3D Pose Transfer with Correspondence Learning and Mesh Refinement

Chaoyue Song[1], Jiacheng Wei[2], Ruibo Li[1,3], Fayao Liu[4] and Guosheng Lin[1,3]*

[1]S-Lab, Nanyang Technological University, Singapore
[2]School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
[3]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[4]Institute for Inforcomm Research, A*STAR, Singapore
{chaoyue.song, gslin}@ntu.edu.sg

Abstract

3D pose transfer is one of the most challenging 3D generation tasks. It aims to transfer the pose of a source mesh to a target mesh and keep the identity (e.g., body shape) of the target mesh. Some previous works require key point annotations to build reliable correspondence between the source and target meshes, while other methods do not consider any shape correspondence between sources and targets, which leads to limited generation quality. In this work, we propose a correspondence-refinement network to help the 3D pose transfer for both human and animal meshes. The correspondence between source and target meshes is first established by solving an optimal transport problem. Then, we warp the source mesh according to the dense correspondence and obtain a coarse warped mesh. The warped mesh will be better refined with our proposed *Elastic Instance Normalization*, which is a conditional normalization layer and can help to generate high-quality meshes. Extensive experimental results show that the proposed architecture can effectively transfer the poses from source to target meshes and produce better results with satisfied visual performance than state-of-the-art methods. Our code and data are available at https://github.com/ChaoyueSong/3d-corenet.

1 Introduction

3D pose transfer has been drawing a lot of attention from the vision and graphics community. It has potential applications in 3D animated movies and games for generating new poses for existing shapes and animation sequences. 3D pose transfer is a learning-driven generation task which is similar to style transfer on 2D images. As shown in Figure 1, pose transfer takes two inputs, one is identity mesh that provides identity information of the mesh (e.g., body shape), the other is pose mesh that provides pose information. It aims at transferring the pose of a source pose mesh to a target identity mesh and keeping the identity of the target identity mesh.

A fundamental problem for previous methods is to build reliable correspondence between source and target meshes. It can be very challenging when the source and target meshes have significant differences. Most of the previous methods try to solve it with the help of user effort or other additional inputs, such as key point annotations [3, 35, 42], etc. Unfortunately, it is time-consuming to obtain such additional inputs that will limit the usage in practice. In [38], they proposed to implement pose transfer without correspondence learning. Their method is convenient but the performance will be degraded since they do not consider the correspondence between meshes. In this work, we propose a *COrrespondence-REfinement Network (3D-CoreNet)* to solve the pose transfer problem for both the human and animal meshes. Like [38], our method does not need key point annotations or other

*Corresponding author

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

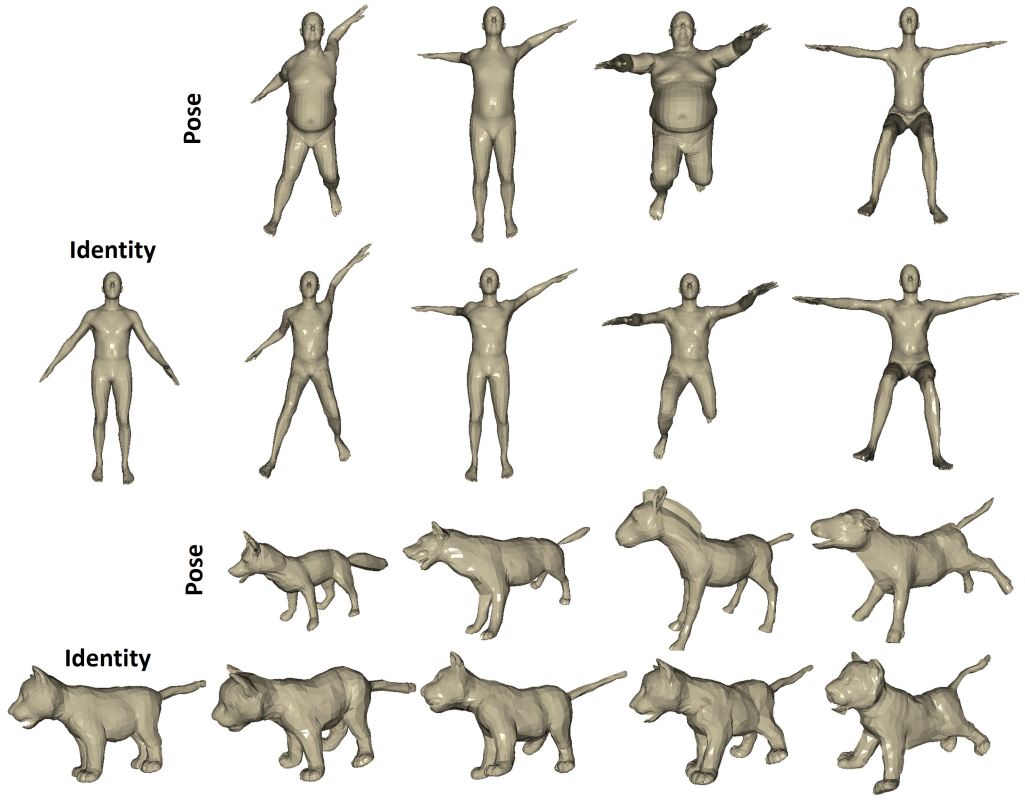

Figure 1: **Pose transfer results generated by our *3D-CoreNet*.** In the first two rows, the human identity and pose meshes are from SMPL [24], in the last two rows, the animal identity and pose meshes are from SMAL [47].

additional inputs. We learn the shape correspondence between identity and pose meshes first, then we warp the pose mesh to a coarse warped output according to the correspondence. Finally, the warped mesh will be refined to have a better visual performance. Our method does not require the two meshes to have the same number or order of vertices.

For the correspondence learning module, we treat the shape correspondence learning as an optimal transport problem to learn the correspondence between meshes. Our network takes vertex coordinates of identity and pose meshes as inputs. We extract deep features at each vertex using point cloud convolutions and compute a matching cost between the vertex sets with the extracted features. Our goal is to minimize the matching cost to get an optimal matching matrix. With the optimal matching matrix, we warp the pose mesh and obtain a coarse warped mesh. We then refine the warped output with a set of elastic instance normalization residual blocks, the modulation parameters in the normalization layers are learned with our proposed *Elastic Instance Normalization (ElaIN)*. In order to generate smoother meshes with more details, we introduce a channel-wise weight in ElaIN to adaptively blend feature statistics of original features and the learned parameters from external data, which help to keep the consistency and continuity of the original features.

Our contributions can be summarized as follows:
• We solve the pose transfer problem with our proposed correspondence-refinement network. To the best of our knowledge, our method is the first to learn the correspondence between different meshes and refine the generated meshes jointly in the 3D pose transfer task.
• We learn the shape correspondence by solving an optimal transport problem without any key point annotations and generate high-quality final meshes with our proposed elastic instance normalization in the refinement module.
• Through extensive experiments, we demonstrate that our method outperforms state-of-the-art methods quantitatively and qualitatively on both human and animal meshes.

2 Related work

2.1 Deep learning methods on 3D data

The representations of 3D data are various, like point clouds, voxels and meshes. 3DShapeNets [41] and VoxNet [26] propose to learn on volumetric grids. Their methods cannot be applied on complex data due to the sparsity of data and computation cost of 3D convolution. PointNet [30] uses a shared MLP on every point followed by a global max-pooling. Following PointNet, some hierarchical architectures have been proposed to aggregate local neighborhood information with MLPs [21, 31]. [14, 36] proposed mesh variational autoencoder to learn mesh features whose methods consume a large amount of computing resources due to their fully-connected networks. Many works use graph convolutions with mesh down- and up-sampling layers [16], like CoMA [32], CAPE [25] based on ChebyNet [11], and [46] based on SpiralNet [22], SpiralNet++ [17], etc. They all need a template to implement their hierarchical structure which is not applicable to open-world problems. In this work, we use mesh as the representation of 3D shape and shared weights convolution layers in the network.

2.2 3D pose transfer

Deformation transfer in graphics aims to apply the deformation exhibited by a source mesh onto a different target mesh [35]. 3D pose transfer aims to generate a new mesh based on the knowledge of a pair of source and target meshes. In [3, 35, 42, 43], the methods all require to label the corresponding landmarks first to deal with the differences between meshes. Baran et al. [2] proposed a method that infers a semantic correspondence between different poses of two characters with the guidance of example mesh pairs. Chu et al. [8] proposed to use a few examples to generate results which will make it difficult to automatically transfer pose. For this problem, Gao et al. [15] proposed to use cycle consistency to achieve the pose transfer. However, their method cannot deal with new identities due to the limitations of the visual similarity metric. In [38], they solved the pose transfer via the latest technique for image style transfer. Their work does not need other guidance, but the performance is also restrained since they do not learn any correspondence. To solve the problems, our network learns the correspondence and refines the generated meshes jointly.

2.3 Correspondence learning

In CoCosNet [45], they introduced a correspondence network based on the correlation matrix between images without any constraints. To learn a better matching, we proposed to use optimal transport to learn the correspondence between meshes. Recently, optimal transport has received great attention in various computer vision tasks. Courty et al. [9] perform the alignment of the representations in the source and target domains by learning a transportation plan. Su et al. [34] compute the optimal transport map to deal with the surface registration and shape space problem. Other applications include generative model [1, 6, 12, 40], scene flow [29], semantic correspondence [23] and etc.

2.4 Conditional normalization layers

After normalizing the activation value, conditional normalization uses the modulation parameters calculated from the external data to denormalize it. Adaptive Instance Normalization (AdaIN) [19] aligns the mean and variance between content and style image which achieves arbitrary style transfer. Soft-AdaIN [7] introduces a channel-wise weight to blend feature statistics of content and style image to preserve more details for the results. Spatially-Adaptive Normalization (SPADE) [28] can better preserve semantic information by not washing away it when applied to segmentation masks. SPAdaIN [38] changes batch normalization [20] in SPADE to instance normalization [37] for 3D pose transfer. However, it will break the consistency and continuity of the feature map when doing the denormalization, which has a bad influence on the mesh smoothness and detail preservation. To address this problem, our ElaIN introduces an adaptive weight to implement the denormalization.

3 Method

Given a source pose mesh and a target identity mesh, our goal is to transfer the pose of source mesh to the target mesh and keep the identity of the target mesh. In this section, we will introduce our end-to-end *Correspondence-Refinement Network (3D-CoreNet)* for 3D pose transfer.

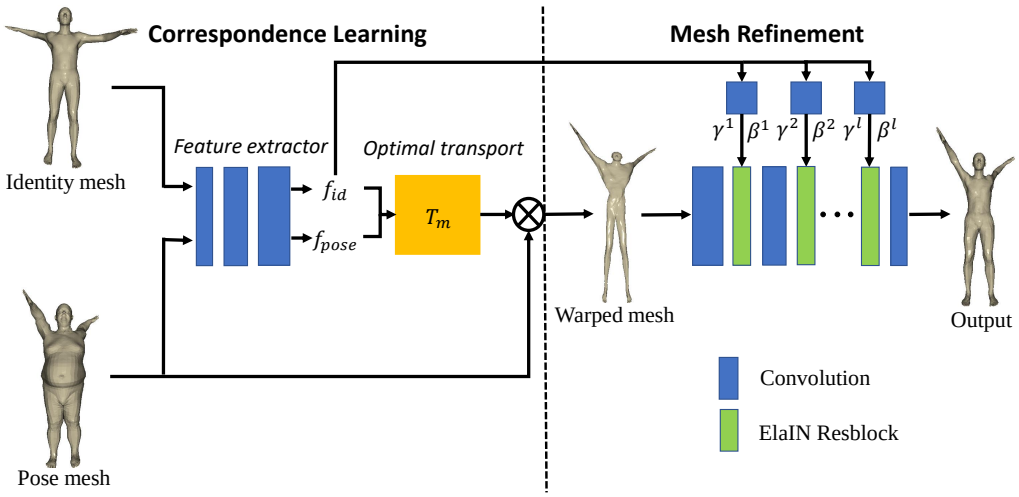

Figure 2: **The architecture of *3D-CoreNet*.** With the extracted features, the shape correspondence between identity and pose meshes is first established by solving an optimal transport problem. Then, we warp the pose mesh according to the optimal matching matrix $\mathbf{T}_m$ and obtain a coarse warped mesh. The warped mesh will be better refined with our proposed ElaIN in the refinement module.

A 3D mesh can be represented as $M(I, P, O)$, where I denotes the mesh identity, P represents the pose of the mesh, O is the vertex order of the mesh. Given two meshes $M_{id} = M(I_1, P_1, O_1)$ and $M_{pose} = M(I_2, P_2, O_2)$, we aim to transfer the pose of M_{pose} to M_{id} and generate the output mesh $M_{output} = M'(I_1, P_2, O_1)$. In our *3D-CoreNet*, we take $V_{id} \in \mathbb{R}^{N_{id} \times 3}$ and $V_{pose} \in \mathbb{R}^{N_{pose} \times 3}$ as inputs, which are the (x, y, z) coordinates of the mesh vertices. N_{id} and N_{pose} denote the number of vertices of identity mesh and pose mesh respectively.

As shown in Figure 2, the vertices of the meshes are first fed into the network to extract multi-scale deep features. We calculate the matching matrix with the vertex feature maps by solving an optimal transport problem, then we warp the pose mesh according to the matrix and obtain the warped mesh. Finally, the warped mesh is refined to the final output mesh with our proposed elastic instance normalization (ElaIN). The output mesh combines the pose from the source mesh and the identity from the target. And it inherits the vertex order from the identity mesh.

3.1 Correspondence learning

Given an identity mesh and a pose mesh, our correspondence learning network calculates an optimal matching matrix, each element of the matching matrix represents the similarities between two vertices in the two meshes. The first step in our shape correspondence learning is to compute a correlation matrix with the extracted features, which is based on cosine similarity and denotes the matching similarities between any two positions from different meshes. However, the matching scores in the correlation matrix are calculated without any additional constraints. To learn a better matching, we solve this problem from a global perspective by modeling it as an optimal transport problem.

Correlation matrix We first introduce our feature extractor which aims to extract features for the unordered input vertices. Close to [38], our feature extractor consists of 3 stacked 1×1 convolution and Instance Normalization layers, the activation functions applied are all LeakyReLU. Given the extracted vertex feature maps $f_{id} \in \mathbb{R}^{D \times N_{id}}$, $f_{pose} \in \mathbb{R}^{D \times N_{pose}}$ of identity and pose meshes (D is the channel-wise dimension), a popular method to compute correlation matrix is using the cosine similarity [44, 45]. Concretely, we compute the correlation matrix $\mathbf{C} \in \mathbb{R}^{N_{id} \times N_{pose}}$ as:

$$\mathbf{C}(i, j) = \frac{f_{id}(i)^\top f_{pose}(j)}{\|f_{id}(i)\| \, \|f_{pose}(j)\|} \tag{1}$$

where $\mathbf{C}(i, j)$ denotes the individual matching score between $f_{id}(i)$ and $f_{pose}(j) \in \mathbb{R}^{\mathbb{D}}$, $f_{id}(i)$ and $f_{pose}(j)$ represent the channel-wise feature of f_{id} at position i and f_{pose} at j.

Optimal transport problem To learn a better matching with additional constraints in this work, we model our shape correspondence learning as an optimal transport problem. We first define a matching matrix $\mathbf{T} \in \mathbb{R}_{+}^{N_{id} \times N_{pose}}$ between identity and pose meshes. Then we can get the total correlation as $\sum_{ij} \mathbf{C}(i,j)\mathbf{T}(i,j)$. The aim will be maximizing the total correlation score to get an optimal matching matrix $\mathbf{T}_m$.

We treat the correspondence learning between identity and pose meshes as the transport of mass. A mass which is equal to N_{id}^{-1} will be assigned to each vertex in the identity mesh, and each vertex in pose mesh will receive the mass N_{pose}^{-1} from identity mesh through the built correspondence between vertices. Then if we define $\mathbf{Z} = 1 - \mathbf{C}$ as the cost matrix, our goal can be formulated as a standard optimal transport problem by minimizing the total matching cost,

$$\mathbf{T}_m = \underset{\mathbf{T} \in \mathbb{R}_{+}^{N_{id} \times N_{pose}}}{\arg\min} \sum_{ij} \mathbf{Z}(i,j)\mathbf{T}(i,j) \quad s.t. \quad \mathbf{T}\mathbf{1}_{N_{pose}} = \mathbf{1}_{N_{id}} N_{id}^{-1}, \quad \mathbf{T}^{\top}\mathbf{1}_{N_{id}} = \mathbf{1}_{N_{pose}} N_{pose}^{-1}. \tag{2}$$

where $\mathbf{1}_{N_{id}} \in \mathbb{R}^{N_{id}}$ and $\mathbf{1}_{N_{pose}} \in \mathbb{R}^{N_{pose}}$ are vectors whose elements are all 1. The first constraint in Eq. 2 means that the mass of each vertex in M_{id} will be entirely transported to some of the vertices in M_{pose}. And each vertex in M_{pose} will receive a mass N_{pose}^{-1} from some of the vertices in M_{id} with the second constraint. This problem can be solved by the Sinkhorn-Knopp algorithm [33]. The details of the solved process will be given in the supplementary material.

With the matching matrix, we can warp the pose mesh and obtain the vertex coordinates $V_{warp} \in \mathbb{R}^{N_{id} \times 3}$ of the warped mesh.

$$V_{warp}(i) = \sum_{j} \mathbf{T}_m(i,j) V_{pose}(j) \tag{3}$$

The warped mesh M_{warp} inherits the number and order of vertex from identity mesh and can be reconstructed with the face information of identity mesh as shown in Figure 2.

3.2 Mesh refinement

In this section, we introduce our mesh refinement module which refines the warped mesh to the desired output progressively.

Elastic instance normalization Previous conditional normalization layers [19, 28, 38] used in different tasks always calculated their denormalization parameters only with the external data. We argue that it may break the consistency and continuity of the original features. Inspired by [7], we propose *Elastic Instance Normalization (ElaIN)* which blends the statistics of original features and the learned parameters from external data adaptively and elastically.

As shown in Figure 2, the warped mesh is flatter than we desired and is kind of out of shape, but it inherits the pose from the source mesh successfully. Therefore, we denormalize the warped mesh with the feature maps of the identity mesh to get a better final output. Here, we let $h^i \in \mathbb{R}^{S^i \times D^i \times N^i}$ as the activation value before the i-th normalization layer, where S^i is the batch size, D^i is the dimension of feature channel and N^i is the number of vertex. At first, we normalize the feature maps of the warped mesh with instance normalization and the mean and standard deviation are calculated across spatial dimension ($n \in N^i$) for each sample $s \in S^i$ and each channel $d \in D^i$,

$$\mu^i = \frac{1}{N^i} \sum_{n} h_{warp}^i \tag{4}$$

$$\sigma^i = \sqrt{\frac{1}{N^i} \sum_{n} (h_{warp}^i - \mu^i)^2 + \epsilon} \tag{5}$$

Then the feature maps of the identity mesh are fed into a 1×1 convolution layer to get h_{id}^i, which shares the same size with h_{warp}^i. We adopt global average pooling to pool h_{warp}^i, h_{id}^i into $S^i \times D^i \times 1$ tensors. The tensors are then concatenated in channel dimension to get a $S^i \times (2D^i) \times 1$ tensor.

A fully-connected layer is employed to compute an adaptive weight $w(h^i_{warp}, h^i_{id}) \in \mathbb{R}^{S^i \times D^i \times 1}$ with the concatenated tensor. With $w(h^i_{warp}, h^i_{id})$, we can define the modulation parameters of our normalization layer.

$$\begin{aligned}\gamma'(h^i_{warp}, h^i_{id}) &= w(h^i_{warp}, h^i_{id})\gamma^i + (1 - w(h^i_{warp}, h^i_{id}))\sigma^i, \\ \beta'(h^i_{warp}, h^i_{id}) &= w(h^i_{warp}, h^i_{id})\beta^i + (1 - w(h^i_{warp}, h^i_{id}))\mu^i.\end{aligned} \tag{6}$$

where γ^i and β^i are learned from the identity feature h^i_{id} with two convolution layers. Finally, we can scale the normalized h^i_{warp} with γ' and shift it with β',

$$\text{ElaIN}(h^i_{warp}, h^i_{id}) = \gamma'(h^i_{warp}, h^i_{id})(\frac{h^i_{warp} - \mu^i}{\sigma^i}) + \beta'(h^i_{warp}, h^i_{id}) \tag{7}$$

Refinement network Our refinement network is designed to refine the warped mesh progressively. Following [28, 38, 45], We design the ElaIN residual block with our ElaIN in the form of ResNet blocks [18]. As shown in Figure 2, our architecture contains l ElaIN residual blocks. Each of them consists of our proposed ElaIN followed by a simple convolution layer and LeakyReLU. With the ElaIN residual blocks, the warped mesh is refined to our desired high-quality output. Please refer to the supplementary material for the detailed architecture of ElaIN.

3.3 Loss function

We jointly train the correspondence learning module along with mesh refinement module by minimizing the following loss functions,

Reconstruction loss Following [38], we train our network with the supervision of the ground truth mesh $M_{gt} = M(I_1, P_2, O_1)$. We first process the ground truth mesh to have the same vertex order as the identity mesh. Then we define the reconstruction loss by calculating the point-wise $L2$ distance between the vertices of M_{output} and M_{gt},

$$\mathcal{L}_{rec} = \|V_{output} - V_{gt}\|_2^2 \tag{8}$$

where V_{output} and $V_{gt} \in \mathbb{R}^{N_{id} \times 3}$ are the vertices of M_{output} and M_{gt} respectively. Notice that they all share the same size and order with the vertices of the identity mesh. With the reconstruction loss, the mesh predicted by our model will be closer to the ground truth.

Edge loss In this work, we also introduce edge loss which is often used in 3D mesh generation tasks [27, 38, 39]. Since the reconstruction loss does not consider the connectivity of mesh vertices, the generated mesh may suffer from flying vertices and overlong edges. Edge loss can help penalize flying vertices and generate smoother surfaces. For every $v \in V_{output}$, let $\mathcal{N}(v)$ be the neighbor of v, the edge loss can be defined as,

$$\mathcal{L}_{edg} = \sum_{v} \sum_{p \in \mathcal{N}(v)} \|v - p\|_2^2 \tag{9}$$

Then we can train our network with the combined loss function $\mathcal{L}$,

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \mathcal{L}_{edg} \tag{10}$$

where λ_{rec} denotes the weight of reconstruction loss.

4 Experiment

Datasets. For the human mesh dataset, we use the same dataset generated by SMPL [24] as Wang et al. [38]. This dataset consists of 30 identities with 800 poses. Each mesh has 6890 vertices. For the training data, we randomly choose 4000 pairs (identity and pose mesh) from 16 identities with 400 poses and shuffle them every epoch. The ground truth meshes will be determined according to the identity and pose parameters from the pairs. Before feeding into our network, every mesh will be shuffled randomly to be close to the open-world problem. Notice that the ground truth mesh

Table 1: **Quantitative comparison with other methods.** We compare our method with DT (needs key point annotations) and Wang et al. using PMD, CD, EMD as our evaluation metrics on both human and animal data. For them, smaller is better. The PMD and CD are in units of 10^{-3} and the EMD is in units of 10^{-2}.

	Annotation	Dataset	PMD	CD	EMD
DT [35]	Key points		0.15	0.35	2.21
Wang et al. [38]	-	SMPL [24]	0.66	1.42	4.22
Ours	-		**0.08**	**0.22**	**1.89**
DT [35]	Key points		13.37	35.77	15.90
Wang et al. [38]	-	SMAL [47]	6.75	14.52	11.65
Ours	-		**2.26**	**4.05**	**7.28**

will share the same vertex order with identity mesh for the convenience of supervision training and evaluation. For all input meshes, we shift them to the center according to their bounding box. When doing the test, we evaluate our model with 14 new identities with 200 unseen poses. We randomly choose 400 pairs for testing. They will be pre-processed in the same manner as training data. To further test the generalization of our model, we also try our model with FAUST [5] and MG-dataset [4] in the experiment.

For the animal mesh dataset, we generate an animal training and test data using SMAL model [47]. This dataset has 41 identities with 600 poses. The 41 identities are 21 felidae animals (1 cat, 5 cheetahs, 8 lions, 7 tigers), 5 canidae animals (2 dogs, 1 fox, 1 wolf, 1 hyena), 8 equidae animals (1 deer, 1 horse, 6 zebras), 4 bovidae animals (4 cows), 3 hippopotamidae animals (3 hippos). Every mesh has 3889 vertices. For the training data, we randomly choose 11600 pairs from 29 identities (16 felidae animals, 3 canidae animals, 6 equidae animals, 2 bovidae animals, 2 hippopotamidae animals) with 400 poses. For the test data, we randomly choose 400 pairs from other 12 identities (5 felidae animals, 2 canidae animals, 2 equidae animals, 2 bovidae animals, 1 hippopotamidae animal) with 200 poses. All the inputs are pre-processed in the same manner as we do in the human mesh.

Evaluation metrics. Following [38], we use Point-wise Mesh Euclidean Distance (PMD) as one of our evaluation metrics. PMD is the $L2$ distance between the vertices of the output mesh and the ground truth mesh. We also evaluate our model with Chamfer Distance (CD) and Earth Mover's Distance (EMD) proposed in [13]. For PMD, CD and EMD, smaller is better.

Implementation details. The λ_{rec} in the loss function is set as 2000. We implement our model with Pytorch and use Adam optimizer. Please refer to the supplementary material for the details of the network. Our model is trained for 200 epochs on one RTX 3090 GPU, the learning rate is fixed at $2e$-4 in the first 100 epochs and decays $2e$-6 each epoch after 100 epochs. The batch size is 8.

4.1 Comparison with the state-of-the-arts

In this section, we compare our method with Deformation Transfer (DT) [35] and Wang et al. [38]. DT needs to rely on the control points labeled by user and a reference mesh, as the additional inputs. Therefore, we test DT with the reference mesh and 11, 19 labeling points on animal data and human data respectively. For [38], their method does not consider any correspondence. We train their model using the implementations provided by the authors. The qualitative results tested on SMPL [24] and SMAL [47] are shown in Figure 3 and Figure 4. As we can see, when tested on human data, DT and our method can produce better results that are close to the ground truth. However, it is very time-consuming for DT when dealing with a new identity and pose mesh pair. The results generated by Wang et al. always do not learn the right pose and are not very smooth on the arms or legs since they do not consider any correspondence. When tested on animal data, DT fails to transfer the pose even if we add more labeling points. Their method does not work when the identity of the mesh pairs are very different. Wang et al. may produce very flat legs and wrong direction faces. In comparison, our method still produces satisfactory results efficiently.

We adopt Point-wise Mesh Euclidean Distance (PMD), Chamfer Distance (CD) and Earth Mover's Distance (EMD) to evaluate the generated results of different methods. All metrics are calculated

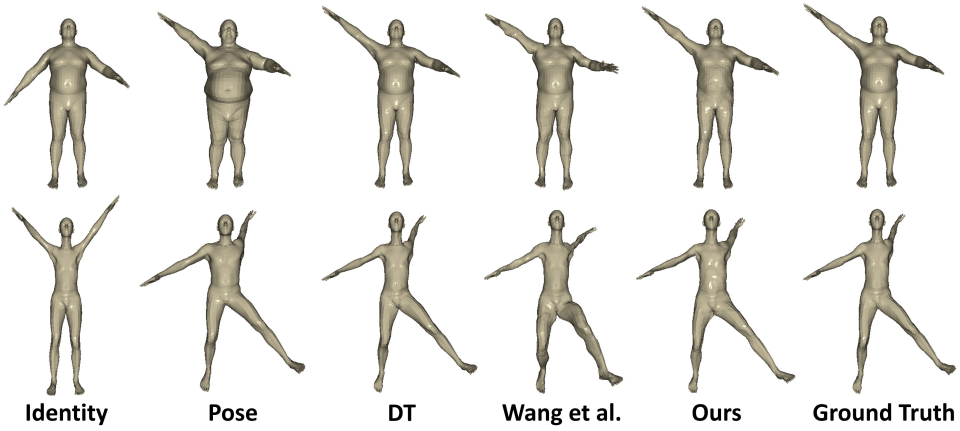

Figure 3: **Qualitative comparison of different methods on human data.** The identity and pose meshes are from SMPL [24]. Our method and DT (needs key point annotations) can generate better results than Wang et al. when doing pose transfer on human meshes. The results generated by Wang et al. are always not smooth on the arms or legs. Since DT needs user to label the key point annotations, our method is more efficient and practical than DT.

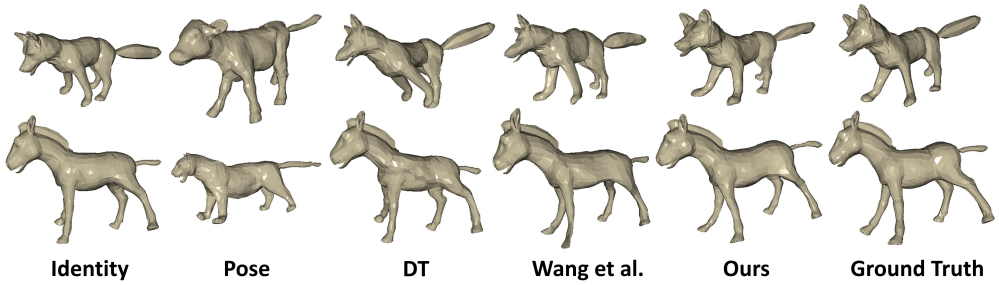

Figure 4: **Qualitative comparison of different methods on animal data.** The identity and pose meshes are from SMAL [47]. Our method produces more successful results when doing pose transfer on different animal meshes. Although DT has key point annotations, it still fails to transfer the pose when the identity of the mesh pairs are very different. The method of Wang et al. produces very flat legs and wrong direction faces.

between the ground truth and the predicted results. The quantitative results are shown in Table 1. Our *3D-CoreNet* outperforms other methods in all metrics over two datasets. When doing pose transfer on animal data that contains more different identities, our method has more advantages.

Table 2: **Ablation study.** We use all 3 measurements here. For them, smaller is better. The PMD and CD are in units of 10^{-3} and the EMD is in units of 10^{-2}. w/o means without this component. In the third and fourth column, we only use our correspondence learning module without refinement. Corr (a) uses the correlation matrix and Corr (b) uses the optimal matching matrix to learn the correspondence. In w/o ElaIN, we replace our ElaIN with SPAdaIN in [38] to compare them.

Dataset		Corr (a)	Corr (b)	w/o ElaIN	w/o $\mathcal{L}_{edg}$	Full model
SMPL [24]	PMD	0.46	0.44	0.15	0.14	**0.08**
	CD	1.39	1.28	0.37	0.34	**0.22**
	EMD	3.49	3.42	2.57	2.28	**1.89**

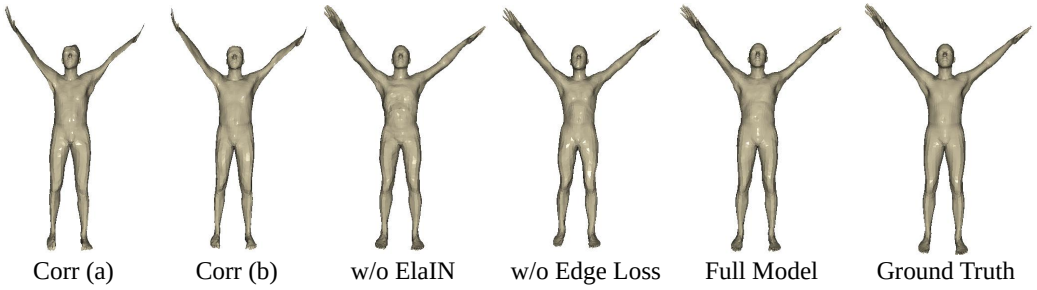

Figure 5: **Ablation study results.** We test 5 variants on SMPL [24]. The first two are tested without refinement, Corr (a) uses the correlation matrix and Corr (b) uses the optimal matching matrix to learn the correspondence. The model does not perform well without refinement. Using the optimal matching matrix has a better performance than using correlation matrix. In the third column, the surface of the mesh has clear artifacts and is not smooth when we replace ElaIN with SPAdaIN.

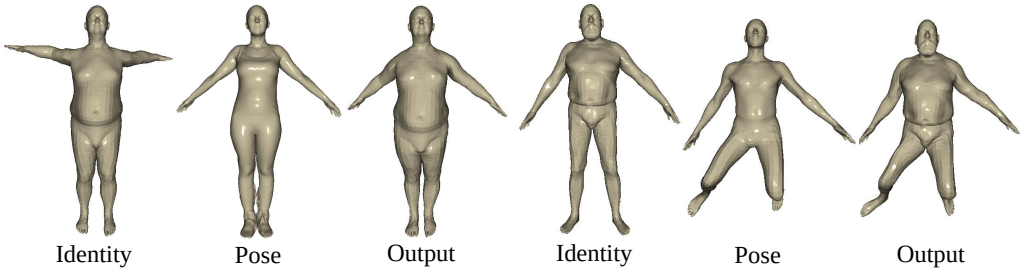

Figure 6: **Pose transfer results on human data from different datasets.** We test our model on FAUST [5] and MG-dataset [4] which contain different human meshes with SMPL [24]. Our method still has a good performance. Please refer to the supplementary material for more generated results.

4.2 Ablation study

In this section, we study the effectiveness of several components in our *3D-CoreNet* on human data. At first, we test our model without the refinement module. We only use our correspondence module with the correlation matrix $\mathbf{C}$ or the optimal matching matrix $\mathbf{T}_m$ respectively. Here, the warped mesh will be viewed as the final output and the reconstruction loss will be calculated between the warped mesh and the ground truth. Then we will compare our ElaIN with SPAdaIN in [38] to verify the effectiveness of ElaIN. And we also test the importance of edge loss $\mathcal{L}_{edg}$.

The results are shown in Table 2 and Figure 5. We evaluate the variants with PMD, CD and EMD. As we can see, when we do not add our refinement module, the model does not perform well both qualitatively and quantitatively. And using the optimal matching matrix has a better performance than using correlation matrix. When we replace our ElaIN with SPAdaIN, the surface of the mesh has clear artifacts and is not smooth. The metrics are also worse than the full model. We can know that ElaIN is very helpful in generating high quality results. We also evaluate the importance of $\mathcal{L}_{edg}$. The connection between vertices will be better and smoother with the edge loss.

4.3 Generalization capability

To evaluate the generalization capability of our method, we evaluate it on FAUST [5] and MG-dataset [4] in this section. Human meshes in FAUST have the same number of vertices as SMPL [24] and have more unseen identities. In MG-dataset, the human meshes are all dressed which have 27554 vertices each and have more realistic details. As shown in Figure 6, our method can also have a good performance on FAUST and MG-dataset. In the first group, we transfer the pose from FAUST to the identity in SMPL. In the second group, we transfer the pose from SMPL to the identity in MG-dataset. Both of them transfer the pose and keep the identity successfully.

5 Conclusion

In this paper, we propose a correspondence-refinement network (*3D-CoreNet*) to transfer the pose of source mesh to the target mesh while retaining the identity of the target mesh. *3D-CoreNet* learns the correspondence between different meshes and refine the generated meshes jointly. Our method generates high-quality meshes with the proposed ElaIN for refinement. Compared to other methods, our model learns the correspondence without key point labeling and achieves better performance when working on both human and animal meshes. In the future, we will extend our approach to unsupervised learning with the help of cycle consistency.

Acknowledgements

This study is supported under the RIE2020 Industry Alignment Fund - Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner. This work is supported by A*STAR through the Industry Alignment Fund - Industry Collaboration Projects Grant. This work is also supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2018-003), and the MOE Tier-1 research grants: RG28/18 (S) and RG95/20.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[2] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. In *ACM SIGGRAPH 2009 papers*, pages 1–6. 2009.

[3] Mirela Ben-Chen, Ofir Weber, and Craig Gotsman. Spatial deformation transfer. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 67–74, 2009.

[4] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5420–5430, 2019.

[5] Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014.

[6] Charlotte Bunne, David Alvarez-Melis, Andreas Krause, and Stefanie Jegelka. Learning generative models across incomparable spaces. In *International Conference on Machine Learning*, pages 851–861. PMLR, 2019.

[7] Yugang Chen, Muchun Chen, Chaoyue Song, and Bingbing Ni. Cartoonrenderer: An instance-based multi-style cartoon image translator. In *International Conference on Multimedia Modeling*, pages 176–187. Springer, 2020.

[8] Hung-Kuo Chu and Chao-Hung Lin. Example-based deformation transfer for 3d polygon models. *J. Inf. Sci. Eng.*, 26(2):379–391, 2010.

[9] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2016.

[10] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26:2292–2300, 2013.

[11] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3844–3852, 2016.

[12] Ishan Deshpande, Yuan-Ting Hu, Ruoyu Sun, Ayis Pyrros, Nasir Siddiqui, Sanmi Koyejo, Zhizhen Zhao, David Forsyth, and Alexander G Schwing. Max-sliced wasserstein distance and its use for gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10648–10656, 2019.

[13] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[14] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019.

[15] Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018.

[16] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[17] Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. Spiralnet++: A fast and highly efficient mesh convolution operator. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[21] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[22] Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[23] Yanbin Liu, Linchao Zhu, Makoto Yamada, and Yi Yang. Semantic correspondence as an optimal transport problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4463–4472, 2020.

[24] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[25] Qianli Ma, Jinlong Yang, Anurag Ranjan, Sergi Pujades, Gerard Pons-Moll, Siyu Tang, and Michael J Black. Learning to dress 3d people in generative clothing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6469–6478, 2020.

[26] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[27] Junyi Pan, Xiaoguang Han, Weikai Chen, Jiapeng Tang, and Kui Jia. Deep mesh reconstruction from single rgb images via topology modification networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9964–9973, 2019.

[28] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[29] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. *arXiv preprint arXiv:2007.11142*, 2020.

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[31] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5105–5114, 2017.

[32] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018.

[33] Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

[34] Zhengyu Su, Yalin Wang, Rui Shi, Wei Zeng, Jian Sun, Feng Luo, and Xianfeng Gu. Optimal mass transport for shape matching and comparison. *IEEE transactions on pattern analysis and machine intelligence*, 37(11):2246–2259, 2015.

[35] Robert W Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on graphics (TOG)*, 23(3):399–405, 2004.

[36] Qingyang Tan, Lin Gao, Yu-Kun Lai, and Shihong Xia. Variational autoencoders for deforming 3d mesh models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5841–5850, 2018.

[37] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[38] Jiashun Wang, Chao Wen, Yanwei Fu, Haitao Lin, Tianyun Zou, Xiangyang Xue, and Yinda Zhang. Neural pose transfer by spatially adaptive instance normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5831–5839, 2020.

[39] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–67, 2018.

[40] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3713–3722, 2019.

[41] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[42] Jie Yang, Lin Gao, Yu-Kun Lai, Paul L Rosin, and Shihong Xia. Biharmonic deformation transfer with automatic key point selection. *Graphical Models*, 98:1–13, 2018.

[43] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020.

[44] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8052–8061, 2019.

[45] Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5143–5153, 2020.

[46] Keyang Zhou, Bharat Lal Bhatnagar, and Gerard Pons-Moll. Unsupervised shape and pose disentanglement for 3d meshes. In *European Conference on Computer Vision*, pages 341–357. Springer, 2020.

[47] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.

A More details of 3D-CoreNet

A.1 Network architecture

The detailed architecture of our correspondence-refinement network (*3D-CoreNet*) is shown in Table 3. We take the vertices of the identity and pose meshes as inputs. Both of them are fed into the feature extractor and the adaptive feature block. The feature extractor consists of three Conv1d-InstanceNorm-LeakyReLU blocks. Then we can calculate the optimal matching matrix with their features by solving an optimal transport problem. With the matrix, we warp the pose mesh to the coarse warped mesh. Finally, the warped mesh is better refined in the mesh refinement module with a set of elastic instance normalization residual blocks. The modulation parameters in the normalization layers are learned with elastic instance normalization.

The design of our elastic instance normalization (*ElaIN*) is shown in Figure 7. At first, we normalize the features of the warped mesh h^i_{warp} with instance normalization and get the mean μ^i and standard deviation σ^i. Then, the features of the identity mesh are fed into a simple convolution layer to get h^i_{id}, which shares the same size with h^i_{warp}. We adopt global average pooling to pool h^i_{warp}, h^i_{id} and concatenate them in the channel dimension. A fully-connected layer is employed to compute an adaptive weight w^i. We blend γ^i, σ^i and β^i, μ^i elastically with w^i to get the modulation parameters γ' and β', where γ^i and β^i are learned from h^i_{id} with two convolution layers. Finally, we scale the normalized h^i_{warp} with γ' and shift it with β'.

Table 3: **The network architecture of *3D-CoreNet***. N indicates the number of vertices, D is the number of feature channels. We give an example when training on SMPL [24]. The number of vertices is 6890. (1×1) means the kernel size of the convolution layer is 1×1.

	Module	Layers	Output ($N \times D$)
Correspondence Learning	Feature extractor	Conv1d (1×1)	6890×64
		Conv1d (1×1)	6890×128
		Conv1d (1×1)	6890×256
	Adaptive feature block	Resblock $\times$ 4	6890×256
		Conv1d (1×1)	6890×256
	Optimal transport	Matching matrix	6890×6890
	Warping	Warped mesh	6890×3
Mesh Refinement	Refinement	Conv1d (3×3)	6890×1024
		Conv1d (1×1)	6890×1024
		ElaIN Resblock	6890×1024
		Conv1d (1×1)	6890×512
		ElaIN Resblock	6890×512
		Conv1d (1×1)	6890×256
		ElaIN Resblock	6890×256
		Conv1d (1×1)	6890×3

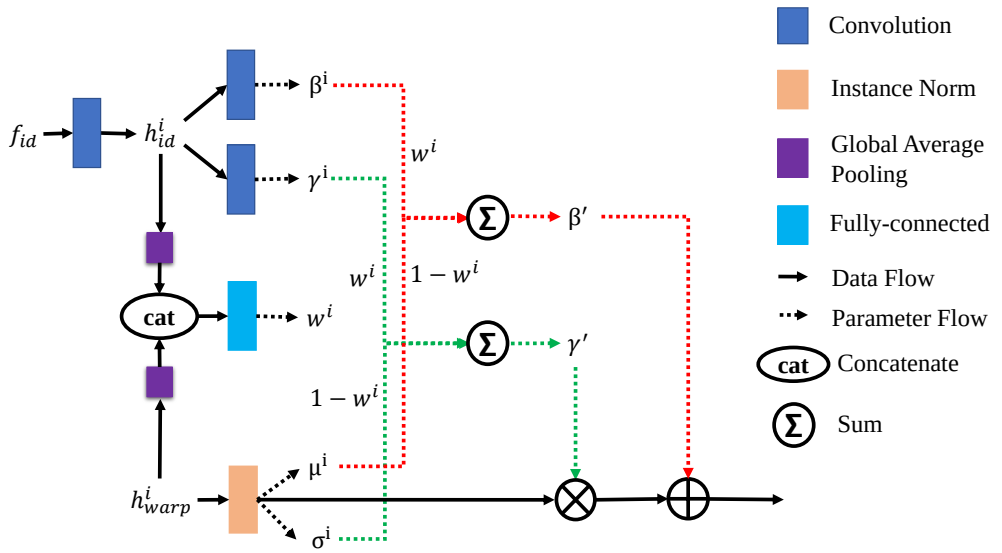

Figure 7: **The detailed design of our elastic instance normalization**. Here, we normalize the features of the warped mesh h^i_{warp} with InstanceNorm and get the mean μ^i and standard deviation σ^i. Then, the features of the identity mesh are fed into a convolution layer to get h^i_{id}, which shares the same size with h^i_{warp}. We adopt global average pooling to pool h^i_{warp}, h^i_{id} and concatenate them in channel dimension. A fully-connected layer is employed to compute an adaptive weight w^i. We blend γ^i, σ^i and β^i, μ^i elastically with w^i to get γ' and β'. γ^i and β^i are learned from h^i_{id}. Finally, we scale the normalized h^i_{warp} with γ' and shift it with β'. The value on the parameter flow means the weight.

A.2 Solving OT problem with Sinkhorn algorithm

In this section, we solve the optimal transport (OT) problem defined in Section 3.1 with Sinkhorn algorithm [33]. Following [10], we introduce an entropic regularization term to solve the OT problem efficiently,

$$\begin{aligned} \mathbf{T}_m = \underset{\mathbf{T}\in\mathbb{R}_+^{N_{id}\times N_{pose}}}{\arg\min} & \sum_{ij} \mathbf{Z}(i,j)\mathbf{T}(i,j) + \varepsilon\mathbf{T}(i,j)(\log\mathbf{T}(i,j) - 1) \\ s.t. \quad & \mathbf{T}\mathbf{1}_{N_{pose}} = \mathbf{1}_{N_{id}} N_{id}^{-1}, \quad \mathbf{T}^\top \mathbf{1}_{N_{id}} = \mathbf{1}_{N_{pose}} N_{pose}^{-1}. \end{aligned} \tag{11}$$

where $\mathbf{T}$, $\mathbf{Z}$ and $\mathbf{T}_m$ are the transport matrix, cost matrix and optimal matching matrix respectively, $\mathbf{1}_{N_{id}} \in \mathbb{R}^{N_{id}}$ and $\mathbf{1}_{N_{pose}} \in \mathbb{R}^{N_{pose}}$ are vectors whose elements are all 1, ε is the regularization parameter. The details of the solving process are shown in Algorithm 1.

Algorithm 1 Optimal transport problem with Sinkhorn algorithm.

Input: Cost matrix $\mathbf{Z}$, regularization parameter ε, iteration number i_{max}.

Output: Optimal matching matrix $\mathbf{T}_m$.

$\mathbf{U} \leftarrow \exp(-\mathbf{Z}/\varepsilon)$;
$\mathbf{a} \leftarrow \mathbf{1}_{N_{id}} N_{id}^{-1}$;
for $i = 0, ..., i_{max} - 1$ **do**
  $\mathbf{b} \leftarrow (\mathbf{1}_{N_{pose}} N_{pose}^{-1})/(\mathbf{U}^\top \mathbf{a})$;
  $\mathbf{a} \leftarrow (\mathbf{1}_{N_{id}} N_{id}^{-1})/(\mathbf{U}\mathbf{b})$;
end for
$\mathbf{T}_m \leftarrow \mathrm{diag}(\mathbf{a})\mathbf{U}\mathrm{diag}(\mathbf{b})$.

A.3 More implementation details

In Algorithm 1, we set $\varepsilon = 0.03$ and $i_{max} = 5$. ϵ in Eq. 5 is $1e$-5. We train our model on one RTX 3090 GPU. The training time is about 24 hours on the human data and about 36 hours on the animal data.

B More experimental results

B.1 More results on the human and animal data

In Figure 8 and Figure 9, we show more results generated by our *3D-CoreNet* on SMPL [24] and SMAL [47] respectively.

B.2 Generalization capability

In Figure 10, we show more results generated by *3D-CoreNet* on FAUST [5] and MG-Dataset [4]. To further test the generalization capability of our model, we compare it with Wang et al. [38]. Since DT [35] needs reference meshes as the additional inputs and there are no reference meshes when testing on the new dataset, we do not compare with DT in this section. As we can see, the results generated by our method are more smooth and realistic than Wang et al.. The results generated by Wang et al. always have some artifacts on the arms.

B.3 Shape correspondence

In Figure 11 and Figure 12, we visualize the learned shape correspondence between different meshes, the vertices of the meshes are shuffled randomly before input.

B.4 Robustness to noise

To test the robustness of our model, we add noise to the vertex ordinates of the pose mesh. The results are shown in Figure 13, our model still produces high-quality results.

B.5 Average inference times

In this section, we compare the average inference times for every pose transfer of different methods in the same experimental settings. As shown in Table 4, the traditional deformation transfer method [35] takes the longest time compared to the deep learning-based methods. For [38], they do not learn the correspondence between meshes, so they have the shortest inference time but the generation performance is degraded. 3D-CoreNet achieves notable improvements in generating high-quality results while the inference time is also acceptable. The fourth and the fifth columns show that solving the optimal transport problem takes a very short time while improving the generation results.

Table 4: **Average inference times of different methods**. 3D-CoreNet ($\mathbf{C}$) means 3D-CoreNet with the correlation matrix and 3D-CoreNet ($\mathbf{T}_m$) means 3D-CoreNet with the optimal matching matrix.

Method	DT	Wang et al.	3D-CoreNet ($\mathbf{C}$)	3D-CoreNet ($\mathbf{T}_m$)
Time	3.3352s	0.0068s	0.0124s	0.0131s

[35] [38]

B.6 Limitations

Although our method produces satisfactory results in most cases and has better performance than previous works, there are still some limitations that need to be solved in the future.

For example, when testing on the animal data as shown in Figure 9, the tails of the animals are difficult to handle properly (the third row and the sixth row). When evaluating our model on new

Table 5: **Licenses of the assets used in this paper**.

Data	License websites
SMPL [24]	https://smpl.is.tue.mpg.de/modellicense
SMAL [47]	https://smal.is.tue.mpg.de/license
MG-Dataset [4]	https://github.com/bharat-b7/MultiGarmentNetwork
FAUST [5]	http://faust.is.tue.mpg.de/data_license

identity meshes in Figure 10, if the generated mesh reveals some parts of the human body that were not exposed in the original identity mesh, such as the underarms, it will produce some artifacts (the fourth row).

C Licenses of the assets

The licenses of the assets used in this paper are shown in Table 5. Their licenses are given in the websites.

D Broader impact

We propose a novel method which has potential applications in animated movies and games by generating new poses for existing shapes with less human intervention. Our research can also have a positive impact in the vision and graphics community to inspire new ideas to generate meshes efficiently. However, new meshes generated by the model could be abused to synthesize fake content, which is the negative aspect of this research. Such issues have already drawn a lot of attention from the public. And many approaches have been proposed to solve them technologically and legally.

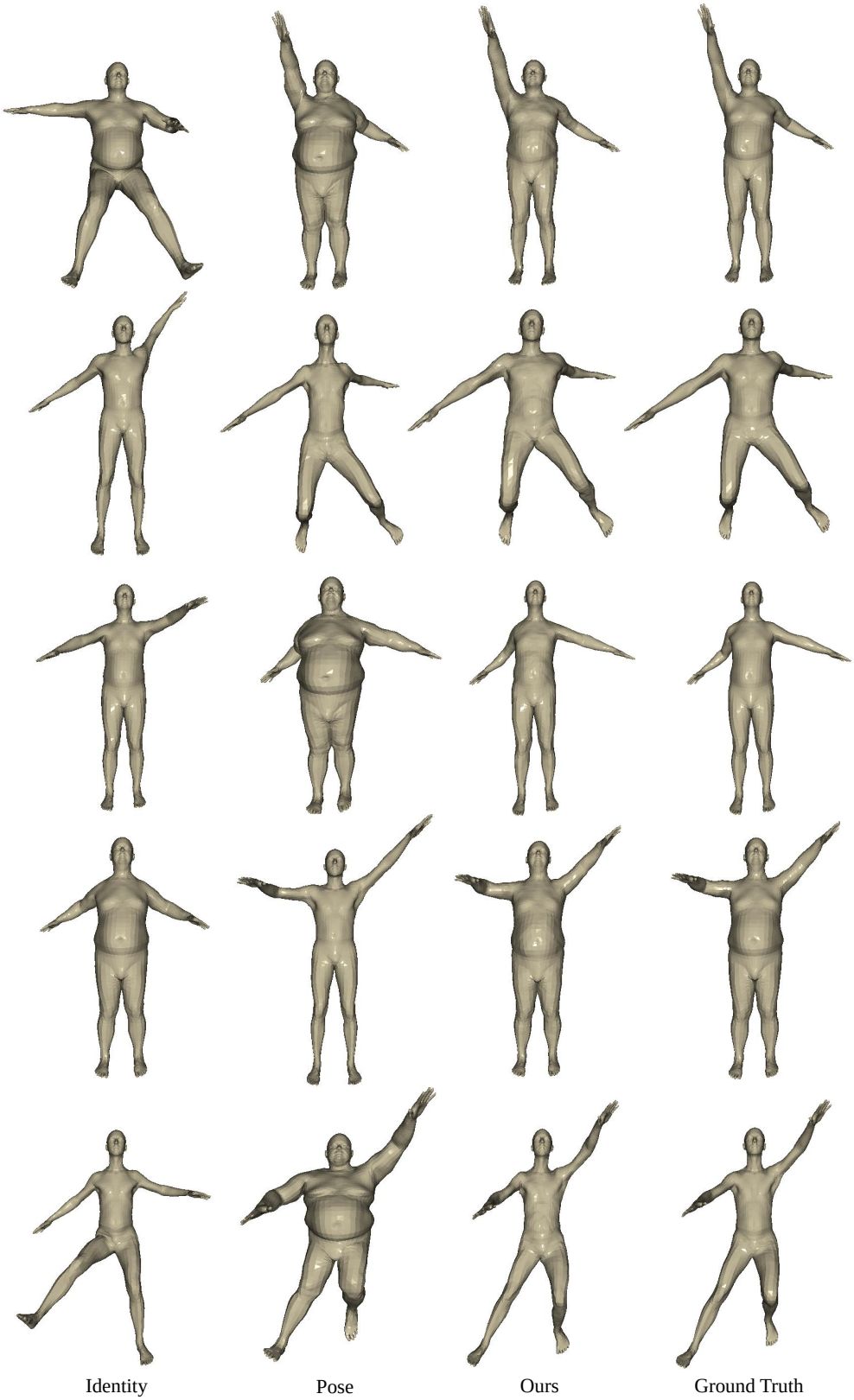

Figure 8: **More results generated by *3D-CoreNet* on SMPL [24].**

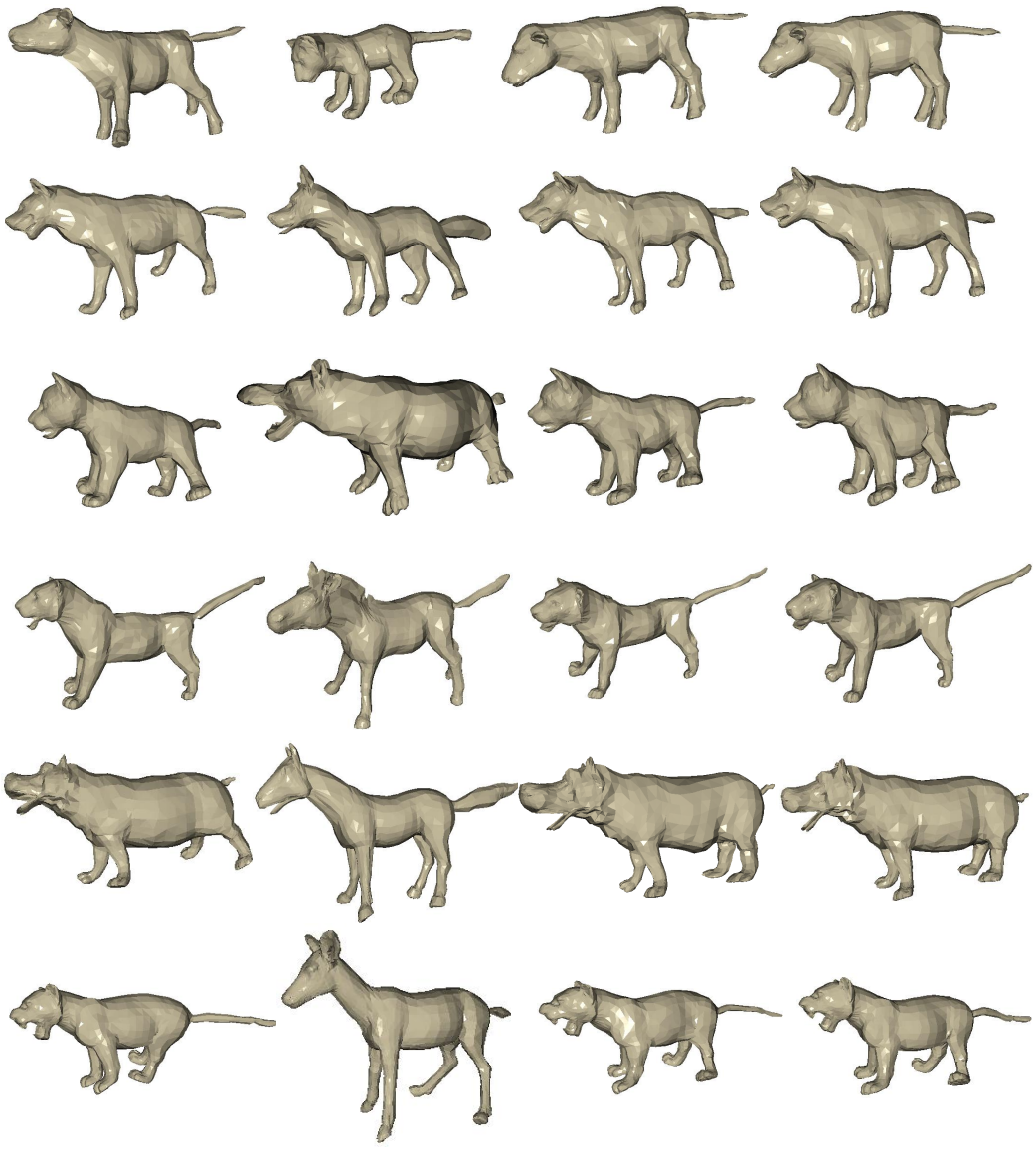

Figure 9: **More results generated by *3D-CoreNet* on SMAL [47].**

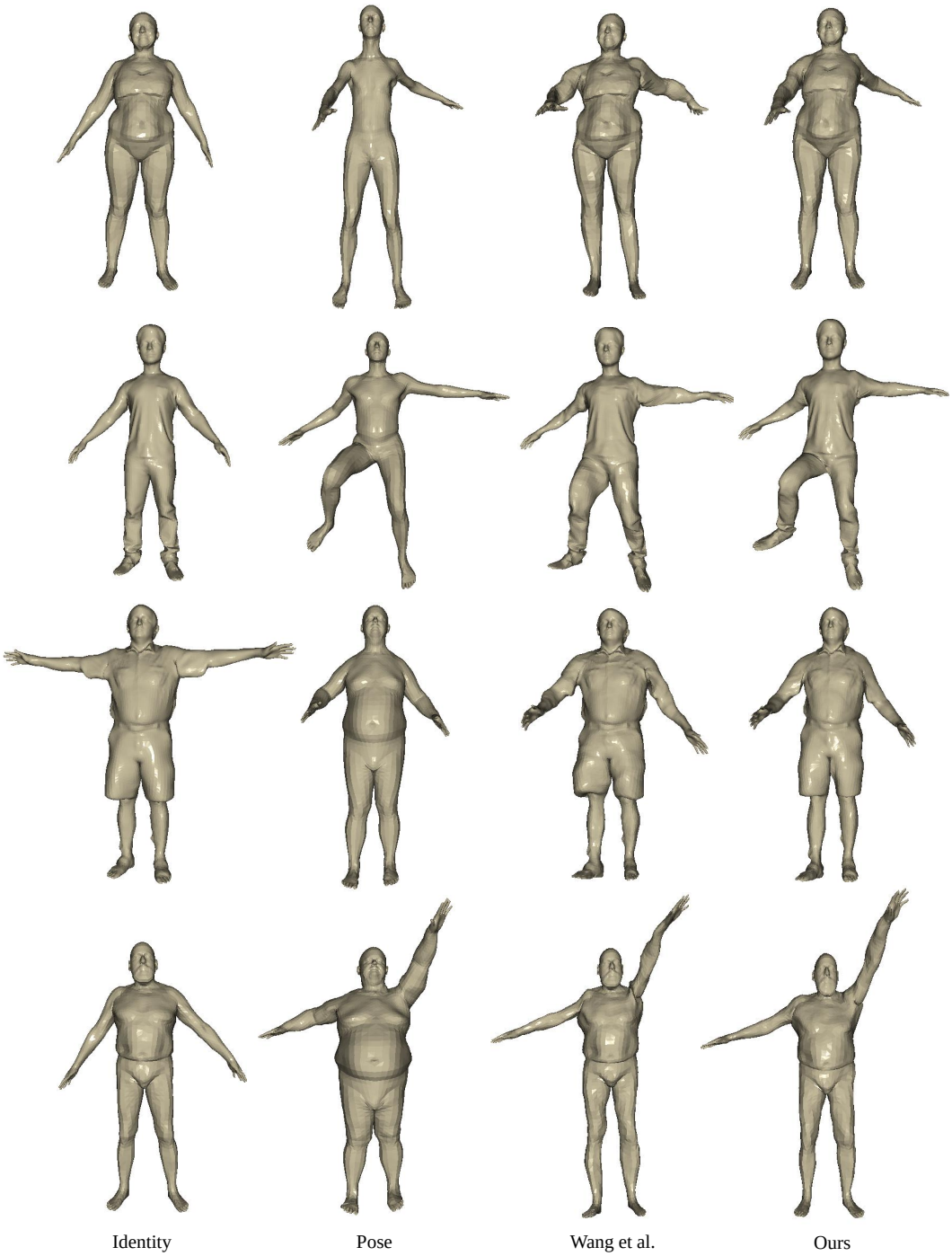

Figure 10: **More results generated by *3D-CoreNet* on FAUST [5] and MG-Dataset [4].** The identity meshes are from FAUST and MG-Dataset. We compare our method with Wang et al. [38] to test the generalization capability of our 3D-CoreNet.

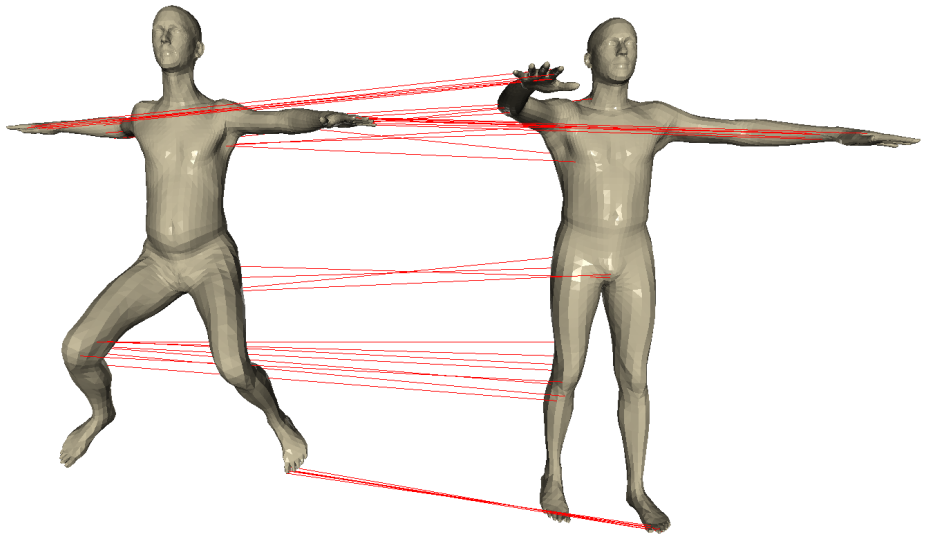

Figure 11: **Visualization of the learned correspondence between different human meshes**. We select several corresponding vertices on two human meshes to visualize. The vertices of the meshes are shuffled randomly before input.

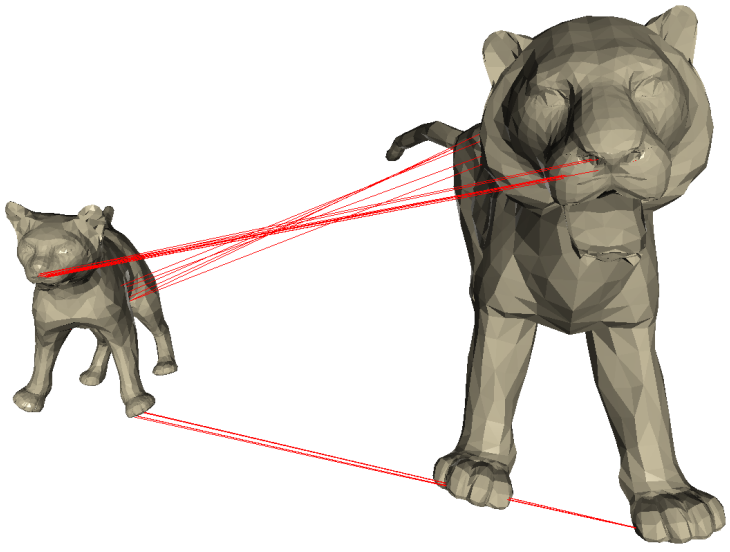

Figure 12: **Visualization of the learned correspondence between different animal meshes**. We select several corresponding vertices on two animal meshes to visualize. The vertices of the meshes are shuffled randomly before input.

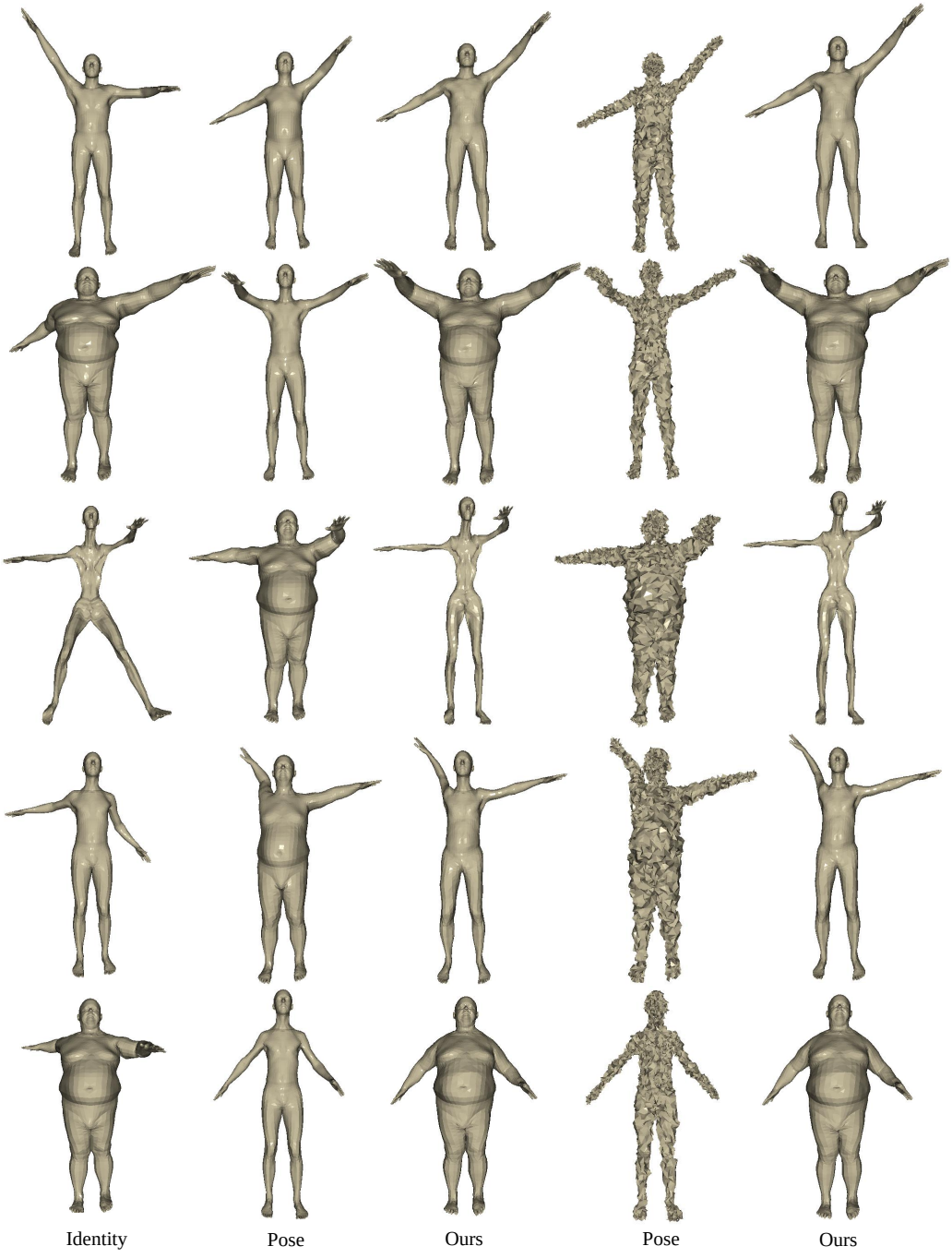

Figure 13: **Robustness to noise**. Here, we add noise to the pose mesh. Our model can still produce high-quality results.