that our features are more robust to noise, i.e., the performance does not vary much with different noise levels (the intensity of the noise can be seen in Fig. 13 - right), which is not the case with other features, such as SHOT, whose performance degrades very quickly.

B.6. Convergence Speed

In our experiments, we observed that our VADER descriptors take less time to train and facilitate learning. To demonstrate this, we show in Fig. 14 the evolution of validation accuracy during learning of the RNA segmentation task (Sec. 5.2 of the main text). It can be seen that compared to the other features, VADER requires far fewer training iterations to achieve similar performance. This clearly indicates the better descriptiveness and generalizability of our features.

References


