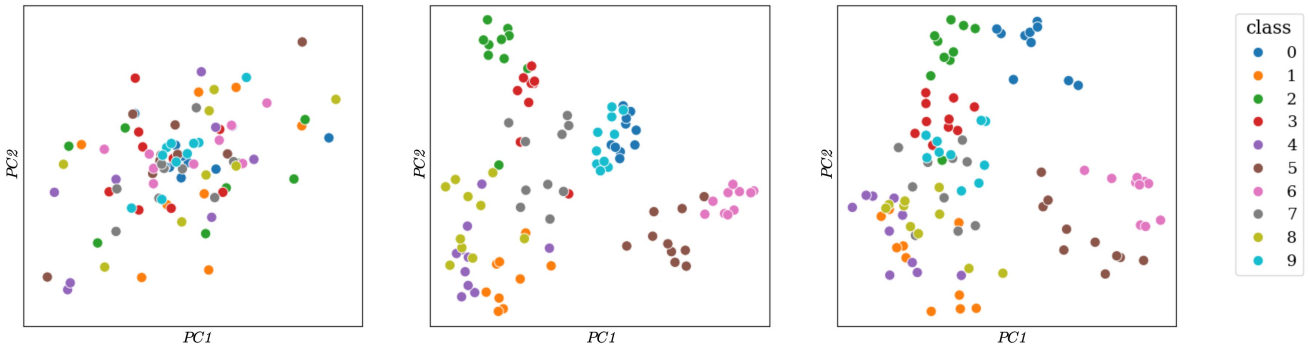
Denoising Functional Maps: Diffusion Models for Shape Correspondence

Supplementary Material



a) C_{1T} before Sign Correction b) C_{1T} after Sign Correction c) Conditioning y

Figure 4. For the FAUST [7] dataset, we plot the first two PCA components for: a) the functional maps between each shape and the template before sign correction, b) the functional maps after sign correction, c) the projection matrix after sign correction. Each entry is colored according to its shape class. Note the clusters that form for the functional maps and projection matrices after sign correction.

# Ev	F_r	S_r	S'19_r	F_a	S_a
32	98.8	99.3	98.9	98.7	99.0
64	98.7	97.9	98.1	98.2	97.8
96	98.7	94.9	96.3	95.1	94.7

Table 5. Mean Sign Correction Accuracy in % for each evaluation dataset, averaged over 100 epochs.

This supplementary material presents several components of our work that were not included in the main text:

- Evaluation of the sign correction network (Sec. A)
- Additional implementation details (Sec. B)
- A more thorough comparison with the baselines, ablation studies, and qualitative examples (Sec. C)
- Pseudocode of all algorithms (Sec. D)

A. Sign Correction

This section provides a quantitative and qualitative evaluation of the sign correction network. This component of our model allows to select a specific sign for eigenvectors of the Laplacian by projecting them onto the learned feature vectors and making the projection positive (Sec. 4.2). After performing the sign correction on all eigenvectors, we obtain a specific basis, independent of the initial one provided by the numerical eigensolver.

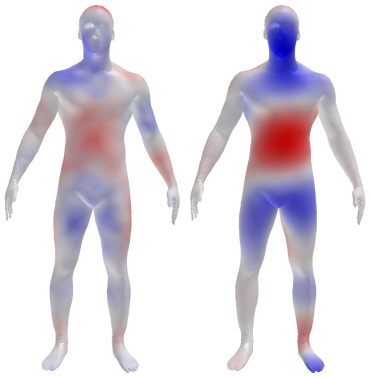
A.1. Accuracy

We first quantitatively evaluate the sign correction network on the human datasets considered in Sec. 5.2-5.3. We use a metric that we call the Mean Sign Correction Accuracy: For each mesh, we calculate the 32, 64, and 96-dimensional eigenbasis twice, perform the sign correction (Eq. 3), and report the mean number of equal eigenvectors. The metric is averaged over 100 epochs for each dataset. The results are shown in Table 5: We observe that the trained sign corrector achieves a high accuracy of $>95\%$ on all datasets.

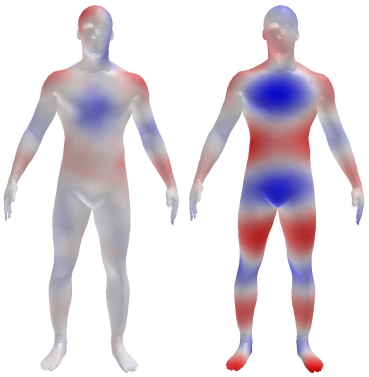
A.2. Distribution of Functional Maps

Here we study the change in the distribution of functional maps after sign correction. We use the FAUST [7] dataset, which contains meshes of 10 classes of humans with various body types in different poses. For each mesh, we obtain a 32-dimensional eigenbasis and correct it with a pretrained sign correction network. We then compute the template-wise functional map and the conditioning matrix y (Eq. 7).

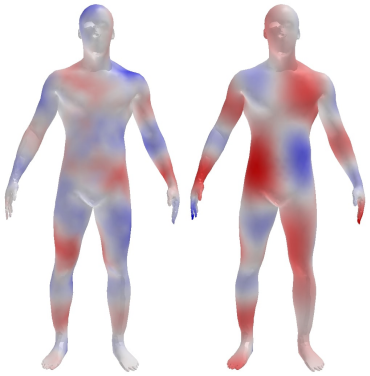
Next, we perform PCA decomposition [2] on the functional maps before and after sign correction, as well as on the conditioning matrix. The pairwise plots of the first two components are shown in Fig. 4, where each point is colored according to its class. As we can see, before sign correction, the functional maps did not have a sensible distribution since each of them was defined in a random basis. After resolving the sign ambiguity, the functional maps form noticeable clusters grouped by class. For the conditioning matrix, the overall structure resembles the distribution of functional maps after sign correction. This explains the basic principle of our method: the sign correction network transformed the space of functional maps into a learnable distribution by making them refer to a specific basis, and the denoising diffusion model approximated this distribution.



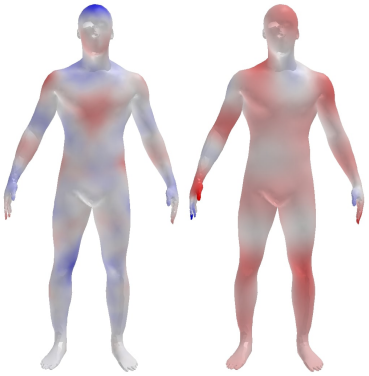
(a) Learned feature vector ς_{24} (left) and the corresponding eigenvector ϕ_{24} (right). The projection is 0.51, so we keep the sign of the eigenvector.



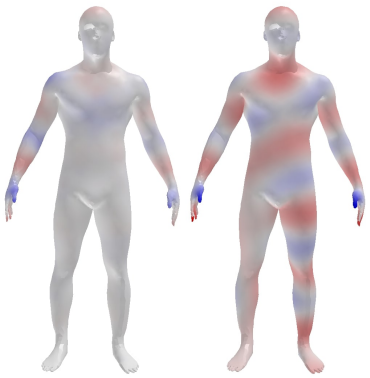
(b) Same for ς_{32} and ϕ_{32}. The projection is 0.24.



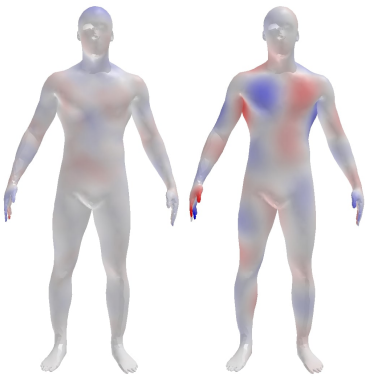
(c) Same for ς_{40} and ϕ_{40}. The projection is -0.18, so the eigenvector should be multiplied by -1.



(d) Same for ς_{48} and ϕ_{48}. The projection is -0.35, so the eigenvector should be multiplied by -1.



(e) Same for ς_{56} and ϕ_{56}. The projection is 0.69.



(f) Same for ς_{64} and ϕ_{64}. The projection is -0.29, so the eigenvector should be multiplied by -1. Note the weights on the left hand (from our point of view).

Figure 5. Learned feature vectors and their corresponding eigenvectors. Positive and negative values are shown in red and blue, respectively. The low-order feature vectors resemble the eigenvectors themselves, while the high-order ones are mainly concentrated in the arm and hand regions.

A.3. Learned Features

We visualize several of the learned feature vectors in Fig. 5, along with the corresponding eigenvectors. As we can see, the low-order feature vectors resemble the eigenvectors themselves, while the high-order ones are mostly concentrated in the arm and hand regions.

B. Additional Implementation Details

B.1. Robustness to Basis Ambiguity

To make the sign correction network applicable to high-order eigenvectors, we need to make it robust to possible basis ambiguity that arises when an eigenvalue has high multiplicity (see Sec. 3.2). Numerically, an eigenvalue with multiplicity $d > 1$ is represented as d adjacent eigenvalues $\lambda_i \dots \lambda_{i+d}$ with close values.

A straightforward way to account for the basis ambiguity is to use a single feature vector ς_i for several adjacent eigenvectors $(\phi_i, \phi_{i+1} \dots)$ instead of only one eigenvector ϕ_i. In practice, we split the eigenvectors into groups of 32 and select increasingly more eigenvectors per feature vector for each group: one for $\phi_1 - \phi_{32}$, two for $\phi_{33} - \phi_{64}$, and four for $\phi_{65} - \phi_{96}$. This takes into account the fact that high-order eigenvalues are more likely to have high multiplicity.

B.2. Training the Sign Corrector

The training process described in Sec. 4.2.1 is based on correcting the signs of two sets of eigenvectors Φ_1 and Φ_2 on the mesh S, which can be obtained by performing eigendecomposition with a numerical solver twice. Due to the sign ambiguity, the difference between them can be written as $\Phi_2 = \Phi_1 \sigma$, where $\sigma \in \{-1, 1\}^n$ is the ground-truth sign difference. However, we do not need to perform the costly eigendecomposition at each training iteration. Instead, we can compute the eigenbasis Φ once, and during training randomly sample two sign combinations $\sigma_1, \sigma_2 \in \{-1, 1\}^n$ to obtain new basis combinations $\Phi_1 = \Phi\sigma_1, \Phi_2 = \Phi\sigma_2$. This significantly reduces the training time.

B.3. Map Selection

Here, we provide more details about our map selection step (Sec. 4.1.1). We repeat the denoising process n times and rank the candidate maps based on their Dirichlet energy (i.e. map smoothness). If we simply report the map with the lowest Dirichlet energy, it may be prone to outliers such as smooth maps with flipped left-right symmetry. To address this, we select k maps with the lowest Dirichlet energy and iterate over each point on the source shape to combine them. At each step, we have k candidate matching points on the target shape, potentially containing outliers.

We find the *medoid* match (the one that has the lowest total distance to the other $k - 1$ candidates) and report it; see Fig. 6 for an illustration. After repeating these steps for each point on the source shape, we construct a map that we call the "Dirichlet Medoid". In practice, we sample a total of $n = 128$ raw maps and use $k = 16$ for the medoid selection.

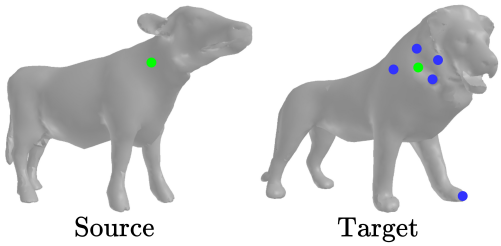


Figure 6. Illustration of the Dirichlet Medoid map selection. Given $k = 6$ candidate matches on the target shape, we report the one that has the lowest total distance to the other $k - 1$ candidates (shown in green).

Test	F_r	S_r	S'19_r
GeomFMaps	1.9	2.4	7.5
ConsistentFMaps	2.2	2.3	4.3
DiffZO	1.9	**2.2**	3.6
ULRSSM	**1.6**	**2.1**	4.6
SSL	2.0	3.1	4.0
SmS (with align.)	2.7	2.5	**3.2**
SmS (no align.)	3.4	3.2	4.0
Ours – 64×64	1.8	2.3	**3.5**
Ours – 96×96	**1.7**	**2.1**	3.9

Table 6. Comparison with descriptor-based methods trained on FAUST+SCAPE. The **best** and **2nd best** results are highlighted. SmS [12] requires rigid pre-alignment of training and test shapes to the same orientation, so we evaluated it on both aligned and unaligned datasets. Note that our method is fully intrinsic and insensitive to rigid orientation.

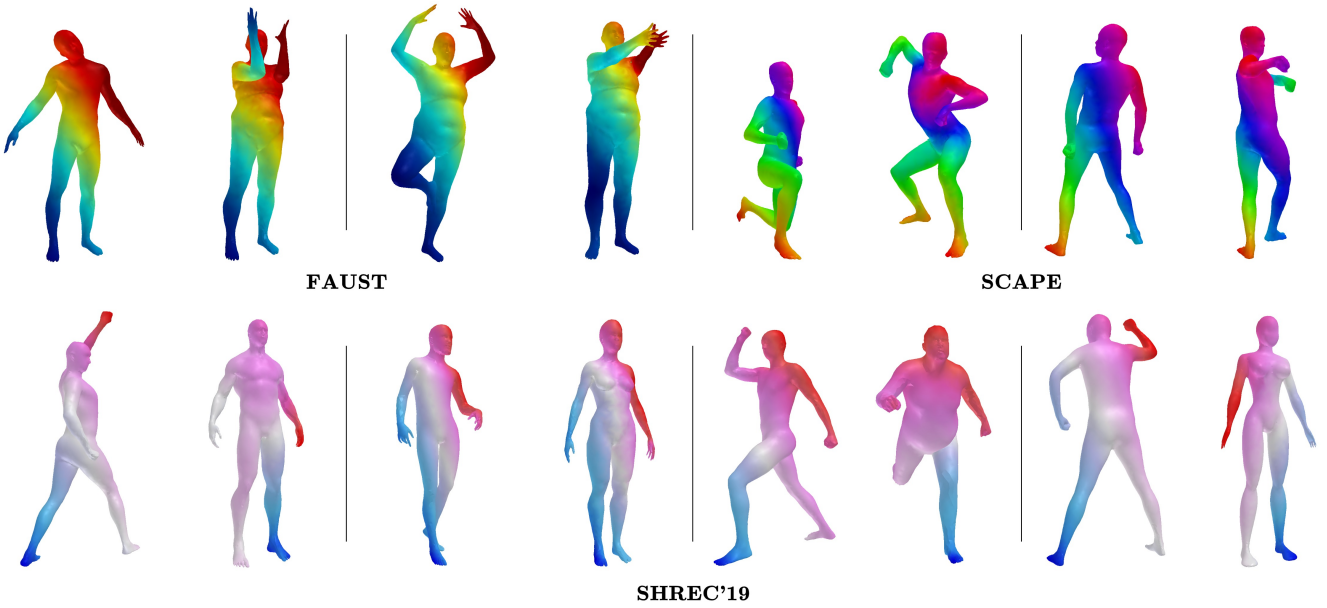
C. Additional Evaluation

C.1. Qualitative Examples

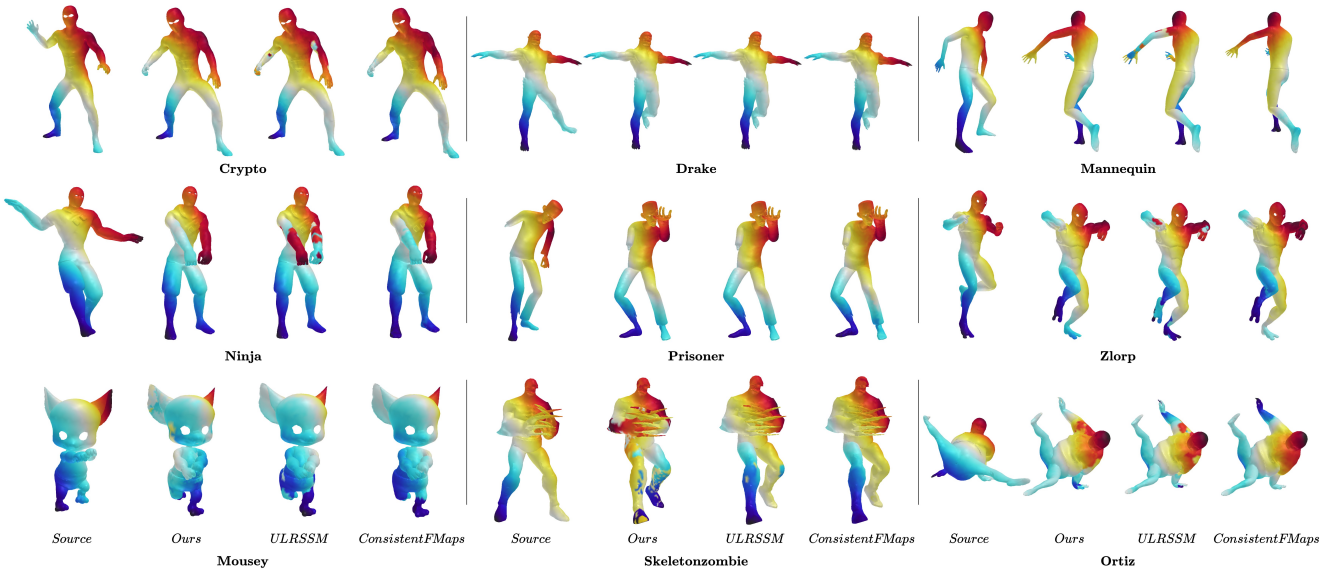
Fig. 7a shows qualitative examples of correspondences obtained by our method on FAUST, SCAPE, and SHREC'19 datasets. Fig. 7b qualitatively compares our model with two baselines on all shape classes from the DT4D dataset. Our method obtains smooth and accurate correspondences for human-like shapes with diverse body types and poses, but it is not as accurate on classes that are too different from the training data.

C.2. Baselines Trained on FAUST and SCAPE

Previously we evaluated descriptor-based methods that were trained on the FAUST dataset since it contains diverse SMPL-like shapes, while SCAPE is too small to fine-tune



(a) Examples on human shapes from the FAUST, SCAPE, and SHREC'19 datasets.



(b) Examples on the DT4D intra-category dataset, compared to ULRSSM [11] and ConsistentFMaps [69], state-of-the-art descriptor-based approaches. Our method outperforms them on most shape classes, except for a few categories that are too different from the training data: "Mousey", "Skeletonzombie", "Ortiz". The correspondences for the "Drake" class are mostly correct but sometimes have flipped left-right symmetry.

Figure 7. Additional qualitative examples. Our method provides correct correspondences for challenging human and humanoid meshes.

Test	F	S	S'19
GeomFMaps	1.9	2.5 ↑	7.1 ↓
ULRSSM	1.6	2.5 ↑	4.8 ↑
SSL	2.0	3.2 ↑	4.4 ↑

Table 7. Results of descriptor-based methods trained on 5,000 shapes from SURREAL. Arrows indicate the change in error compared to FAUST+SCAPE (see Table 6).

	64×64		96×96	
	no ZO	**Ours**	**no ZO**	**Ours**
F	2.0	1.8	1.8	1.7
S	2.6	2.3	2.2	2.1
S19	4.1	3.5	4.1	3.9

(a) Results on FAUST, SCAPE, and SHREC'19 datasets.

	32×32	
	no ZO	**Ours**
manneq.	2.0	1.0
zlorp	2.2	1.1
crypto	2.3	1.1
prisoner	3.9	1.1
ninja	2.7	1.4
ortiz	19.5	9.3
mousey	18.1	10.2
drake	12.5	10.6
skeletonz.	33.2	16.3

(b) Results on the DT4D dataset.

Table 8. Results of our models with and without ZoomOut [44] refinement.

large models (Sec. 5.2). For a complete comparison, here we additionally evaluate baselines trained on both FAUST and SCAPE. The results are shown in Table 6. As we can see, our method still performs *on par* with state-of-the-art methods, even without additional variety provided by the shapes from SCAPE.

C.3. Baselines Trained on SURREAL

Here, we study the performance of other functional map-based methods with respect to dataset size. We trained several baselines on 5,000 randomly selected shapes from SURREAL and evaluated them. As shown in Table 7, the large dataset size negatively impacts the performance of previous works, with slightly worse results compared to training on FAUST and SCAPE.

Additionally, we note that DiffusionNet-based methods require storing the full decomposition of the Laplacian, which does not scale well to large datasets. For the full SURREAL dataset, this would require about 1 TB of space. Instead, our model only stores the conditioning matrices and the functional maps, which takes at most 24 GB.

Class	Intra			Inter		
	32	**64**	**96**	**32**	**64**	**96**
manneq.	1.0	1.0	1.0	3.3	3.2	3.2
zlorp	1.1	1.1	1.1	4.2	**3.6**	4.4
crypto	1.1	1.1	**1.0**	–	–	–
prisoner	**1.1**	2.3	9.1	29.6	28.1	34.5
ninja	**1.4**	2.0	9.3	**4.3**	5.2	9.5
ortiz	9.3	17.0	24.2	–	–	–
mousey	10.2	17.9	25.6	–	–	–
drake	10.6	**6.8**	9.4	10.4	8.5	13.2
skeletonz.	16.3	30.1	36.5	49.4	31.5	39.3

Table 9. Results of our 32×32, 64×64, and 96×96 models on shape classes from DT4D-H.

Ablation Setting	SHREC'19
w/o basis ambiguity (Sec. B.1)	5.8
w/o ZoomOut	4.1
w/o any map selection (Sec. 4.1.1)	4.7
w/o medoid map selection (Sec. B.3)	3.7
Ours	**3.5**

Table 10. Ablation study of the 64×64 model on SHREC'19.

C.4. Impact of ZoomOut

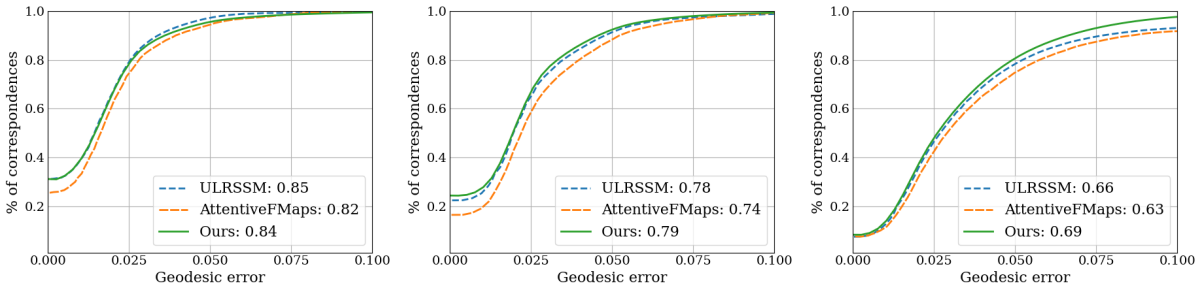
To illustrate the role that ZoomOut refinement [44] plays in our pipeline, we tested our models without the spectral upsampling step. The results are shown in Tables 8a-8b. As we can see, ZoomOut plays a key role for 32×32 and 64×64 models, while it is less important for the 96×96 one.

C.5. DT4D: Impact of the Model Dimension

Our previous experiments show that models predicting high-dimensional functional maps achieve higher accuracy, while low-dimensional ones provide better generalization to unseen shape classes. To illustrate this, we show in Table 9 the results of all our models on shape classes from DT4D. As we can see, the 32×32 model achieves excellent performance on most shape classes, while the 96×96 one shows high accuracy on a few specific classes but fails on others. The 64×64 model falls in between.

C.6. Ablation Study

Next, we ablate different parts of our method. We use a 64×64 model and the SHREC'19 dataset since it provides the largest shape diversity; see Table 10 for the results. As we can see, addressing the basis ambiguity (Sec. B.1) is crucial for the matching performance. Second, using ZoomOut [44] to increase the resolution of the functional map helps to improve the accuracy. Finally, using map selection criteria based on Dirichlet energy is also a necessary component of our model, and selecting the medoid among several smoothest maps (rather than using the map with the lowest Dirichlet Energy) helps to filter out the outliers.



(a) FAUST (b) SCAPE (c) SHREC'19

Figure 8. PCK curves for the evaluation datasets. Our method demonstrates superior or similar performance compared to existing approaches.

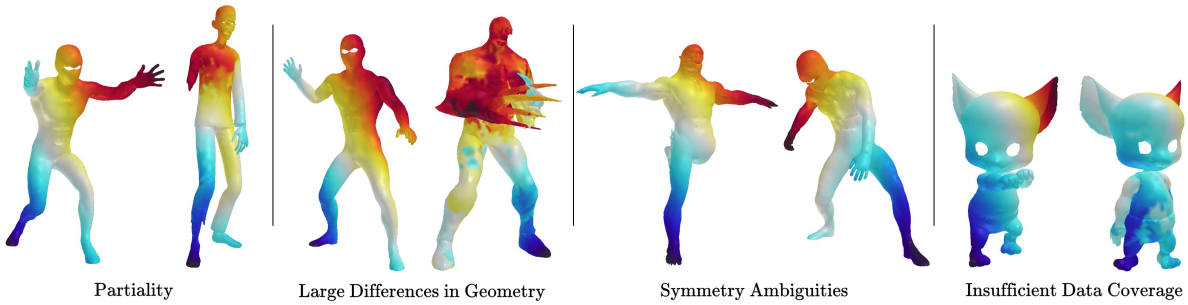


Figure 9. Limitations of our method. They include shapes with large differences from the synthetic human data, as well as topological inconsistencies like missing body parts. These limitations can be addressed in future work.

Class	Intra			Inter		
	Ours	[11]		Ours	[11]	
		FT	no FT		FT	no FT
manneq.	1.0	1.0	3.4	**3.3**	4.0	11.3
zlorp	**1.1**	1.3	5.2	**4.2**	6.9	21.3
crypto	**1.1**	1.5	3.3	–	–	–
prisoner	1.1	1.1	1.4	29.6	20.5	33.0
ninja	**1.4**	5.6	11.7	**4.3**	16.2	30.6
ortiz	9.3	**4.4**	7.8	–	–	–
mousey	10.2	**2.7**	5.2	–	–	–
drake	10.6	**1.0**	2.6	10.4	8.2	26.4
skeletonz.	16.3	**1.4**	4.7	49.4	36.5	55.1

Table 11. Results of our method on shape classes from DT4D [40] compared to ULRSSM [11] with and without fine-tuning.

C.7. Comparison with ULRSSM on DT4D

In Sec. 5.4, we evaluated our method on shape classes from the DT4D dataset against the baselines, ConsistentFMaps [69] and ULRSSM [11]. Compared to other descriptor-based methods, ULRSSM fine-tunes the parameters of the feature extractor by backpropagating the unsupervised loss during test time for each evaluation pair. For a complete comparison, in Table 11 we evaluate ULRSSM without refinement, which leads to worse results of this baseline.

C.8. PCK Curves

We show PCK curves for the FAUST, SCAPE, and SHREC'19 datasets in Fig. 8, which plot the percentage of correctly predicted keypoints within different distance thresholds from the ground truth. Our method demonstrates competitive performance compared to ULRSSM [11] and AttentiveFMaps [33].

C.9. Limitations

In Fig. 9, we illustrate the main limitations of our method. These include shapes with significant differences from the synthetic human data we used for training, as well as shapes with missing body parts. In these cases, our method tends to output maps with incorrect left-right symmetry. These limitations may be addressed in future work.

D. Algorithms

We additionally provide the algorithms discussed in Sec. 4 as pseudocode. The training and inference of the diffusion model are described in Alg. 1 and Alg. 2, respectively. The sign correction is described in Alg. 3 and the training process in Alg. 4.

Algorithm 1: DDPM Training Pipeline

Train the Sign Correction Network Θ;
Correct all eigenvectors Φ with Θ in the dataset;
Get functional maps to the template, C_{1T};
Calculate conditioning y;
Train a DDPM to predict C_{1T} given y;

Algorithm 2: DDPM Inference with Selection

Input: Shape collection $\{S_i\}$, List of Test Pairs $\mathcal{L}$, Sign Correction Net Θ, DDPM

`// Template Stage`
for *shape S, eigenvectors* Φ *in* $\{S_i\}$ **do**
  repeat *n = 128 times* `// See 4.1.1`
    Correct the signs of Φ, get conditioning y;
    $C_{1T} \coloneqq$ **DDPM** (y);
    Convert C_{1T} to pointwise map Π_{T1};
end
`// Pairwise Stage`
for *shapes* S_1, S_2 *in* $\mathcal{L}$ **do**
  $\mathcal{T}_\Pi \coloneqq \{\}$;
  repeat *n = 128 times* `// See 4.1.1`
    $C_{12} \coloneqq$ **LstSq** $(\Pi_{T2}\Phi_2, \Pi_{T1}\Phi_1)$;
    Zoomout C_{12} to $[200, 200]$;
    Convert C_{12} to pointwise map Π_{21};
    Add Π_{21} to $\mathcal{T}_\Pi$;
  $\hat{\Pi}_{21} \coloneqq$ **Selection**$(\mathcal{T}_\Pi)$
end
Output: pointwise maps $\hat{\Pi}_{21}$ for each pair

Algorithm 3: Learned Sign Correction

Input: Shape S, eigenvectors $\Phi \sim$ (v, n), vertex-area matrix $A \sim$ (v, v), feature extractor Θ

$\Sigma \coloneqq \Theta(S)$; `// See B.1,` (v, n)
$P \coloneqq \Sigma^T A \Phi$; `//` (n, n)
if *training* **then**
  $\hat{\sigma} \coloneqq$ **Diag**(P) ; `//` (n)
else
  $\hat{\sigma} \coloneqq$ **Sign** (**Diag**(P)) ; `//` (n)
end
$\hat{\Phi} \coloneqq \Phi\hat{\sigma}$; `//` (v, n)

Output: Eigenvectors with corrected signs $\hat{\Phi}$, correcting sign sequence $\hat{\sigma}$

Algorithm 4: Unsupervised Training of Sign Correction

Input: Shape collection $\{S_i\}$, feature extractor Θ

for *N epochs* **do**
  for *shape S, eigenvectors* Φ *in* $\{S_i\}$ **do**
    `// -1 or 1`
    $\sigma_1 \coloneqq$ **RandomSigns** (n);
    $\sigma_2 \coloneqq$ **RandomSigns** (n);
    $\Phi_1 \coloneqq \Phi\sigma_1$;
    $\Phi_2 \coloneqq \Phi\sigma_2$;
    `// Alg.3`
    $\hat{\sigma}_1 \coloneqq$ **SignCorr** (S, Φ_1, Θ);
    $\hat{\sigma}_2 \coloneqq$ **SignCorr** (S, Φ_2, Θ);
    $\mathcal{L}_{\text{sign}} \coloneqq$ **MSE**$(\sigma_1\sigma_2, \hat{\sigma}_1\hat{\sigma}_2)$;
    Backpropagate$(\mathcal{L}_{\text{sign}})$
  end
end
Output: Trained feature extractor $\hat{\Theta}$