Supplementary Materials for "Neural Pose Transfer by Spatially Adaptive Instance Normalization"

Jiashun Wang[1*‡] Chao Wen[1*§] Yanwei Fu[1†‡] Haitao Lin[1] Tianyun Zou[1] Xiangyang Xue[1] Yinda Zhang[2†]

[1]Fudan University [2]Google LLC

We provide details about network architecture, implementation details, comparison to more baselines, model analysis, and more results on various datasets.

A. Network Architecture

The network architecture is shown in Tab. 3, where N is the batch size and V is the number of vertices. Our network consists of two main parts - the pose feature extractor (1-9) and the style transfer decoder for pose transfer (10-17). Both components are composed of 1×1 convolution and instance normalization. The detailed architecture of SPAdaIN Resnet Block and SPAdaIN unit are given in Tab. 1 and Tab. 2.

Different from most of other work that uses batch normalization, we use instance normalization. Specifically, we consider our input 3D mesh $\boldsymbol{M} \in \mathbb{R}^{N\times 3\times V}$ as a tensor and apply normalization individually for each training instance along the spatial dimension V. Furthermore, as mentioned in Sec 3.3 of the main submission, we learn the parameters $\gamma \in \mathbb{R}^{N\times C\times V}$ and $\beta \in \mathbb{R}^{N\times C\times V}$ of InstanceNorm which keep the spatial information.

Index	Inputs	Operation	Output shape
(1)	Input	Identity Mesh	N×3×V
(2)	Input	Input Features	N×C×V
(3)	(1), (2)	SPAdaIN 1 (C=C)	N×C×V
(4)	(3)	conv1d(C→C, 1×1), Relu	N×C×V
(5)	(1), (4)	SPAdaIN 2 (C=C)	N×C×V
(6)	(5)	conv1d(C→C, 1×1), Relu	N×C×V
(7)	(1), (2)	SPAdaIN3 (C=C)	N×C×V
(8)	(7)	conv1d(C→C, 1×1), Relu	N×C×V
(9)	(5), (8)	Add	N×C×V

Table 1: The network architecture for SPAdaIN Res-Block.

Index	Inputs	Operation	Output shape
(1)	Input	Identity Mesh	N×3×V
(2)	Input	Input Features	N×C×V
(3)	(1)	conv1d(3→C, 1×1)	N×C×V
(4)	(1)	conv1d(3→C, 1×1)	N×C×V
(5)	(2)	Instance Norm	N×C×V
(6)	(3), (5)	Multiply	N×C×V
(7)	(4), (6)	Add	N×C×V

Table 2: The network architecture for SPAdaIN unit.

B. Data Preparation

We prepare our training and testing data using SMPL [8] model. SMPL [8] model has 10 morphology parameters controlling the shape and 24 sets of joint parameters controlling the pose. For shape parameters, we randomly sample from the parameter space. For pose parameters, each set of parameters has three sub-parameters represented as a tuple (x, y, z), indicating rotated joint angle around x-axis, y-axis and z-axis respectively. In order to generate natural looking poses, we constrain the rotation angle of the joints according to what human joints can physically reach. Then we sample from the constrained angle space. The details of the range can be seen in Tab. 5.

C. Comparison to Baselines

In this section, we design and evaluate some competitive baselines.

C.1. Comparison to Skeleton Pose Driven Approach

We compare our method with skeleton-based skinning shape deformation. We first extract human pose skeleton from both the pose and identity meshes by fitting an SMPL [8] model. We take the T-pose SMPL as the initialization, and update the SMPL parameters through gradient descent using LBFGS [7]. We use the joints of this fitted model as the key points of our skeleton representation. We then calculate the binding weights of LBS (Linear Blend Skinning) [9, 6, 4, 5] using tools from Baran *et al*. [1]. After that, we transform the identity skeleton to the pose skeleton.

*indicates equal contributions.
†indicates corresponding author.
‡Yanwei Fu and Jiashun Wang are with School of Data Science, and MOE Frontiers Center for Brain Science, Shanghai Key Lab of Intelligent Information Processing Fudan University.
§Chao Wen is with Academy of Engineering and Technology, and Institute of AI and Robotics, Fudan University

Index	Inputs	Operation	Output Shape
(1)	Input	Identity Mesh	N×3×V
(2)	Input	Pose Mesh	N×3×V
(3)	(1)	conv1d(3→64, 1 × 1)	N×64×V
(4)	(3)	Instance Norm, Relu	N×64×V
(5)	(4)	conv1d(64→128, 1 × 1)	N×128×V
(6)	(5)	Instance Norm, Relu	N×128×V
(7)	(6)	conv1d(128→1024, 1 × 1)	N×1024×V
(8)	(7)	Instance Norm, Relu	N×1024×V
(9)	(2), (8)	Concatenate	N×1027×V
(10)	(9)	conv1d(1027→1027, 1 × 1)	N×1027×V
(11)	(10)	SPAdaIN ResBlk 1 (C=1027)	N×1027×V
(12)	(11)	conv1d(1027→513, 1 × 1)	N×513×V
(13)	(12)	SPAdaIn ResBlk 2 (C=513)	N×513×V
(14)	(13)	conv1d(513→256, 1 × 1)	N×256×V
(15)	(14)	SPAdaIN ResBlk 3 (C=256)	N×256×V
(16)	(15)	conv1d(256→3, 1 × 1)	N×3×V
(17)	(16)	tanh	N×3×V

Table 3: The network architecture for our full model.

Pose Source	PMD ↓ ($\times 10^{-4}$)		
	skeleton	maxpooling	ours
seen-pose	27.4	2.1	1.1
unseen-pose	31.1	12.7	9.3

Table 4: **Quantitative comparison to other baselines.**

Since the skeleton joints of SMPL model assemble a kinematic tree, we calculate the transformation matrix between two skeletons according to the connection relationship of the joints through the local coordinate system. Finally, we recover the mesh from skeleton using the binding weights computed before.

We show the quantitative result in Tab. 4. According to the table, the skeleton based approach cannot perform as well as our method due to the accumulated error at each stage. Particularly, this method has trouble dealing with varying limb length caused by body shape variations. Qualitative evaluation is shown in Fig. 1. The skeleton based deformation approach often produces artifacts near joint points, due to different limb lengths.

C.2. Comparison to Compact Pose Feature

We also create a strong deep learning baseline. Instead of maintaining the per-vertex feature on the pose mesh, we apply a global max pooling as suggested in PointNet to extract a compact global pose feature. This feature is then concatenated with each vertex in the identity mesh, and further fed into the decoder. Note that we need to remove the first instance normalization in the decoder to make this work, otherwise the instance normalization would whitening all the pose feature as they are exactly the same on all the vertices.

The quantitative result is shown in Tab. 4. As can be seen, this baseline works much better than the skeleton based deformation, but not as good as our method. One possible reason could be that the global max pooling may drop some fine-grained information from the pose mesh which is helpful for pose transfer.

D. More Qualitative Results

In this section, we show more qualitative results to demonstrate the robustness and generalization capability of our system.

D.1. Invariance to Vertex Order

To the best of our knowledge, our model is the first one that achieves permutation invariance on the order of vertices in both input meshes. That says, the identity mesh can be provided in arbitrary pose and vertex order. We verify the model behavior with random permutation, and the results are shown in Fig. 3. For each example one the left and right, we randomly shuffle the vertex order in both the identity and pose mesh, and feed them into the same network (we use the color to encode the vertex order). As can be seen, our network successfully produces visually the same target mesh with correct identity and pose. Note that for each random shuffle, the output vertex order is the same as the identity mesh. This indicates that the deformed mesh are point-wise aligned with the initial identity mesh, which can be very useful for many graphics applications, *e.g.* texture transfer.

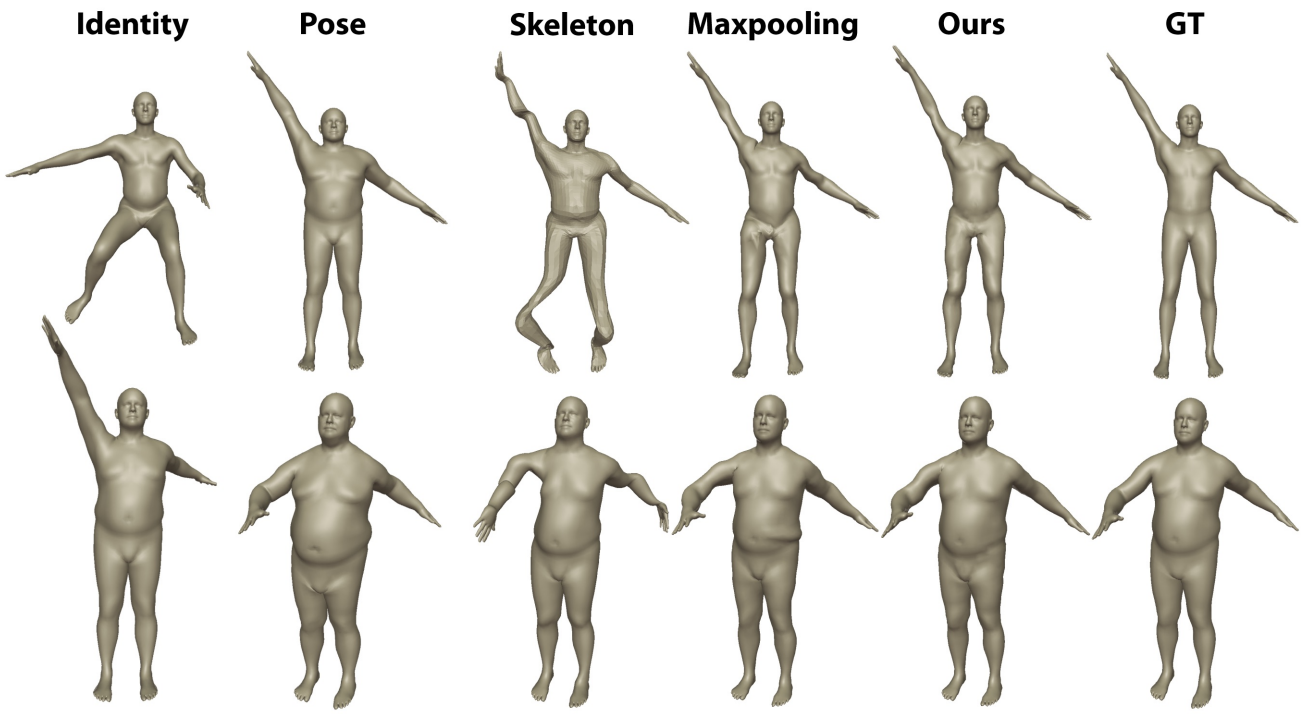


Figure 1: **Qualitative comparison to other baselines.** From left to right, we show in each row: input identity mesh, input pose mesh, the results of skeleton pose driven approach, the results of max pooling method, our results and the ground truth. We have more accurate results.

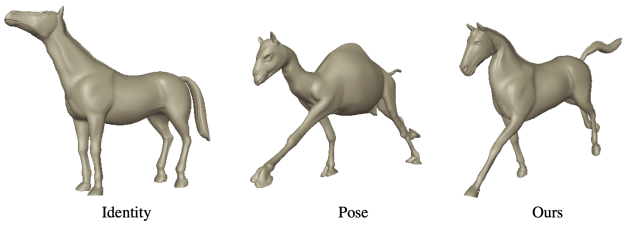


Figure 2: **Camel and horse pose transfer.**

D.2. Robustness to Pose Mesh Noise

We test the robustness of our system given noisy mesh. In Fig. 4, we provide our model pose meshes in the same pose but with different shape and noise level. Our network successfully extracts the correct pose information and produces final output mesh in the correct pose.

D.3. Generalization to New Identity

We also test the generalization capability of our model to unseen identities, especially those non-SMPL model meshes. In Fig. 5, we show more results on identity meshes from FAUST dataset [3]. Our model generalizes to these meshes automatically without any finetune. Features that not measured by SMPL, such as the mustache of the man in the first row, are successfully maintained.

We also try more challenging cases using meshes from MG-dataset [2]. The meshes in this dataset contains apparel, which are more different with SMPL meshes compared to those from the FAUST [3]. As can be seen in Fig. 6, though with some small artifacts, our model still maintains the identity, *i.e*. person and apparel, correctly.

D.4. Our Results on Seen and Unseen Poses

We show more qualitative results of our model on seen and unseen poses in Fig. 7 and Fig. 8 respectively.

D.5. Our Results on Non-Human Models

In the end, we show the results of our model on transferring pose from camel to horse in Fig. 2, by training on the animal dataset [10]. We adopt the compact pose feature encoder to handle different vertices number between identity mesh and pose mesh, and then using our decoder to transfer pose for non-human meshes. Even though we specifically focus on human, our model also works for non-human meshes but require domain-specific training.

Parameter Index	Rotation Degree of Axes		
	x-axis	y-axis	z-axis
1	(-2,2)	(-2,2)	(-2,2)
2	(-90,0)	0	(0,40)
3	(-90,0)	0	(-40,0)
4	(-1,1)	(-1,1)	(-1,1)
5	(0,100)	0	0
6	(0,100)	0	0
7	(-1,1)	(-1,1)	(-1,1)
8	(-10,10)	(-10,10)	(-1,1)
9	(-10,10)	(-10,10)	(-1,1)
10	(-1,1)	(-1,1)	(-1,1)
11	(-1,1)	(-1,1)	(-1,1)
12	(-1,1)	(-1,1)	(-1,1)
13	(-3,3)	(-3,3)	(-3,3)
14	0	(-30,30)	(-30,30)
15	0	(-30,30)	(-30,30)
16	(-3,3)	(-3,3)	(-3,3)
17	0	(-30,30)	(-30,30)
18	0	(-30,30)	(-30,30)
19	0	(-60,0)	0
20	0	(0,60)	0
21	(-10,10)	(-10,10)	(-10,10)
22	(-10,10)	(-10,10)	(-10,10)
23	(-5,5)	(0,10)	(-10,0)
24	(-5,5)	(-10,0)	(0,10)

Table 5: **Pose parameters preparation.** Human posture can be easily adjusted by rotating 24 key joints represented as parameter index. We give more details of the range of angles of each pose parameter. We randomly sample in this pose space to generate our input data.

References

[1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. In *ACM Transactions on graphics (TOG)*, volume 26, page 72. ACM, 2007. 1

[2] Bharat Lal Bhatnagar, Garvita Tiwari, Christian Theobalt, and Gerard Pons-Moll. Multi-garment net: Learning to dress 3d people from images. In *IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct 2019. 3, 8

[3] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 3, 7

[4] Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (TOG)*, 31(4):77, 2012. 1

[5] Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011. 1

[6] John P Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000. 1

[7] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 1

[8] 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):248, 2015. 1

[9] Paul Molodowitch. The pinocchio auto-rigging / weighting tool. https://github.com/elrond79/Pinocchio. 1

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

Figure 3: **More qualitative results of different vertex order.** Each row represents one vertices order with respect to input identity mesh, pose mesh and our results. Our results are consistent with the identity mesh. Different orders of the inputs do not affect the output visually. For all meshes, the vertex color represents the vertex order. For the first 3 rows, vertices are colored according to the *index→color mapping*, and for the last row, vertices are colored according to the *value of the index*.

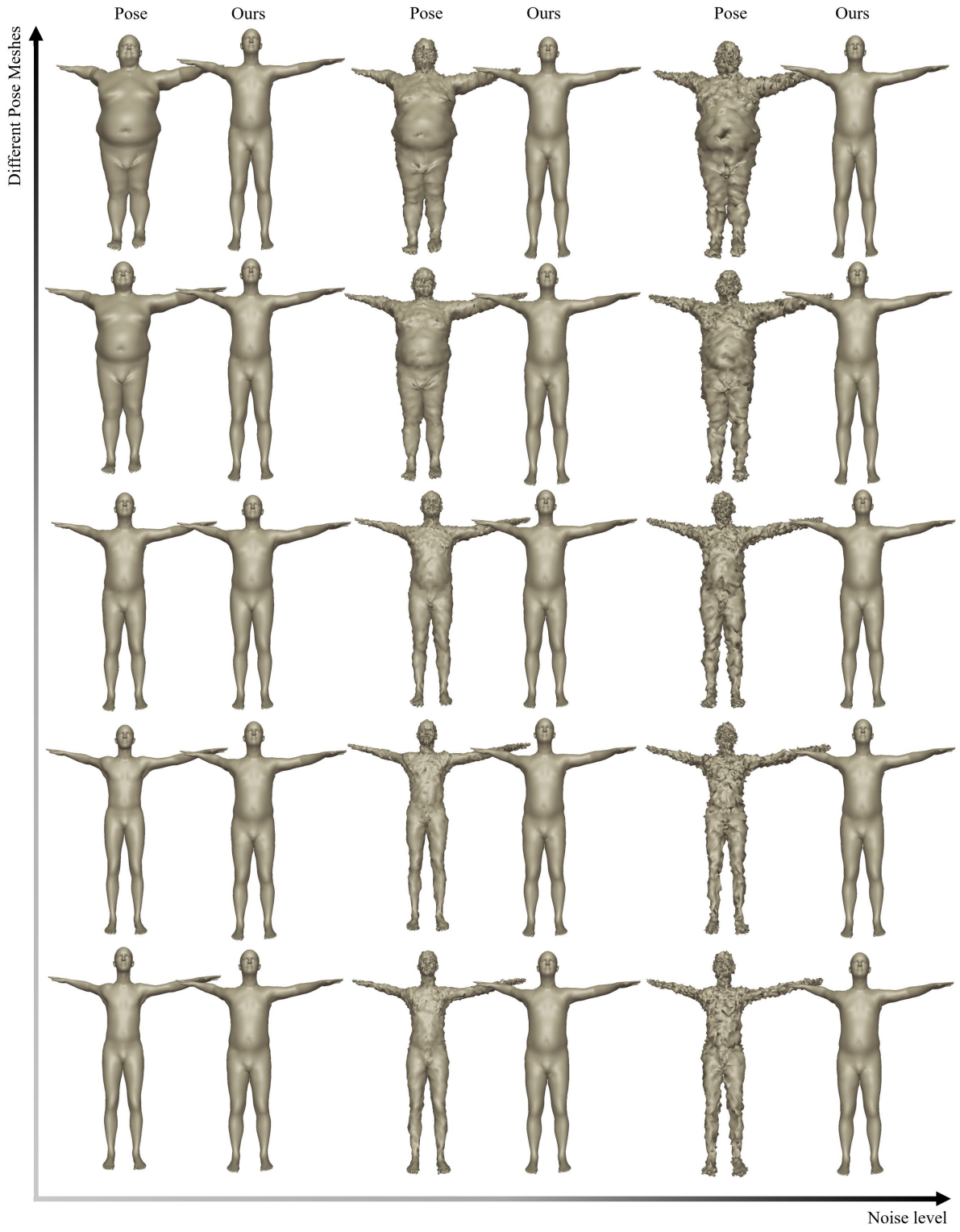
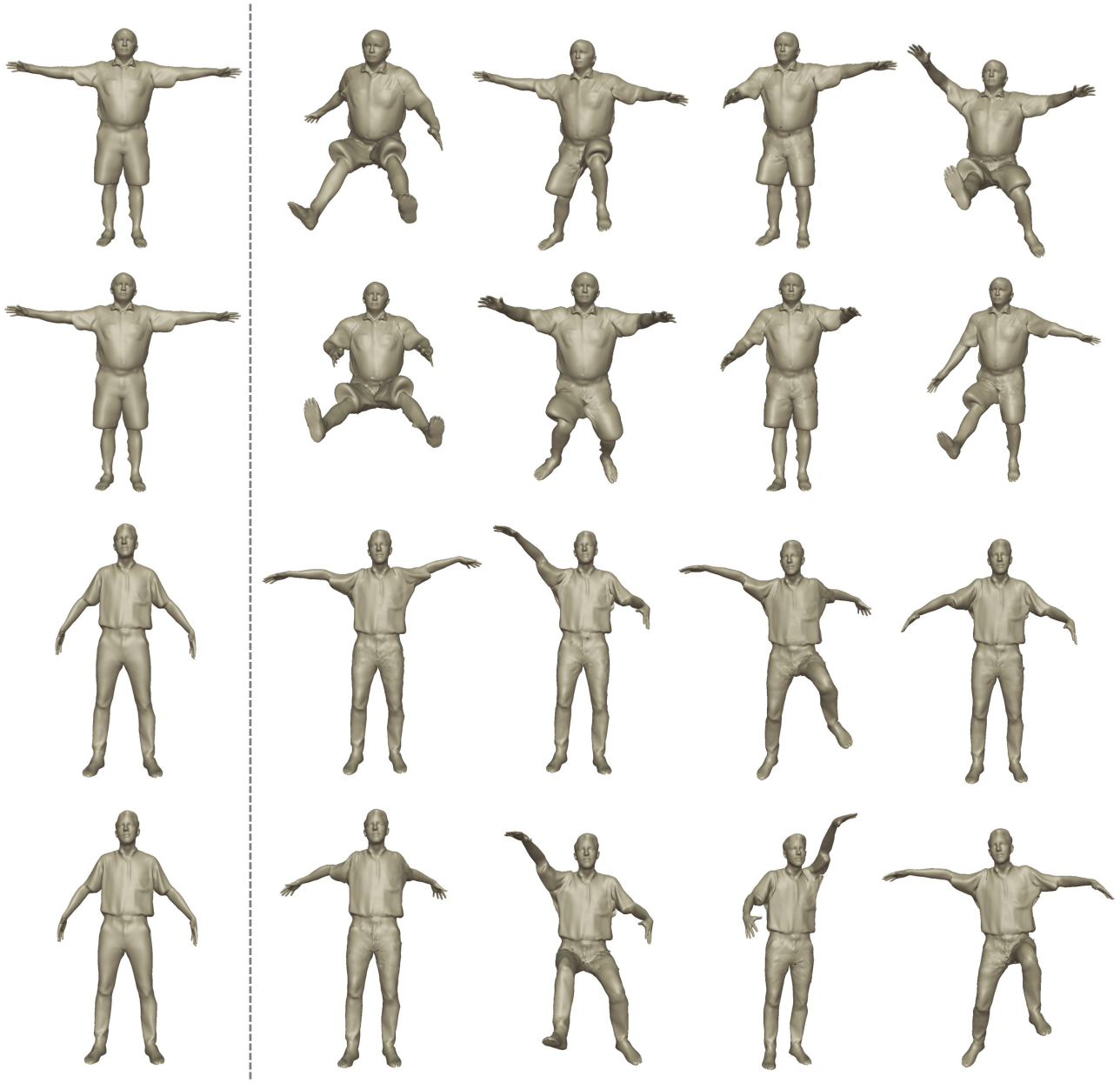


Figure 4: **Robustness to different pose meshes.** The pose meshes in the same pose with different shape or the pose meshes with different level of noise would not influence the result. Our method can produce similar and correct output.

(a) Identity (b) Ours

Figure 5: **More examples of identity mesh from FAUST [3].** (a) Identity input meshes. (b) Output meshes using our methods. Our method can deform the identity mesh to various poses with good visual quality.



(a) Identity (b) Ours

Figure 6: **More examples of identity mesh from MG-dataset [2].** (a) Identity input meshes (b) Output meshes using our methods. Though the appearance of MG-dataset [2] wearing clothes is quite different from meshes of SMPL, our method can still produce very good results.

Figure 7: **More examples of seen poses.** From left to right, we show in each row: input identity mesh, input pose mesh, the results of ours and the ground truth.

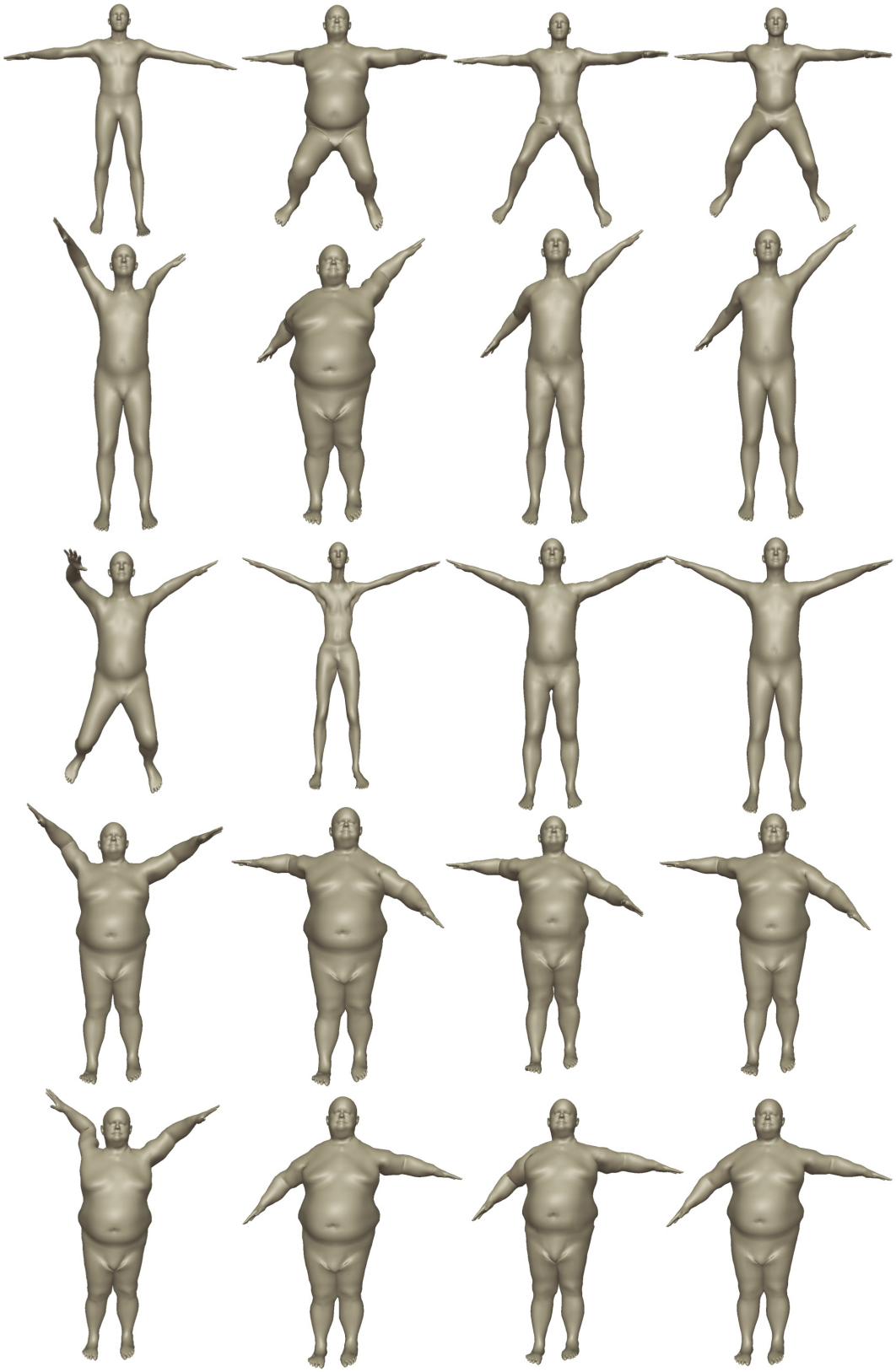


Figure 8: **More examples of unseen poses.** From left to right, we show in each row: input identity mesh, input pose mesh, the results of ours and the ground truth.