Topology-aware Feature Propagation for Unsupervised Non-rigid Point Cloud Correspondence

Supplementary Material

1. Model Design Appendix

This section introduces details about: (1) the topology confidence estimation module in Sec. 1.1; (2) the design of the basic encoder in Sec. 1.2; (3) the loss system in Sec. 1.3.

1.1. Design of Topology Confidence Estimation

We briefly introduce the keypoint estimation method KeyGrid [4], which is used as the topology confidence estimation module. Readers are recommended to read the original paper for more details since it is not our contribution.

KeyGrid is based on a UNet structure where the encoder is PointNet++ [8], and yields a keypoint confidence matrix $W_{kn} \in \mathbb{R}^{K\times N}$ to estimate the position of keypoints $P_K \in \mathbb{R}^{K\times 3}$ through the input point cloud $P_N \in \mathbb{R}^{N\times 3}$:

$$P_K = \text{Softmax}(W_{kn}) \times P_N, \tag{1}$$

where $\times$ means matrix multiplication. To restrict the position of keypoints, KeyGrid reconstructs the original P_N with the UNet decoder and residual middle features from the encoder. Notably, it further designs a grid heatmap that aids in reconstruction by providing the contained structure information. To form the grid heatmap, it first approximates the skeleton to represent the object shape using weighted connected lines between all keypoint pairs, where the weight W_{kk} indicates the confidence whether a segment line represents the shape skeleton. Next, a set of grid points $P \in \mathbb{R}^{N_x\times N_y\times N_z\times 3}$ is uniformly sampled in the normalized cubic 3D space, where N_x, N_y, N_z denote the number of points sampled on each side of the cube, respectively. Here we set $N_x = N_y = 16, N_z = 8$. The distances are calculated between the grid points and all the line segments. Formally, the distance $d_{ij}(p)$ between a grid point $p \in \mathbb{R}^3$ and a skeleton segment (k_i, k_j) connecting keypoint k_i and k_j in P_K is formulated as:

$$d_{ij}(p) = \begin{cases} \|p - k_i\|_2 & \text{if } t \le 0 \\ \|p - ((1-t)k_i + tk_j)\|_2 & \text{if } 0 < t < 0 \\ \|p - k_j\|_2 & \text{if } t \ge 1 \end{cases} \tag{2}$$

where

$$t = \frac{(p - k_i)\cdot(k_j - k_i)}{\|k_j - k_i\|_2} \in \mathbb{R}. \tag{3}$$

The grid heatmap feature for each grid point $D(p)$ is the maximum of the weighted distances from this point to all the skeleton segments:

$$D(p) = \max_{ij}\{w_{ij}\cdot \exp(d_{ij}^2(p)/\sigma^2)\}, \tag{4}$$

where w_{ij} is the elements in W_{kk}, and σ is a hyperparameter. The final grid heatmap H consists of features of all grid points in P:

$$H = (D(P)) \in \mathbb{R}^{N_x\times N_y\times N_z\times 1}. \tag{5}$$

H will be concatenated with the residual feature from the encoder for self-reconstruction. The better the reconstruction is, the more precisely the grid heatmap can represent the shape. This means the positions of P_K are semantically meaningful, and the W_{kk} represents the shape skeleton with higher confidence for line segments that adhere to the shape topology. P_K, W_{kn}, W_{kk} thus contain topology information, helping us learn the topology without annotation and guide the topology-aware propagation from local to global.

In practice, to guarantee that points in P_K belong to the original point cloud P_N, we copy the closest points of P_K in P_N to replace them. Also, we block the gradients of W_{kn}, W_{kk} when they are output by the topology confidence estimation module, so that the training of this module is only influenced by the keypoint loss mentioned in Sec. 1.3.

1.2. Design of Basic Encoder

The basic encoder comprises an equivariant local reference frame network and a module for modeling relative geometric relations.

Equivariant Local Reference Frame Network. Following EquiShape [12], we predict local reference frames (LRFs) with neural networks. Specifically, a variant of GVP-GCN [5, 9] constructs KNN [6] graphs with $k = 27$ to output groups of two rotation-equivariant vectors as point-wise features. The input features of this process are zero tensors as initial scalar and vectors from the centroid of the point cloud to each point as initial vector. These point-wise vector groups are further normalized and orthogonalized by the Gram-Schmidt process:

$$u_{i1} \leftarrow \frac{v_{i1}}{||v_{i1}||}, \tag{6}$$

$$u_{i2}^{'} \leftarrow v_{i2} - < v_{i2}, u_{i1} > u_{i1}, \tag{7}$$

$$u_{i2} \leftarrow \frac{u_{i2}^{'}}{||u_{i2}^{'}||}, \tag{8}$$

where $< \cdot, \cdot >$ is the inner product of two vectors, v_{i1}, v_{i2} are the i^{th} point's group of (V_1, V_2) in the point cloud. The two orthogonal unit vectors and their cross-product construct the orientation matrix O_i of the i^{th} point as its LRF:

$$O_i \leftarrow [u_{i1}, u_{i2}, u_{i1} \times u_{i2}], i = 1, \cdots, N, \tag{9}$$

which construct the final point-wise LRFs $O \in \mathbb{R}^{N\times3\times3}$. Rotating a local point set with O equals projecting it to its LRF, and becomes the initial point-wise feature for the following propagation. Unlike [12], we do not integrate cross-attention here, as we share information between the source and target in the geometric relation modeling module.

Relative Geometric Relation Modeling. We utilize the structure of the proposed topology-aware transformer Prop$(\cdot)$ mentioned in Sec. 3.3 of the main paper for the first-stage feature extraction, employing only the relative geometric relation matrix G without the topology weights W_{kn} and W_{kk}. Therefore, in this stage, the features are not topology-aware. Specifically, a DGCNN-based [13] feature extractor encodes point-wise features with integrated transformers at each DGCNN layer. For these transformers, the querying feature F_Q ($N \times 1 \times C$) are per-point features of P_N under LRFs (*i.e.*, F_N), the referring feature F_R ($N \times k \times C$) are the features of KNN neighbors of P_N using neighbor size $k = 27$, and forming the relative geometric relation matrix G ($N \times 1 \times k \times C$). For each point feature $F_Q^i \in \mathbb{R}^{1\times C}$, the propagation operator Prop$(\cdot)$ integrates its k-nearest neighbor features $F_R^i \in \mathbb{R}^{k\times C}$ and the associated geometric relation tensor $G^i \in \mathbb{R}^{1\times k\times C}$, producing an updated feature representation $\in \mathbb{R}^{1\times C}$. By applying this process to every point in P_N, we obtain the final output features $\in \mathbb{R}^{N\times C}$, where each point has propagated and refined its representation based on its local context.

We further extend this process to the global stage, where we conduct interactions between point-wise features and superpoint features, taking into account their relative geometric relations. The module first downsamples the original point cloud P_N level by level through the FPS algorithm [8] to super points $P_{N_l} \in \mathbb{R}^{N_l\times3}$ where $l = 0, 1, ..., S$. Each superpoint $p_s^{l+1} \in \mathbb{R}^{1\times3}$ in level $l+1$ seeks its KNN points $p_k^l \in \mathbb{R}^{k\times3}$ in level l where $k = \frac{N_l}{N_{l+1}}$. Then the feature of p_s^{l+1} fuses those of p_k^l via Prop$(\cdot)$ with relative geometric relations, where the query is superpoint features of this level and the referring is the point-wise features of the previous level, until the information from point-wise features is conveyed to the final layer of superpoints and becomes the superpoint feature $F_S \in \mathbb{R}^{N_S\times C}$. Then, point-wise features F_N are propagated to F_S step by step with the relative geometric relations between them:

$$\begin{aligned} F_{N_1} &= \text{Prop}(F_{N_1}, F_N, G_{N_1N}), \\ F_{N_2} &= \text{Prop}(F_{N_2}, F_{N_1}, G_{N_2N_1}), \\ F_S &= \text{Prop}(F_S, F_{N_2}, G_{SN_2}), \end{aligned} \tag{10}$$

Where Prop$(\cdot)$ is the propagation process mentioned in Sec. 3.3 of the main paper, F_N is the point-wise features, terms like G_{AB} is the relative geometric relations between P_A and P_B. The superpoint levels in this stage are set to 256,64,8.

The final level of superpoint features F_S further propagates intra- and inter-shape information and is propagated back to the point-wise features F_N with only the structure of the topology-aware transformer and relative geometric relations:

$$F_S = \text{Prop}(F_S, F_S, G_{SS}), \tag{11}$$
$$F_S^1 = \text{Prop}(F_S^1, F_S^2, G_{SS}^{12}), \tag{12}$$
$$F_N = \text{Prop}(F_N, F_S, G_{NS}), \tag{13}$$
$$F_N = \text{Prop}(F_N, F_N, G_{NN}), \tag{14}$$

where G_{SS} are the relative geometric relations between each point pair in P_S, G_{NS} are the relative geometric relations between P_S and P_N, G_{NN} are the relative geometric relations between point pairs within P_N. To propagate information of the target superpoints to source superpoints, $F_S^1 = F_S^{src}$, $F_S^2 = F_S^{tgt}$, G_{SS}^{12} is the relative geometric relation between superpoints P_S^{src} and P_S^{tgt}, vice versa.

The final $F_S \in \mathbb{R}^{K\times C}$ as the output of the basic encoder further undergoes the downsampling and compression process mentioned in Eq. 10 from F_N to yield the superpoint feature of key points $F_K \in \mathbb{R}^{K\times C}$ anchored at $P_K \in \mathbb{R}^{K\times3}$.

1.3. Detailed Loss System

To train the correspondence pipeline, we follow previous unsupervised work [7] to use surrogate reconstruction loss. For convenience, we represent the source and target point clouds P_{src}, P_{tgt} with $X, Y \in \mathbb{R}^{N\times3}$, respectively.

Chamfer Distance. Chamfer distance is the average square distance between a point of one point cloud and its nearest point in the other point cloud. Given two point clouds X, Y, the chamfer distance loss is defined as:

$$\begin{aligned} \mathcal{L}_{\text{CD}} = &\frac{1}{\mid X \mid} \sum_{x_i\in X} \min_{y_j\in Y} d(x_i - y_j) \\ &+ \frac{1}{\mid Y \mid} \sum_{y_j\in Y} \min_{x_i\in X} d(x_i - y_j), \end{aligned} \tag{15}$$

where $d(\cdot)$ is the distance function, which we use the L_2 norm.

Feature Similarity. Given the point-wise features $f_X, f_Y \in \mathbb{R}^{N\times C}$ from the pipeline, the feature similarity is calculated by the cosine similarity:

$$s_{ij} = \frac{f_X^i \cdot (f_Y^j)^T}{||f_X^i||_2 ||f_Y^j||_2}, \tag{16}$$

where f_X^i, f_Y^j are the features of the i^{th} and j^{th} point in X, Y, respectively. $(\cdot)^T$ denotes the transpose operation. $S_{XY} \in \mathbb{R}^{N\times N}$ is the complete similarity matrix between X and Y where $S_{XY}^{ij} = s_{ij}$.

Cross&Self-Reconstruction. Following [7], we use cross and self-reconstruction to train our method without ground

truth annotation. For each point $x_i \in X$, a softmax is employed to normalize the similarity to its k_{cc} nearest neighbors:

$$w_{ij} = \frac{e^{s_{ij}}}{\sum_{l \in \mathcal{N}_Y(x_i)} e^{s_{il}}}, \tag{17}$$

where $\mathcal{N}_Y(x_i)$ is the k_{cc} indices of x_i's latent nearest neighbors in Y. In practice, k_{cc} is set to 10. Then, an approximate matching point $\hat{y}_{x_i}$ is calculated as a weighted summation of its neighbors' coordinates:

$$\hat{y}_{x_i} = \sum_{j \in \mathcal{N}_Y(x_i)} w_{ij} y_j. \tag{18}$$

The cross-reconstruction of Y by X is denoted as $\hat{Y}_X \in \mathbb{R}^{N \times 3}$, where $\hat{Y}_X^i = \hat{y}_{x_i}$. Similarly, the cross-reconstruction of X by Y is denoted as $\hat{X}_Y \in \mathbb{R}^{N \times 3}$.

Self-reconstruction is employed in a similar way to cross-reconstruction, where each point $x_i \in X$ is approximated by its $k_{sc} = 10$ neighboring latent points:

$$\hat{x}_i = \sum_{l \in \mathcal{N}_X(x_i)} w_{il} x_l, \tag{19}$$

where $\mathcal{N}_X(x_i)$ is the latent self-neighborhood of x_i in X. The similarity in Eq. 16 is measured between f_X^i and f_X^l instead of f_X^i and f_Y^j. w_{il} is computed as in Eq. 17 with $\mathcal{N}_X(x_i)$ instead of $\mathcal{N}_Y(x_i)$. The final self-reconstruction of X by X and Y by Y are denoted as $\hat{X}_X$ and $\hat{Y}_Y \in \mathbb{R}^{N \times 3}$.

Reconstruction Loss. We adopt construction loss following the baseline method [7]. The reconstructed point clouds $\hat{Y}_X$,$\hat{X}_Y$,$\hat{X}_X$,$\hat{Y}_Y$ are constrained with the construction loss formulated as:

$$\begin{aligned} \mathcal{L}_{cc} &= \lambda_{cc}(\mathcal{L}_{\text{CD}}(Y, \hat{Y}_X) + \mathcal{L}_{\text{CD}}(X, \hat{X}_Y)), \\ \mathcal{L}_{sc} &= \lambda_{sc}(\mathcal{L}_{\text{CD}}(Y, \hat{Y}_Y) + \mathcal{L}_{\text{CD}}(X, \hat{X}_X)), \end{aligned} \tag{20}$$

where λ_{cc} and λ_{sc} are hyperparameters controlling the weight of different reconstruction operations, $\mathcal{L}_{\text{CD}}$ stands for chamfer distance loss mentioned above.

Neighborhood Loss. Neighborhood loss $\mathcal{L}_{\text{neigh}}$ is utilized to regularize the spatially close points in the source point cloud to remain spatially close when mapped to points in the target shape:

$$\mathcal{L}_{\text{neigh}} = \frac{1}{nk_m} \sum_i \sum_{l \in N_X(x_i)} v_{il} ||\hat{y}_{xi} - \hat{y}_{xl}||_2^2, \tag{21}$$

where $N_X(x_i)$ represents $k_m = 10$ nearest neighbors of x_i in X measured by Euclidean distance. v_{il} is a distance-aware weight defined by:

$$v_{il} = e^{-||x_i - x_l||_2^2/\alpha}, \tag{22}$$

where α is the hyperparameter set to 8 in practice.

Keypoint Loss. Following [4], the keypoint estimation network is trained by a similarity loss and a farthest point keypoint loss. The similarity loss $\mathcal{L}_{\text{sim}}$ is the Chamfer Distance between the input point and the reconstructed point cloud. The farthest point keypoint loss $\mathcal{L}_{\text{far}}$ calculates the Chamfer Distance between K keypoints and $J = K + 4$ FPS downsampled points to restrict the distribution of keypoints to cover the whole shape. The overall loss $\mathcal{L}_{\text{key}}$ is defined as:

$$\mathcal{L}_{\text{key}} = \lambda_{\text{sim}} \mathcal{L}_{\text{sim}} + \lambda_{\text{far}} \mathcal{L}_{\text{far}}, \tag{23}$$

where $\lambda_{\text{sim}} = \lambda_{\text{far}} = 1$ are scalar loss coefficients.

Codebook Loss. Following [10], the codebook loss $\mathcal{L}_{vq}$ consists of a commitment loss with an orthogonal regularization term. Specifically, given input feature F, the representative vectors F_{vq} are chosen via cosine similarity from the stored representative vectors F_{code}, and attach the back-propagated gradient by:

$$F_{vq} = F + \text{detach}(F_{vq} - F), \tag{24}$$

where detach$(\cdot)$ operation blocks the gradient inside the term. The codebook loss is formulated as:

$$\mathcal{L}_{vq} = \text{MSE}(\text{detach}(F_{vq}), F) + \lambda_{orth} \text{Cossim}(F_{\text{code}}), \tag{25}$$

where MSE(a, b) calculates the mean square error loss between a, b. This allows the gradient of the main correspondence loss (*i.e.*, $\mathcal{L}_{cc}, \mathcal{L}_{sc}, \mathcal{L}_{\text{neigh}}$) can be passed to the input F without influencing the training of the codebook. Cossim(F_{code}) calculates the mean cosine similarity within vectors stored in the codebook, and $\lambda_{orth} = 0.001$ in practice. Reducing cosine similarity enforces the vectors in the codebook to be orthogonal. We update the codebook with exponential moving averages (EMA) [11] with decay=0.8. During training, the codebook actively replaces any codes that have an exponential moving average cluster size less than 2.

Overall Loss. The final loss for our optimization process is formulated with all the losses mentioned above:

$$\mathcal{L}_{\text{total}} = \lambda_{cc} \mathcal{L}_{cc} + \lambda_{sc} \mathcal{L}_{sc} + \lambda_m \mathcal{L}_{\text{neigh}} + \lambda_{vq} \mathcal{L}_{vq} + \lambda_{\text{key}} \mathcal{L}_{\text{key}}, \tag{26}$$

where $\lambda_{cc}, \lambda_{sc}, \lambda_m$ are hyperparameters mediating the significance of different loss terms. We set $\lambda_{cc} = 1, \lambda_{sc} = 10, \lambda_m = 1, \lambda_{vq} = 1, \lambda_{\text{key}} = 1$ in practice.

1.4. Extended Ablation

In this section, we present extended ablations of our pipeline, including topology and attention types, and using MLP for topology learning.

Ablation on GT, KNN, and Keypoint-based Topology. We create a pseudo-GT skeleton by manually annotating $K = 16$ keypoints per SHREC'19 shape, as shown in Fig. 2. Boolean topology weights are defined as: $W_{kk}^{ij} = 1$

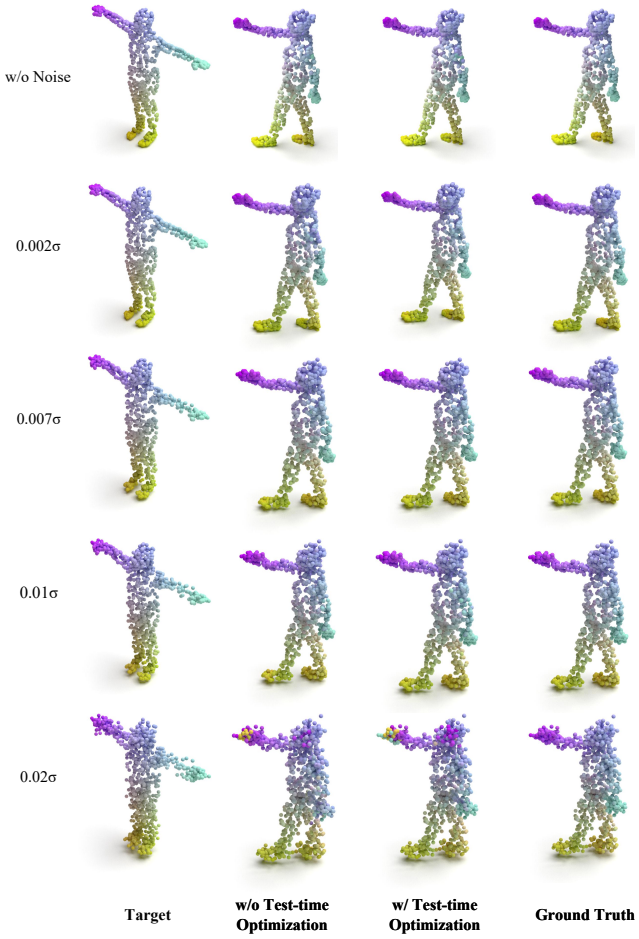

Figure 1. **Correspondence Examples for Noisy Data**. The model trained on SURREAL without noise is directly evaluated on SHREC'19 with different levels of added Gaussian noise. The standard deviation of noise is indicated by a variety of coefficients plus σ.

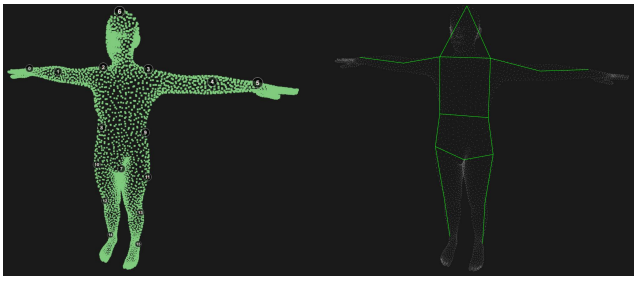

Figure 2. **Visualization of Pseudo-GT Skeleton.** Left: Positions of the selected keypoints; Right: The manual pseudo-GT skeleton.

for connected skeleton keypoints; $W_{kn}^{ij} = 1$ if keypoint i is nearest to point j. Additionally, we construct a KNN skeleton from FPS-sampled superpoints (each connected to its two nearest superpoints) and yield W_{kk}^{knn}. As shown in Tab. 1, training with the GT skeleton yields acc: 25.5%, err: 5.0 on SHREC'19, demonstrating the potential of topology-aware propagation for correspondence, while our method (acc: 20.8%, err: 6.2) outperforms the KNN-skeleton variant (acc: 18.2%, err: 8.3), confirming the efficacy of our learned topology.

Table 1. **Method using GT, KNN, and Keypoint-based Topology.** We evaluate our pipeline with pseudo-GT, KNN-based, and keypoint-based skeletons as topology. The improvements brought by the pseudo-GT skeleton demonstrates the potential of topology-aware propagation for correspondence, while our method (Keypoint-based skeleton) outperforms the KNN-skeleton variant, confirming the efficacy of our learned topology.

Topology Type	SURREAL	
	acc ↑	err ↓
KNN-based	18.2%	8.3
Keypoint-based	20.8%	6.2
Pseudo-GT	25.5%	5.0

Ablation on Topology-aware Attention. The topology matrix T serves as an integrated feature with a different aggregation scheme than attention masking. Unlike conventional masked attention reweights attention score via softmax (attn_weights · mask), our method propagates T into the output: output $\leftarrow f(\text{attn\_weights}, V, T)$ (see Eq. 3 of the main paper). This leverages the inherent diversity in shape part topology to enhance feature discriminability, which we empirically find effective over conventional mask attention, as shown in Tab. 2.

Table 2. **Comparison between Conventional Mask Attention and our Topology-aware Attention.** We evaluate our pipeline with the conventional mask attention and our topology-aware attention. The results show that our topology-aware attention can better leverage the inherent diversity in shape part topology to enhance feature discriminability than the conventional mask attention, and yields better performances.

Attention Type	SURREAL	
	acc ↑	err ↓
Conventional Mask Attention	32.7%	4.3
Topology-aware Attention	**33.2%**	**4.0**

Learning Topology with MLP. We use a simple MLP to predict the topology weights W_{kn} and W_{kk}, with P_K derived from W_{kn} by Eq. 4 of the main paper. The pipeline is trained end-to-end on correspondence losses ($\lambda_{cc}\mathcal{L}_{cc} + \lambda_{sc}\mathcal{L}_{sc} + \lambda_m\mathcal{L}_{\text{neigh}}$) without gradient blocking. As shown in Tab. 3, with codebook optimization, the MLP-based pipeline achieves acc: 32.9%, err: 4.3, which is inferior to our final model (acc: 33.2%, err: 4.0) and close to the pipeline without topology weights (acc: 32.8%, err: 4.5, see (A3) in Tab. 2 of the main paper). This indicates that predicting beneficial topology for non-rigid shape corre-

spondence without supervision is under-explored, while our method proposes a beneficial learning strategy with performance improvements.

Table 3. **Learning Topology with MLP.** Predicting beneficial topology for non-rigid shape correspondence without supervision is under-explored, while our method proposes a beneficial learning strategy with performance improvements.

Topology Predictor	SURREAL	
	acc ↑	err ↓
-	32.8%	4.5
MLP	32.9%	4.3
Ours	**33.2%**	**4.0**

2. Extended Robustness Analysis

This section reports results under perturbation of the unaligned orientation (Sec. 2.1), noise (Sec. 2.2), sparsity and missing parts (Sec. 2.3), and training results with random seed variation (Sec. 2.4).

2.1. Robustness to Orientation Perturbation

Tab. 4 experiments on randomly rotating the input shapes on all four benchmarks during testing without retraining to check the robustness to initial orientations. The random rotation is achieved by rotating around the z-axis like [2] for P^{src}. The results demonstrate that without adaptation to the rotated test data, our method shows robustness to initial orientations, with slight performance differences between results under random rotation and without rotation, thanks to the topology awareness.

Table 4. **Robustness Analysis of Orientation Perturbation.** "w/o. Rot" means no random rotation during testing, and "w/. Rot" means randomly rotating the source point clouds during testing. Our method yields almost consistent performance under orientation perturbations on all four benchmarks. We report the performance with test-time optimization.

Benchmark	w/o. Rot		w/. Rot	
	acc ↑	err ↓	acc ↑	err ↓
SURREAL	37.93%	3.21	37.94%	3.19
SHREC'19	24.00%	6.39	23.96%	6.37
SMAL	60.51%	1.70	60.45%	1.70
TOSCA	71.54%	0.82	71.60%	0.81

2.2. Robustness to Noise Perturbation

We evaluate the robustness against noise perturbation of our method. We gradually add random Gaussian noise to the input shape pairs. The level multiplies the bounding box σ of the point cloud (*i.e.*, the maximum value of the point coordinates subtracts the minimum value) and becomes the adaptive standard deviation, and multiplies the random Gaussian noise to become the final noise added to the point cloud. We directly test the model trained on SURREAL without adaption to the noisy data. As shown in Fig. 1, small degrees of noise slightly influence the performance of our method. While severe noise inevitably reduces the performance, our method can still yield a certain degree of approximately reasonable correspondence.

Further, we compare the most recent open-sourced reproducible methods(SE-ORNet [2], DIFF3F [3]) with ours under various levels of Gaussian noise in Tab. 5. For all the methods that require training, we directly test the SURREAL-trained models without adaptation to noisy shapes. The results show that our method has the best acc and second-best err on each level of noise, illustrating our method has a certain degree of noise robustness.

Table 5. **Robustness to Noise Perturbation.** We evaluate the performances of our method and the most recent open-sourced reproducible methods. "Noise Level" correlates to the level of added Gaussian Noise. The results illustrate our method has a certain degree of noise robustness with the best acc and second-best err on each level of noise. We report the performance with test-time optimization.

Noise Level	SE-ORNet [2]		DIFF3F [3]		Ours	
	acc ↑	err ↓	acc ↑	err ↓	acc ↑	err ↓
0	21.5%	4.6	<u>26.4%</u>	**1.7**	**37.9%**	<u>3.2</u>
0.001σ	21.9%	4.8	<u>26.1%</u>	**1.7**	**38.7%**	<u>3.0</u>
0.0025σ	21.2%	4.6	<u>23.5%</u>	**1.7**	**36.1%**	<u>3.1</u>
0.005σ	18.2%	4.6	<u>19.1%</u>	**1.8**	**28.5%**	<u>3.7</u>
0.007σ	15.5%	4.8	<u>16.4%</u>	**1.8**	**22.4%**	<u>4.5</u>

2.3. Robustness to Sparsity and Missing Parts

We evaluate the performance of the most recent open-sourced reproducible methods (SE-ORNet [2], DIFF3F [3]) and ours on more sparse point clouds with 512 and 256 points on the SHREC'19 testset. DIFF3F specially requires dense inputs (8k points or mesh), and only selects features of 1024 points for metrics calculation, which is unfair to compare with methods that directly input downsampled points. For all the methods that require training, we directly test the SURREAL-trained models without adaptation to sparser shapes. The result in Tab. 6 shows that our method trained on SURREAL accepts any resolution with the best acc, indicating certain robustness to sparse inputs.

We also evaluate the performance under missing part perturbation. To simulate the test set with missing parts, we divide each shape in the SHREC'19 test set into 51 point clusters via the k-means algorithm [1], randomly delete 30% clusters, and randomly select 1024 points in the remaining clusters. For all the methods that require training, we directly test the SURREAL-trained models without adaptation to shapes with missing parts. The results in Tab. 6 show that compared to previous methods, our method

trained on SURREAL has the best acc under shapes with missing parts, indicating certain robustness to missing part perturbation.

Table 6. **Robustness against Sparsity and Missing Parts.** We evaluate the performance of the most recent open-sourced reproducible methods under sparsity or missing part perturbation. The results illustrate our method has a certain degree of robustness to sparsity and missing parts with the best acc compared to other methods. We report the performance with test-time optimization.

	SE-ORNet [2]		DIFF3F [3]		Ours	
	acc ↑	err ↓	acc ↑	err ↓	acc ↑	err ↓
1024 Points	21.5%	4.6	26.4%	**1.7**	**37.9%**	3.2
512 Points	23.0%	5.2	-	-	**42.9%**	**3.2**
256 Points	18.8%	**6.9**	-	-	**39.5%**	**6.9**
Missing Parts	19.2%	5.4	24.8%	**1.4**	**35.1%**	4.6

2.4. Training with Different Random Seeds

We use different random seeds to train our method on all the benchmarks and report three groups of results with their mean and variance in Tab. 7. The results show that the performances are not dramatically changed with the variation of the random seed, indicating the reproducibility of our method.

Table 7. **Results of Training with Different Random Seeds.** "Mean" and "Var" mean the average performance and variance across three groups of results. Relatively stable results with random seed variation demonstrate the reproducibility of our method. We report results with test-time optimization.

	SURREAL		SHREC'19		SMAL		TOSCA	
	acc ↑	err ↓	acc ↑	err ↓	acc ↑	err ↓	acc ↑	err ↓
A	37.5%	3.3	25.0%	6.0	61.1%	1.5	71.5%	0.3
B	37.9%	3.2	24.0%	6.4	60.5%	1.7	71.5%	0.8
C	37.6%	3.2	24.5%	6.3	60.0%	1.7	70.8%	0.2
Mean	37.67%	3.23	24.5%	6.2	60.53%	1.63	71.3%	0.43
Var	0.0433	0.0033	0.2500	0.0433	0.3033	0.0133	0.1633	0.1033

3. Visualization

This section visualizes the correspondence results of our method under deformation perturbation for both human and animal pairs in Sec. 3.1. Further, we visualize correspondence (1) on the real-scanned Owlii dataset [14] in Sec 3.2; (2) with partially degraded topology confidence in Sec. 3.3. Failure cases are shown in Sec. 3.4.

3.1. Visualization on Human and Animal Pairs

In Fig. 8, we present qualitative results of our method on human pairs, while on animal pairs in Fig. 9 and Fig. 10. "w/o Test-time Optimization" shows results without test-time optimization, while "w/ Test-time Optimization" shows results with test-time optimization. Our method can provide accurate correspondence regardless of orientation or deformation perturbation for both human and animal pairs even without test-time optimization, while further boosting performance with optimization. We further visualize the correspondence error in red in Fig. 7, which showcases that our method performs well with insignificant correspondence errors.

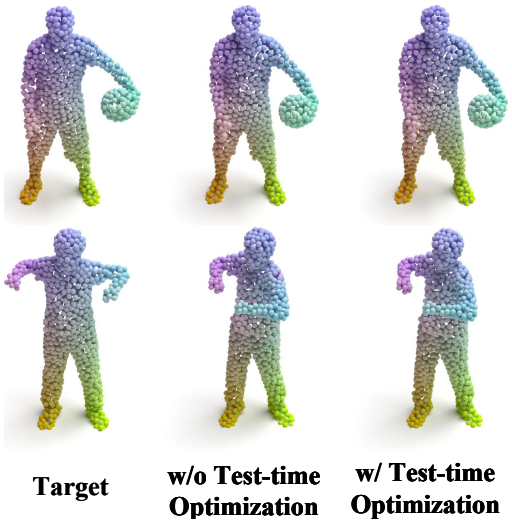

Figure 3. **Visualization on Real Scanned Owlii Dataset.** Our method can still achieve consistent correspondence on the unseen real-scanned dataset without training on it.

3.2. Visualization of Real-scanned Dataset

The real-scanned Owlii dataset [14] is used to evaluate the robustness and generalization capabilities of our method, which contains human scans with challenges of scan noise, unseen poses, and unseen accessories in the training datasets. As shown in Fig. 3, we directly evaluate the performance of the pretrained model on SURREAL without training a new model on Owlii, and find it still achieves semantically accurate and consistent correspondence faced with the challenges above.

3.3. Extended Visualization of Correspondence with Partially Degraded Topology Confidence

In Sec. 4.3 of the main paper, we visualize typical cases where the estimated topology confidence fails to distinguish physically-connected shape parts clearly. In Fig. 4, the extended results with degraded topology confidence still achieve better performance than those without topology estimation, demonstrating that the learned partially degraded topology still preserves informative topological cues to assist correspondence.

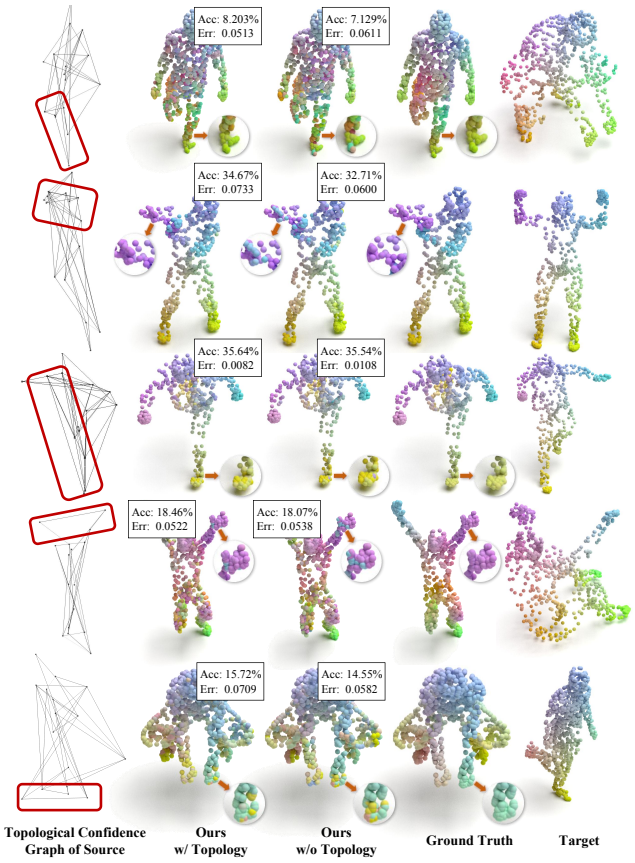

Figure 4. **Extended Results of Correspondence with Inferior Topology Estimation.** The extended results with degraded topology still achieve better performance than those without topology estimation, demonstrating that the learned topology still preserves informative topological cues.

3.4. Visualization of Failure Cases

Although our method can provide consistent correspondence with topology-awareness, severe shape and posture variation still challenge our method in corresponding details, as shown in Fig. 5. Two reasons result in the failed correspondences: (a) Shape pairs with severe posture differences usually have some mismatches in symmetric or locally similar regions. This is because these regions are naturally similar to other irrelevant shape parts, while deformation further brings severe inconsistent characteristics to corresponding regions, such as spatial locations and relative relations. (b) Non-physically connected shape parts in some postures are spatially close to each other, disturbing the topology learning module from telling them apart. This makes topology learning difficult and yields topology weights of low quality, as illustrated in Fig. 6.

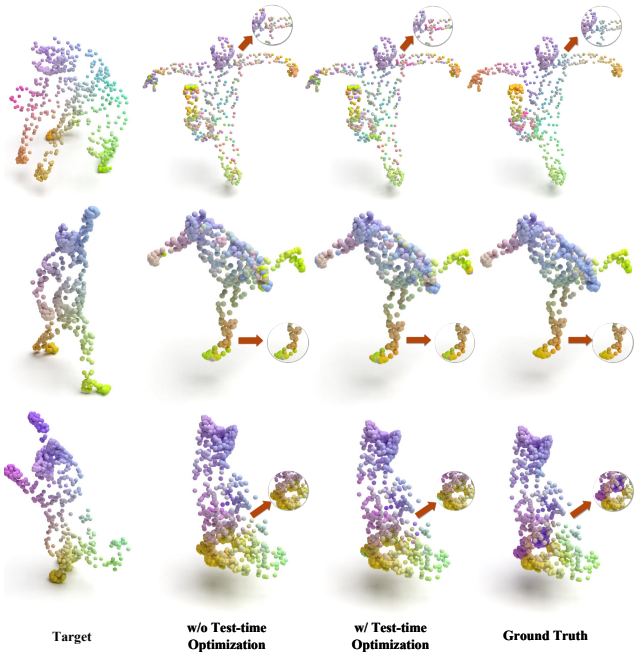

Figure 5. **Visualization of Failure Cases.** We provide samples with wrong matches in the corresponding details. Severe posture variations and non-physically connected parts with close spatial locations pose challenges to our method.

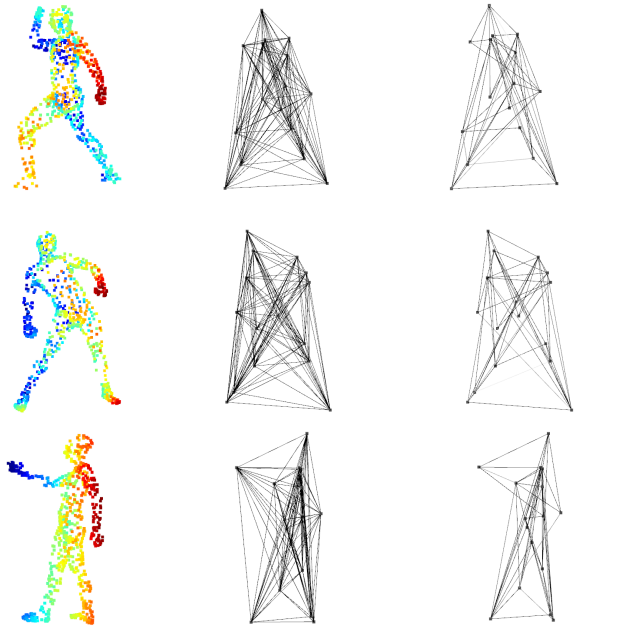

Figure 6. **Visualization of Topology Learning Failure Cases.** Severe posture and deformation variations challenge the accurate learning of shape topology.

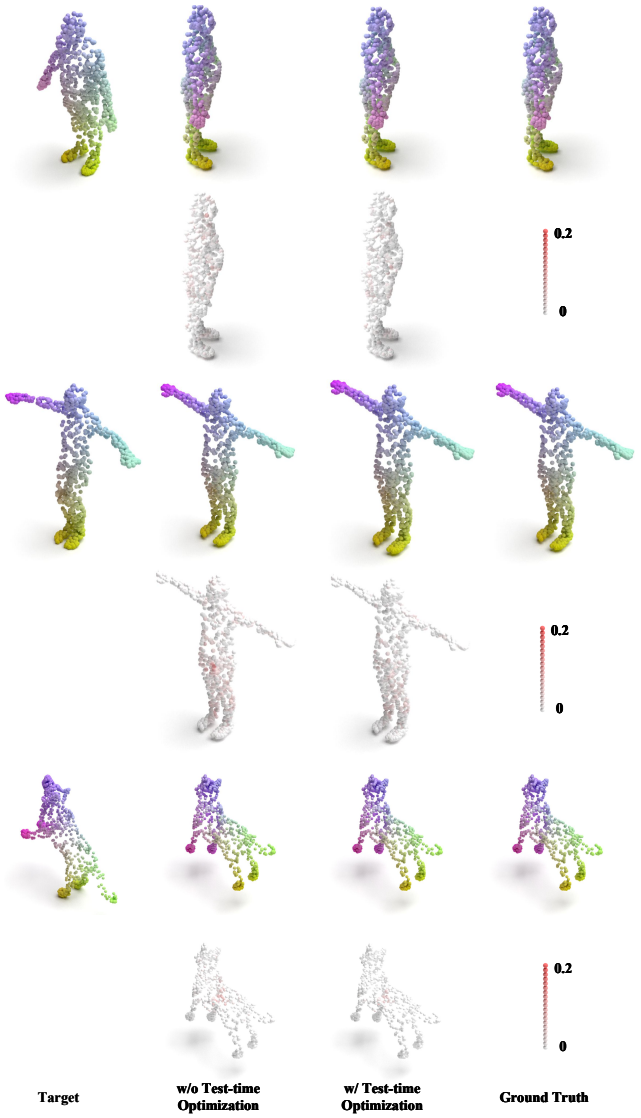

Figure 7. **Visualization of Correspondence Error.** We visualize the correspondence error in red. Our method yields consistent correspondence with insignificant error.

References

[1] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020. 5

[2] Jiacheng Deng, Chuxin Wang, Jiahao Lu, Jianfeng He, Tianzhu Zhang, Jiyang Yu, and Zhe Zhang. Se-ornet: Self-ensembling orientation-aware network for unsupervised point cloud shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5364–5373, 2023. 5, 6

[3] Niladri Shekhar Dutt, Sanjeev Muralikrishnan, and Niloy J Mitra. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4494–4504, 2024. 5, 6

[4] Chengkai Hou, Zhengrong Xue, Bingyang Zhou, Jinghan Ke, Lin Shao, and Huazhe Xu. Key-grid: Unsupervised 3d keypoints detection using grid heatmap features. *Advances in Neural Information Processing Systems*, 37:49154–49179, 2024. 1, 3

[5] Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2021. 1

[6] Oliver Kramer and Oliver Kramer. K-nearest neighbors. *Dimensionality Reduction with Unsupervised Nearest Neighbors*, pages 13–23, 2013. 1

[7] Itai Lang, Dvir Ginzburg, Shai Avidan, and Dan Raviv. Dpc: Unsupervised deep point correspondence via cross and self construction. In *International Conference on 3D Vision*, pages 1442–1451. IEEE, 2021. 2, 3

[8] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017. 1, 2

[9] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021. 1

[10] Woncheol Shin, Gyubok Lee, Jiyoung Lee, Eunyi Lyou, Joonseok Lee, and Edward Choi. Exploration into translation-equivariant image quantization. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023. 3

[11] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 3

[12] Ling Wang, Runfa Chen, Fuchun Sun, Xinzhou Wang, Kai Sun, Chengliang Zhong, Guangyuan Fu, and Yikai Wang. Equivariant local reference frames with optimization for robust non-rigid point cloud correspondence. *IEEE Transactions on Image Processing*, 2025. 1, 2

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5):1–12, 2019. 2

[14] Yao Lu Yi Xu and Ziyu Wen. Owlii dynamic human mesh sequence dataset. In *ISO/IEC JTC1/SC29/WG11 m41658, MPEG Meeting*, 2017. 6

Figure 8. **Visualization on Human Shapes**. Our method yields accurate correspondence for human pairs over various poses and orientations.

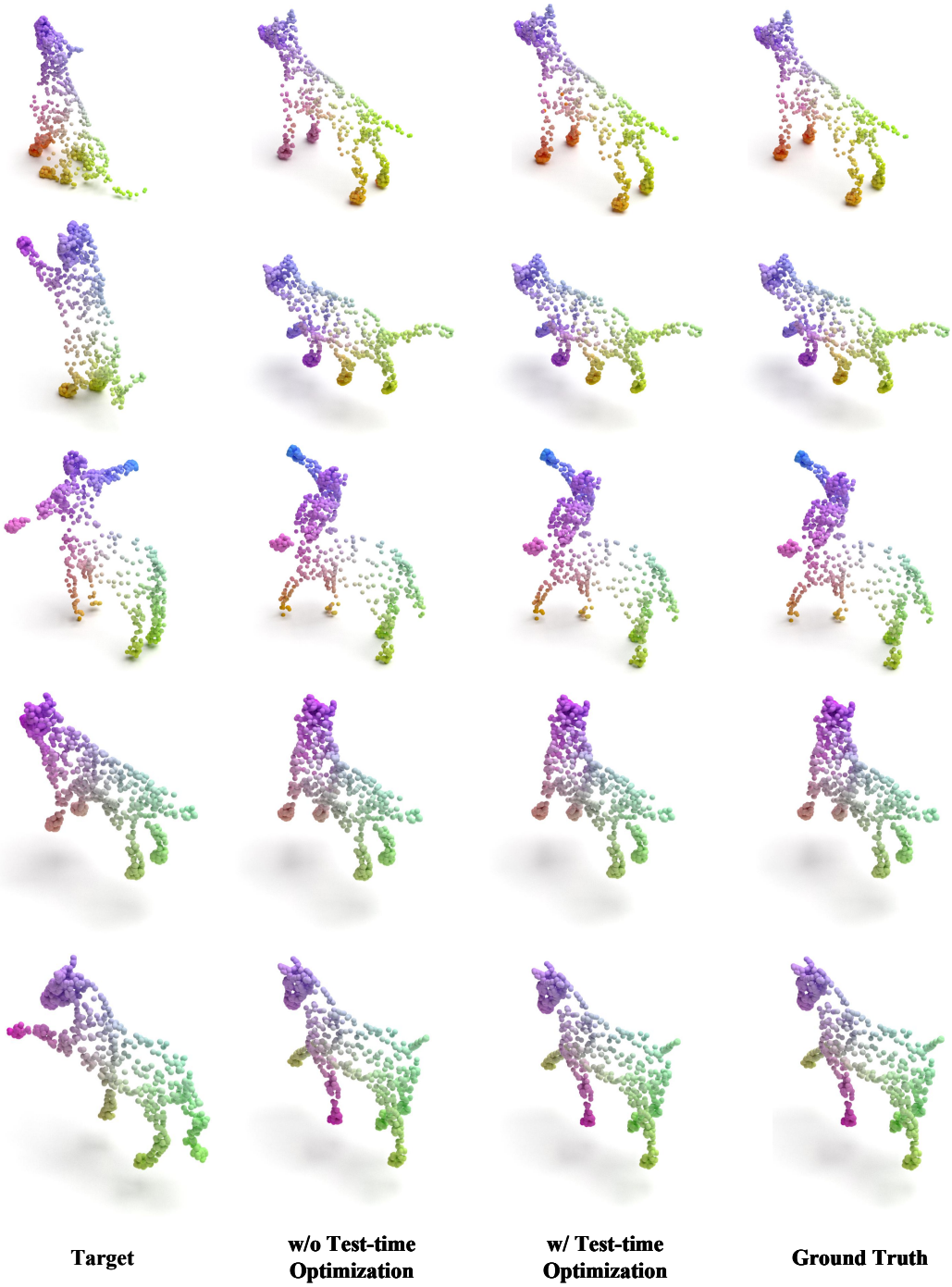

Figure 9. **Visualization on Animal Shapes**. Our method yields accurate correspondence for animal pairs with diversified species and poses.

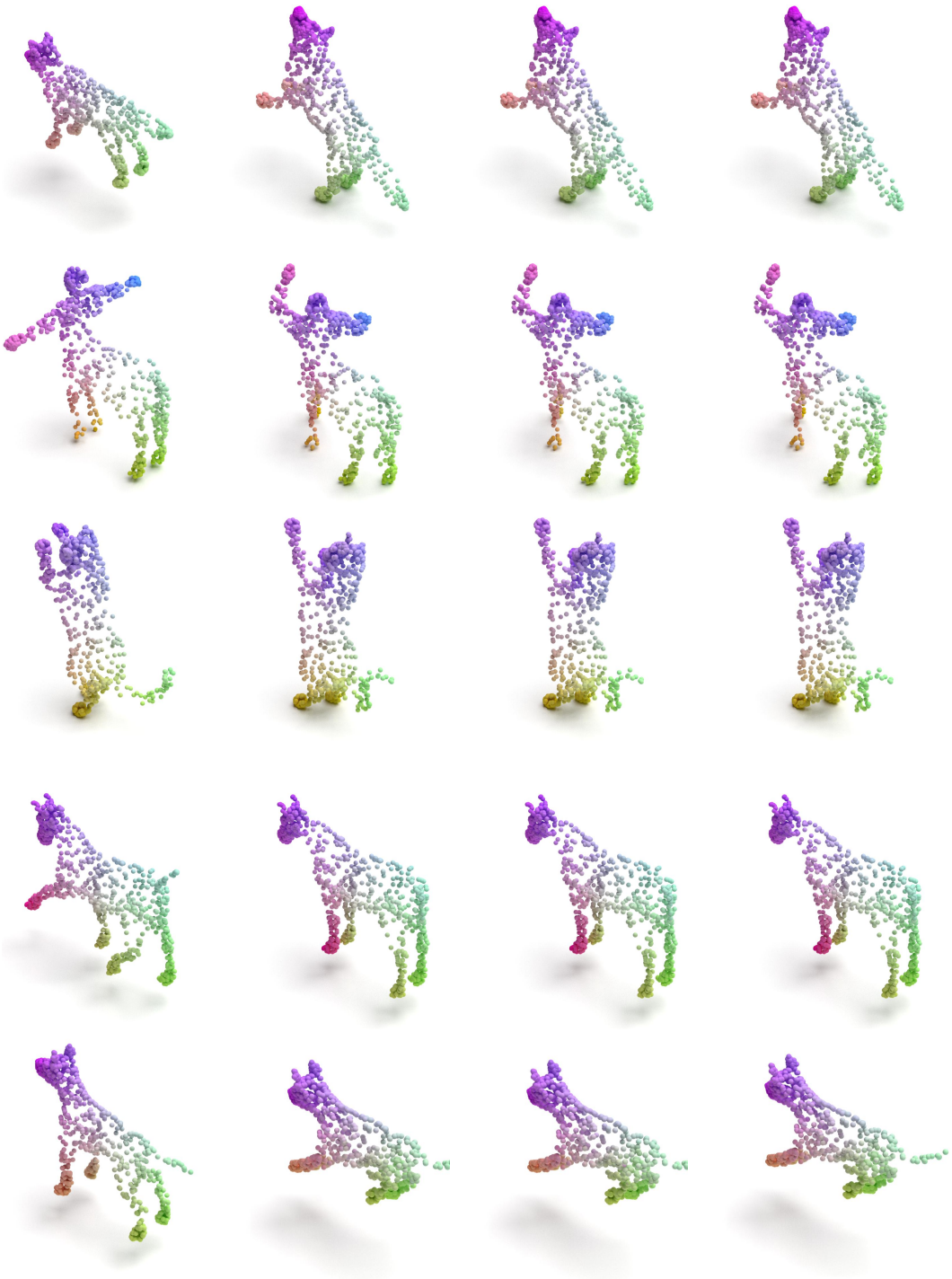

Figure 10. **Visualization on Animal Shapes**. Our method yields accurate correspondence for animal pairs with diversified species and poses.