SMVRT: Implicit Human 3D Modeling Using Sparse Multi-View Volumetric Reconstruction with Transformer Fusion

Supplementary Material

1. Introduction

This supplementary material is organized as follows: Section 2 explains the detailed explanations of the evaluation metrics. Section 3 explains the data preparation process. Section 5 makes a runtime analysis, including memory usage and frame rates. Section 6 demonstrates additional results on THUman2.1 and MultiGarment datasets.

2. Metrics

We evaluate reconstruction performance with several metrics, including CD, IOU, NC, P2S and F-Scores. More specifically, for **Chamfer Distance (CD)**, we first sample points on both the reconstructed and the ground truth surface to approximate the surfaces with point clouds. The Chamfer Distance between two meshes, represented by point clouds P_a and P_b, is measured as the sum of the average of the minimum distances from P_a to P_b and from P_b to P_a. In this paper, following ONet [6] and IFNet [2], we compute both CD_{l1} and CD_{l2}:

$$\begin{aligned} Chamferl1(P_a, P_b) &= \frac{Completeness}{2|P_a|} + \frac{Accuracy}{2|P_b|} \\ Chamferl2(P_a, P_b) &= \frac{Completeness^2}{2|P_a|} + \frac{Accuracy^2}{2|P_b|} \end{aligned} \tag{1}$$

where

$$\begin{aligned} Completness &= \sum_{p_a \in P_a} \min_{p_b \in P_b} d(p_a, p_b) \\ Accuracy &= \sum_{p_b \in P_b} \min_{p_a \in P_a} d(p_b, p_a) \\ Completness^2 &= \sum_{p_a \in P_a} \min_{p_b \in P_b} d(p_a, p_b)^2 \\ Accuracy^2 &= \sum_{p_b \in P_b} \min_{p_a \in P_a} d(p_b, p_a)^2 \end{aligned} \tag{2}$$

Point-to-surface (P2S) is the average of $Accuracy^2$, calculated as follows:

$$P2S(P_a, P_b) = \frac{Accuracy^2}{|P_b|} \tag{3}$$

where P_a is the sampled points from ground truth mesh, and P_b is the sampled points from predicted mesh.

Normal Consistency (NC). The normal consistency between two point clouds P_a and P_b is defined by the following equation:

$$\begin{aligned} NC(P_a, P_b) = &\frac{1}{2|P_a|} \sum_{p_a \in P_a} N_{p_a} N_{nearp_a, P_b} \\ &+ \frac{1}{2|P_b|} \sum_{p_b \in P_b} N_{p_b} N_{nearp_b, P_a} \end{aligned} \tag{4}$$

where N_{nearp_a, P_b} is the nearest point of p_a of P_a in point cloud P_b. And N_p is the normal of point p on the mesh.

F-Score (FS). F-Score between the two point clouds P_a and P_b given a threshold t is defined as follows:

$$F - Score(P_a, P_b, t) = \frac{2Recall \cdot Precision}{Recall + Precession} \tag{5}$$

where

$$Recall(P_a, P_b, t) = |p_a \in P_a, s.t. \min_{p_b \in P_b} d(p_a, p_b)| \tag{6}$$

$$Precision(P_a, P_b, t) = |p_b \in P_b, s.t. \min_{p_a \in P_a} d(p_b, p_a)| \tag{7}$$

We report F0.5, F1.0, F1.5 by setting t = 0.5% 1.0%, 0.5%.

Intersection over Union (IoU) evaluates the volumetric alignment between the predicted and ground truth meshes. Specifically, a large number of points are sampled within the unit cube of the reconstruction volume, and the number of points that lie inside or outside both meshes is counted. The IoU is then computed as :

$$IoU(M_a, M_b) = \frac{TP}{TP + FP + FN} \tag{8}$$

where TP (true positives) denotes the number of points correctly predicted as inside the mesh, FP (false positives) represents points incorrectly predicted as inside while actually outside, and FN (false negatives) represents points incorrectly predicted as outside while actually inside the ground truth mesh. For IoU computation, we uniformly sample one million points within the reconstruction unit volume.

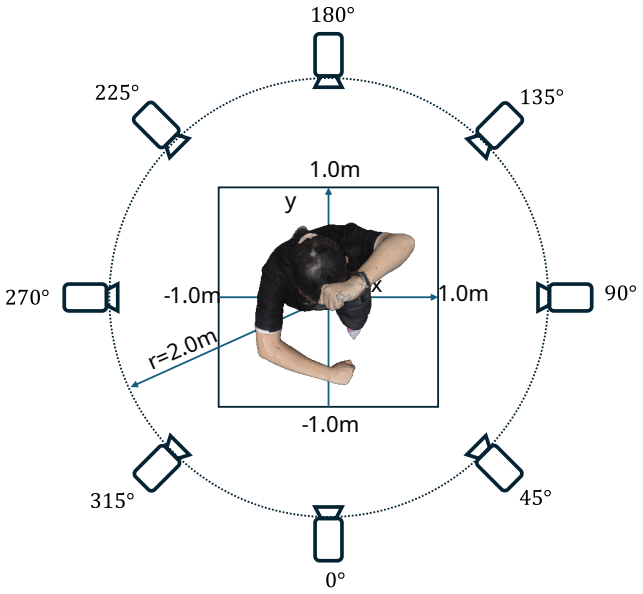

Figure 1. Rendering setup. Eight cameras are evenly placed on a circle path around the center stage. They are located at elevation angle 0° and alzimuth angle $0^\circ, 45^\circ, 90^\circ, 135^\circ, 180^\circ, 225^\circ, 270^\circ, 315^\circ$. The center square stage covers [-1.0m, 1.0m]. Rendered image resolution is set to 512×512 for our standard experiments and 1024×1024 for high resolution experiments. In our experimental configurations, images from 0° and 180° are used in the two inputs experiment. Images from 0°, 90° and 180° and 270° are for four inputs. All these eight images are used in experiment with eight inputs.

3. Data and processing

For training, we prepare three types of data for a given mesh object:

1. Eight posed images are rendered from ground truth color mesh using predefined camera poses. Figure 1 illustrate the rendering system configuration. The mesh height is normalized to [-0.9m, 0.9m], fitting into the volume of size [-1.0m, 1.0m] along x, y, z dimensions. We use Blender [3] for rendering. The image resolution is set to $H \times W = 512 \times 512$ for standard experiments and $H \times W = 1024 \times 1024$ for high resolution experiments.

2. K query points are sampled around the ground truth mesh. First, we uniformly sample K surface points on the mesh. Each surface point p is then then perturbed by isotropic Gaussian noise $n \sim N(0, \Sigma)$, generating a query point $q = p + n$. The covariance matrix $\Sigma \in R^{3\times3}$ is diagonal, with $\Sigma_{0,0} = \Sigma_{1,1} = \Sigma_{2,2} = \sigma$, defining the displacement scales. Three sets of query 500,000 points $K1, K2$, and $K3$ are generated with $\sigma = 0.003, 0.020, 0.300$, respectively. Figure 2 shows the sampled query points. We then randomly select 15%, 35%, and 50% from $K1, K2$, and $K3$ correspondingly, i.e., $K = 0.15\times K1 + 0.35\times K2 + 0.50\times K3$.

3. Ground truth occupancy, which indicates whether each query point lies inside or outside the ground truth mesh for the occupancy field. Points inside the mesh are labeled as 0, while points outside are labeled as 1.

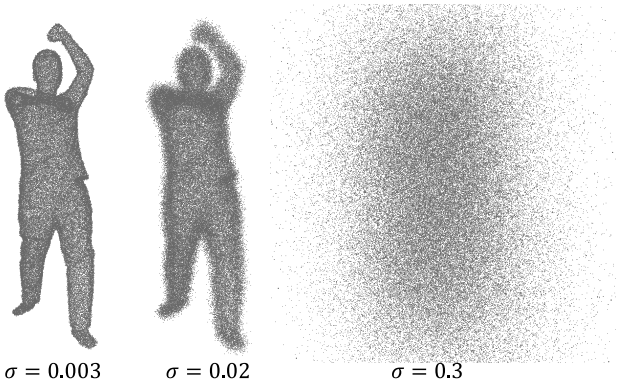

Figure 2. Illustration of query point sampling. Surface points are sampled on the mesh and displaced with gaussian noise. The standard deviation of gaussian are $\sigma = 0.003, 0.020, 0.300$, respectively.

4. Network structure

Here we list the network module details including parameters, input shapes, layers, output shapes. Please refer to table 1 for the summary.

5. Memory and runtime analysis

We analyze the memory footprint and runtime of our model. Table 2 compares our model with baseline methods. Table 3 reports the memory usage and runtime for each block in the reconstruction pipeline. We observe that majority of computational time is spent on implicit field decoding, highlighting a potential area for future exploration to improve reconstruction speed.

6. More results

THUman2.1: Figure 3 - 9 show more qualitative comparison results among MV-PIFu, Zins et al. and ours on THUman2.1.

Figure 10 - 14 show more qualitative results on THUman2.1. From left to right they are rendered at 0°, 90°, 180°, and 270° viewing angles with surface normals visualized as colors.

Figure 15 - 19 show more qualitative results on THUman2.1 with four cameras and eight cameras. Each figure includes four quadrants with three rows per quadrant: ground truth, reconstruction with four cameras, reconstruction with eight cameras. Renderings are shown at 0°, 90°,

Modules	Submodules	Input	Layers	Output
Resnet34	Conv2d	(B×V) ×13×512×512	Conv2d(13,64, (3,3))	(B×V) ×64×512×512
	Layer1	(B×V) ×64×512×512	*	(B×V) ×64×256×256
	Layer2	(B×V) ×64×256×256	*	(B×V) ×128×128×128
	Layer3	(B×V) ×128×128×128	*	(B×V) ×256×64×64
	Layer4	(B×V) ×256×64×64	*	(B×V) ×512×64×64
AFM	Linear	(B×V) ×512× (64×64)	Linear(512,256)	(B×V) ×256× (64×64)
	Transformer/Mamba	B×256× (V×(64×64+2))	**	B×256× (V×(64×64+2))
	Transformer/Mamba	(B×V) ×256× (64×64+2)	**	(B×V) ×256× (64×64+2)
	Transformer/Mamba	B×256× (V×(64×64+2))	**	B×256× (V×(64×64+2))
	Transformer/Mamba	(B×V) ×256× (64×64+2)	**	(B×V) ×256× (64×64+2)
MMFT	Transformer	(B×G) ×451×4	***	(B×G) ×451×1
3D Unet	Linear	B×451×64×64×64	Conv3d(451,64, (3,3,3))	B×64×64×64×64
	Block1	B×64×64×64×64	†	B×64×64×64×64
	Block2	B×64×32×32×32	†	B×128×32×32×32
	Block3	B×128×16×16×16	†	B×256×16×16×16
	Block4	B×256×8×8×8	†	B×256×8×8×8
	Block5	B×256×32×32×32	†	B×128×32×32×32
	Block6	B×128×64×64×64	†	B×64×64×64×64
QFT	Transformer	(B×N) ×513×4	‡	(B×N) ×513×1
Field decoder	Linear	B×516×N	Linear(516,512)	B×512×N
	Linear	B×512×N	Linear(512,256)	B×256×N
	Linear	B×256×N	Linear(256,256)	B×256×N
	Linear	B×256×N	Linear(256,1)	B×256×N

Table 1. Network modules and parameters. The input to the network consists of 13 channels, with V denoting the number of views. For ResNet34, we only modified the first layer to accept multi-modal inputs. *: Refer to [5] for the original structure. **: AFM module first projects patches to a predefined dimension of 256 using a linear layer. It applies two groups of alternating fusion blocks to fuse 2D features globally and locally. The sequence of blocks performs global, frame, global, frame fusion respectively. These blocks can be implemented as transformer blocks or Mamba blocks. Refer to [8] for transformer details, and [4] for Mamba structure. MMFT is to fuse 2D features to form 3D feature volumes. It consists of one layer of common transformer *** [8]. $G = 64 \times 64 \times 64$. After fusion, features are reshaped to volume format for downstream 3D regularization. 3D U-Net is applied to the 3D feature volume. We first reduce the feature size from 451 to 64 to reduce the memory usage. And a series of grouped blocks † [Conv3d-batch normalization-ReLU, Conv3d-batch normalization-ReLU] are applied to regularize features. Down-sampling, up-sampling and skip connections follow common U-Net designs. QFT fuses 2D and 3D features for N query points, using the same structure ‡ as MMFT. N is set to 500,000 points in training and set to 256^3 for inference at volume grid of resolution 256^3. The field decoder is composed of a series of linear layers that infer occupancy for each query point.

	Model(MB)	Time (S)
MV-PIFu [7]	59.5	5.86
Zins et al. [10]	94.8	11.98
Ours w/2 cameras	114.2	7.76
Ours w/4 cameras	114.2	19.95
Ours w/8 cameras	114.2	22.21

Table 2. Runtime comparison (in seconds) and memory usage (in megabytes) for MV-PIFu [7] and Zins et al. [10]. All baselines use four cameras.

	Model(MB)	Time(S)
Normal estimation(Sapiens)	1148	1.70
Depth estimation(Sapiens)	1148	1.70
Feature encoding	109.45	0.17
Implicit field decoding	4.77	19.50
Marching cube(CPU)	-	0.28

Table 3. Runtime analysis (in seconds) and memory usage (in megabytes) for each stage of our network using four camera inputs. Feature encoding includes AFM and MMFT. Meshing is performed on a 256^3 query grid. Please refer to the main paper for details on each component. From the numbers, we could conclude that most time spends on the implicit field decoding, which is caused by the large amount query points of grid resolution 256^3.

180°, and 270° viewing angles, with surface normals visualized as colors. Please zoom in to see fine details.

MultiGarment:Figure 20 - 24 show more qualitative results on MultiGarment [1].

MultiHuman:Figure 26 - 30 show qualitative results on MultiHuman [9].

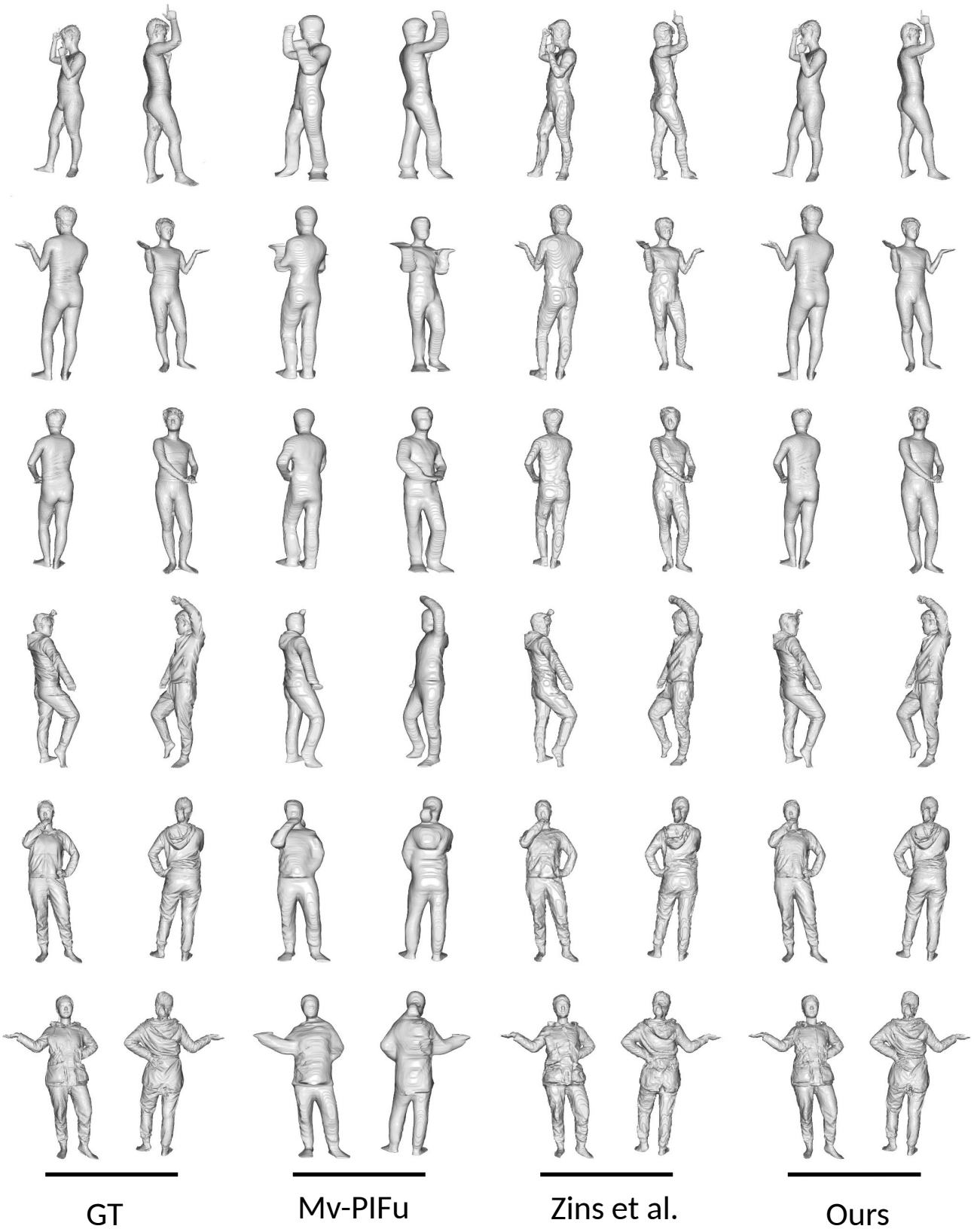

Figure 3. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al., and our method.

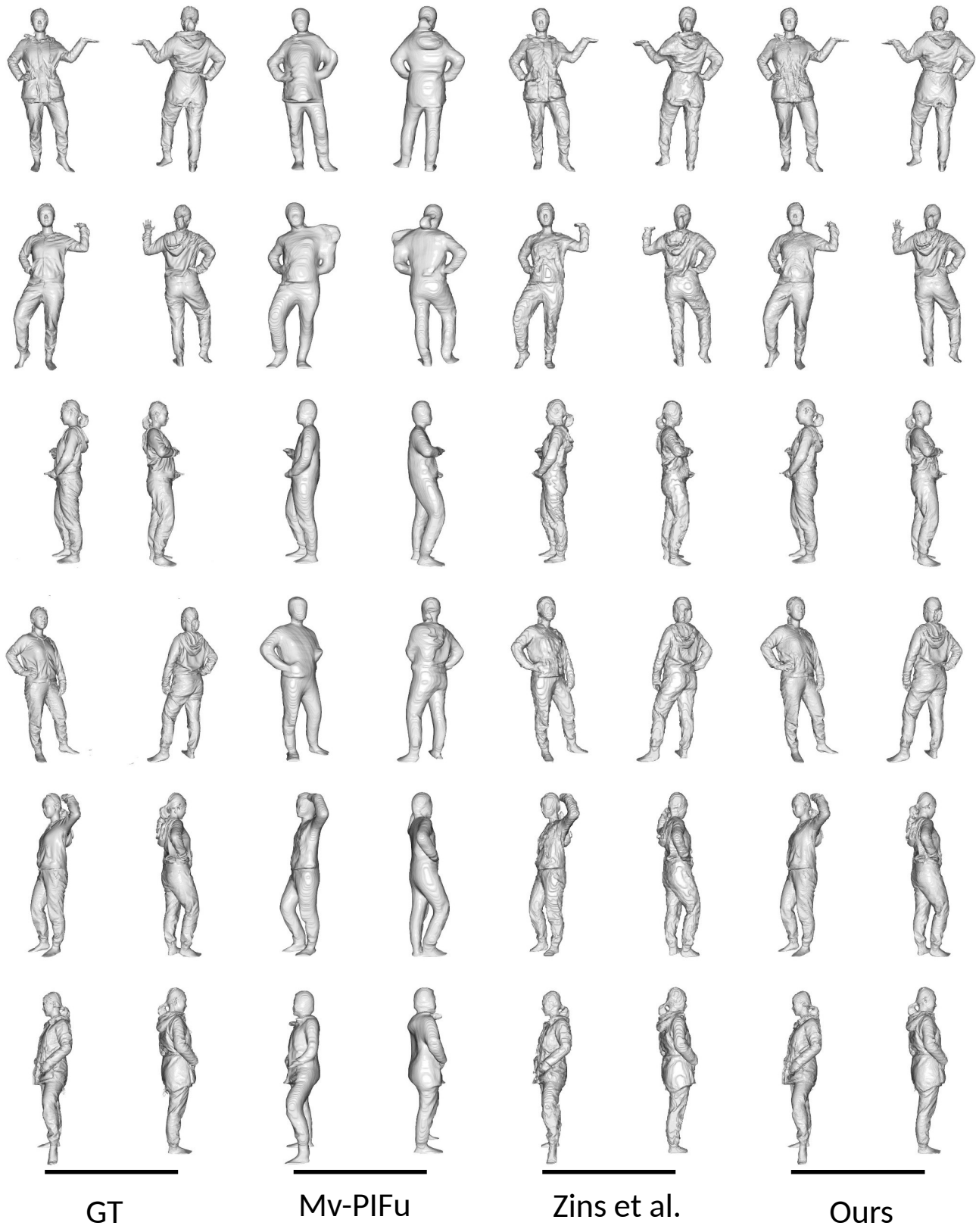

Figure 4. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

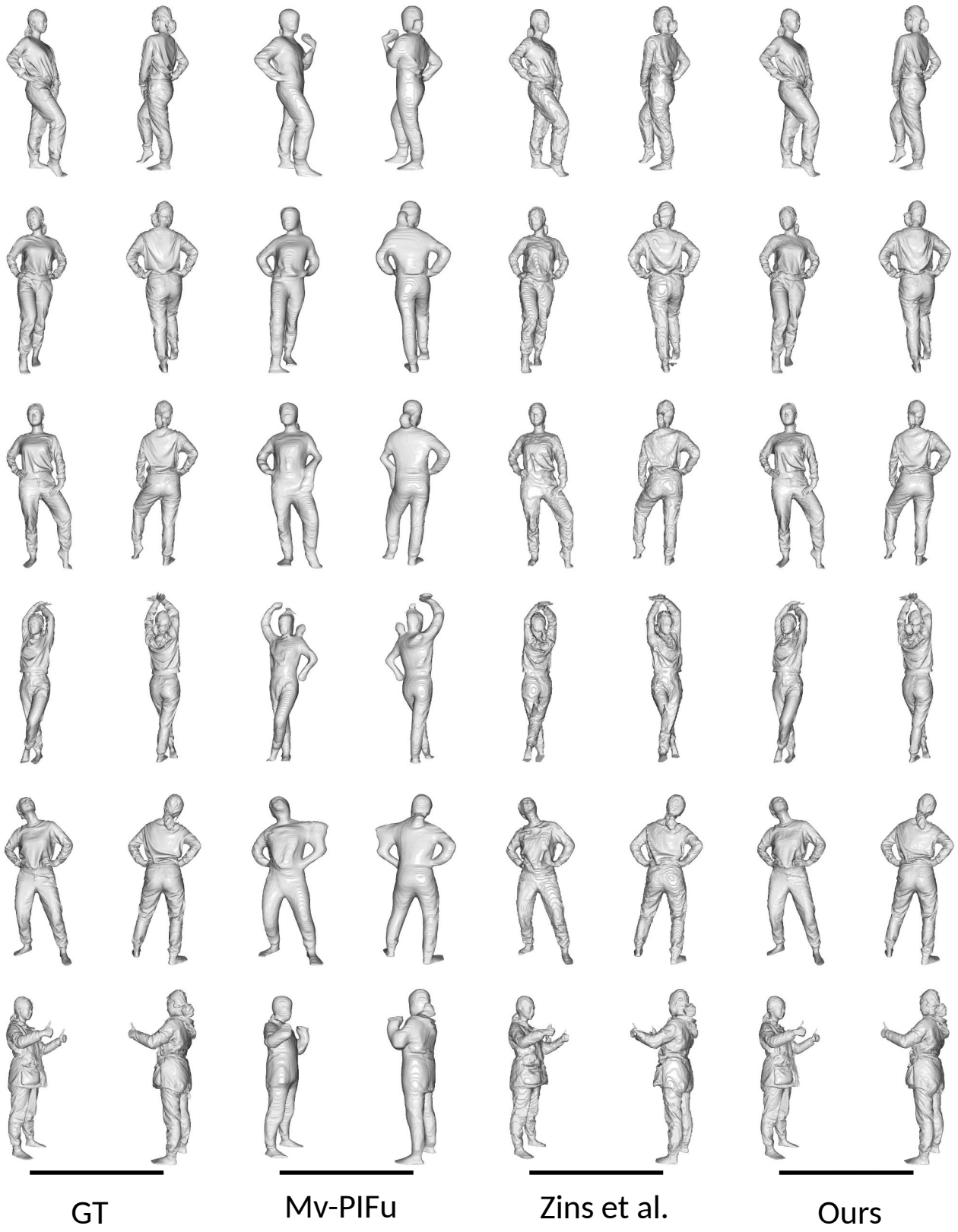

Figure 5. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

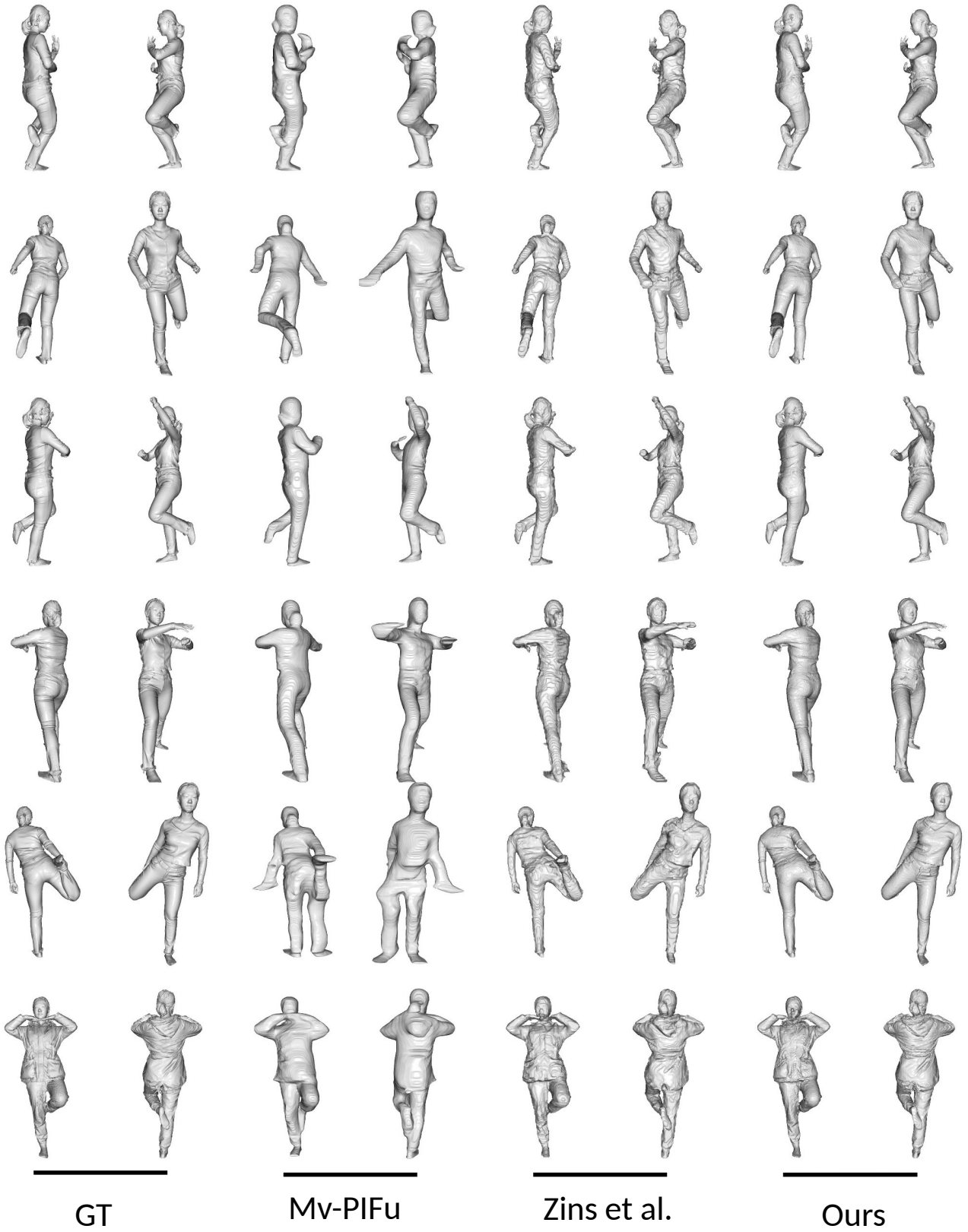

Figure 6. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

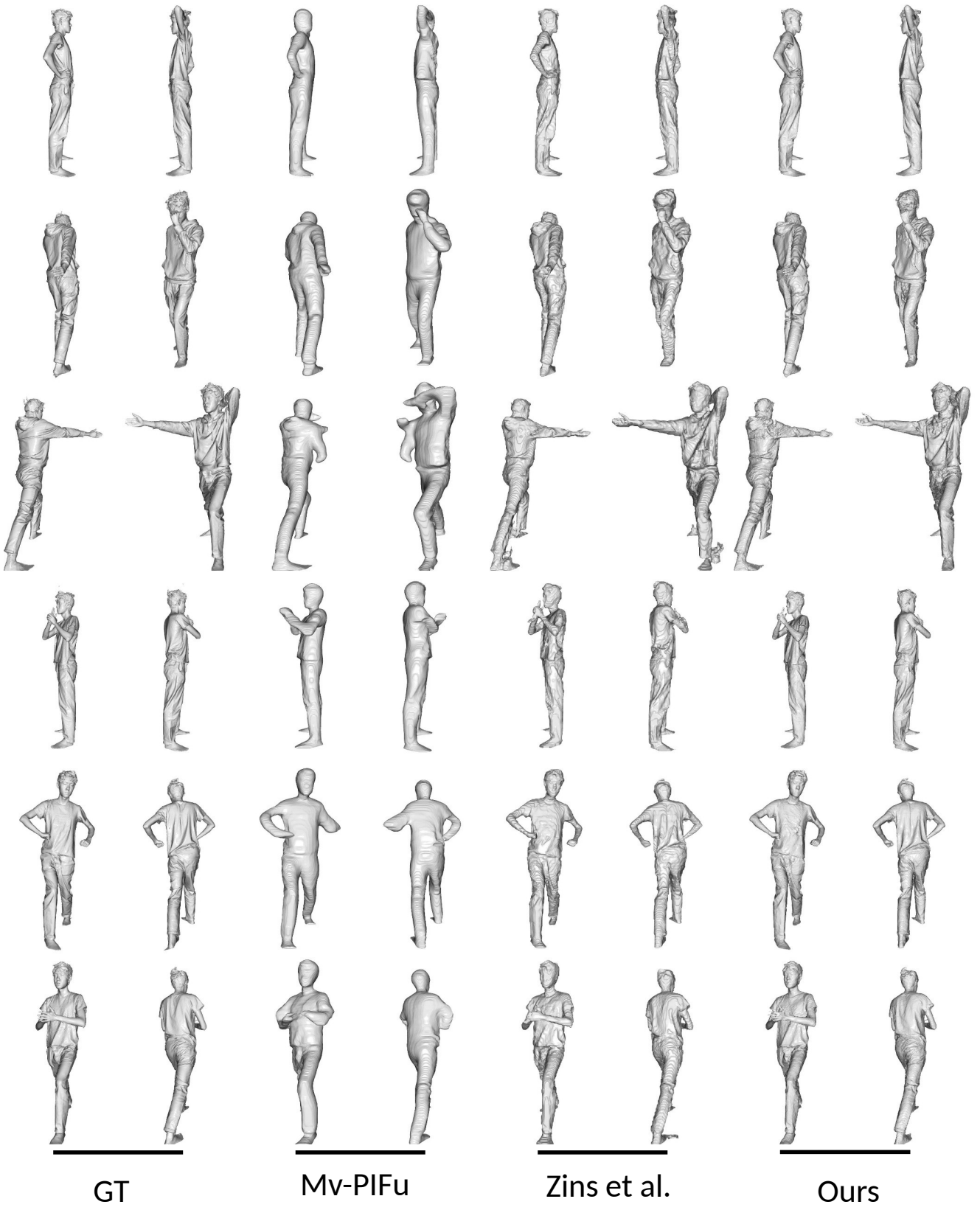

Figure 7. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

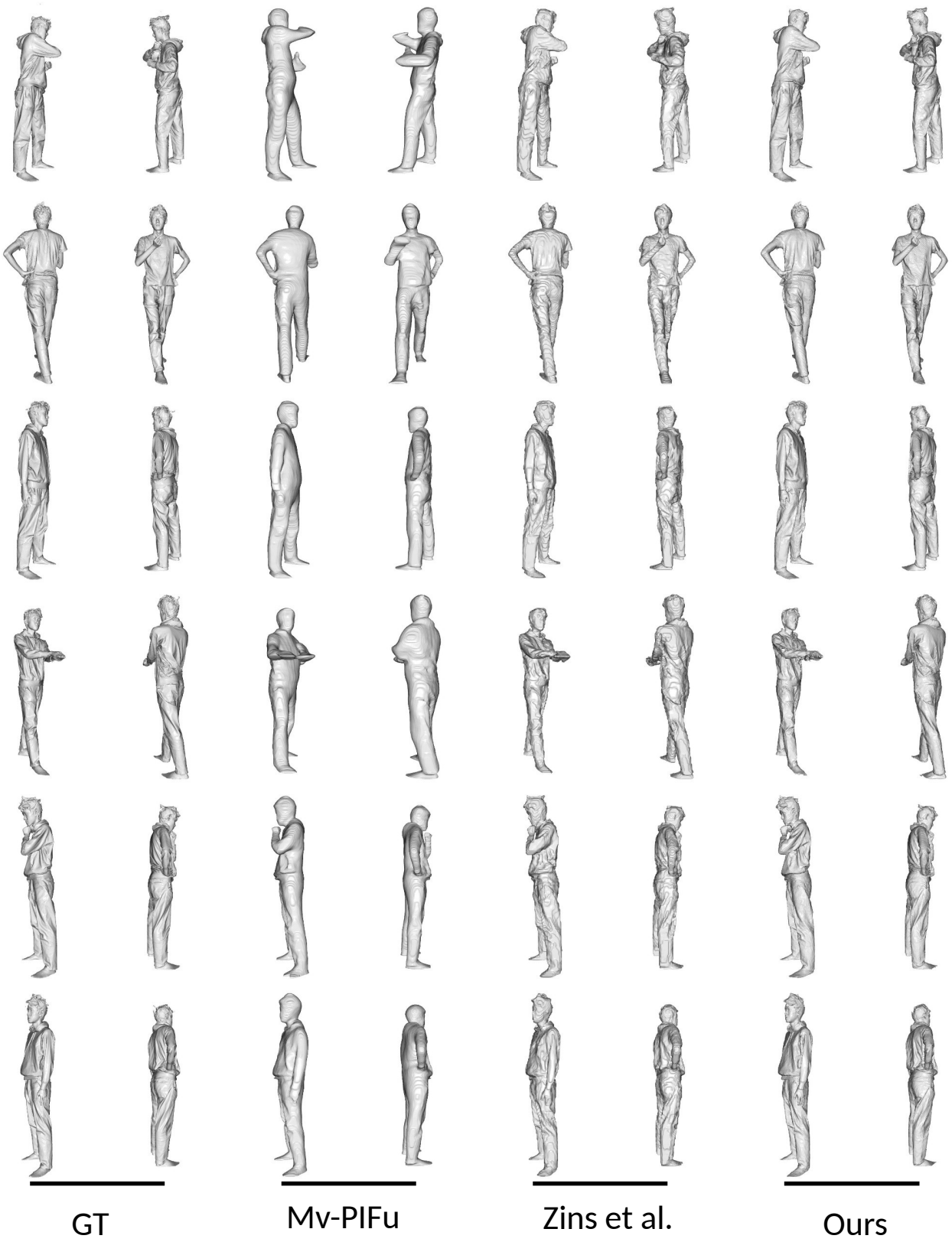

Figure 8. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

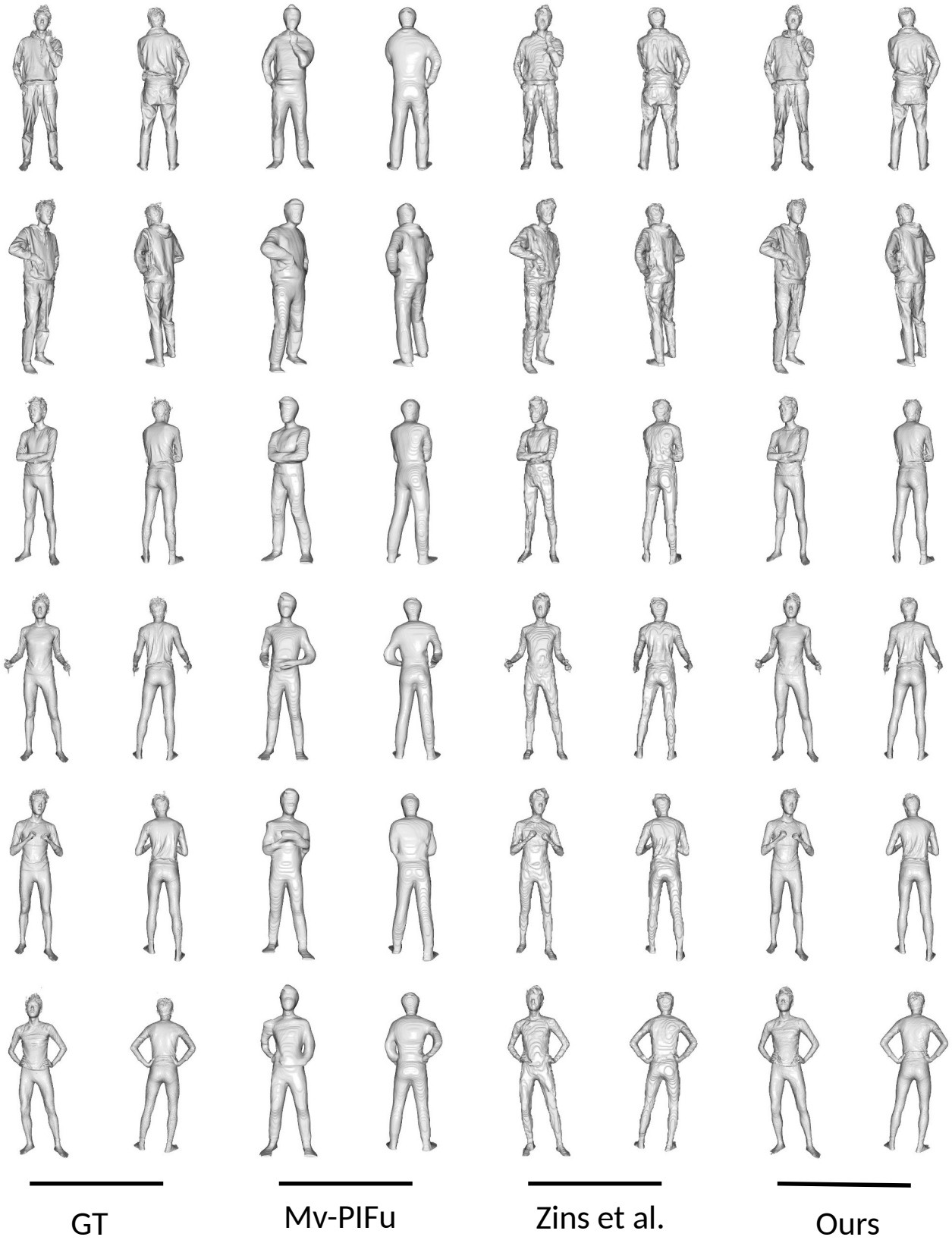

Figure 9. Comparison between our method with 4 camera inputs, MV-PIFu [7], and Zins et al. [10]. From left to right: ground truth, MV-PIFu, Zins et al. and our method.

Figure 10. Demonstration of our method with 4 camera inputs. From left to right, viewpoints are 0°, 90°, 180°, and 270°. Colors indicate surface normals.

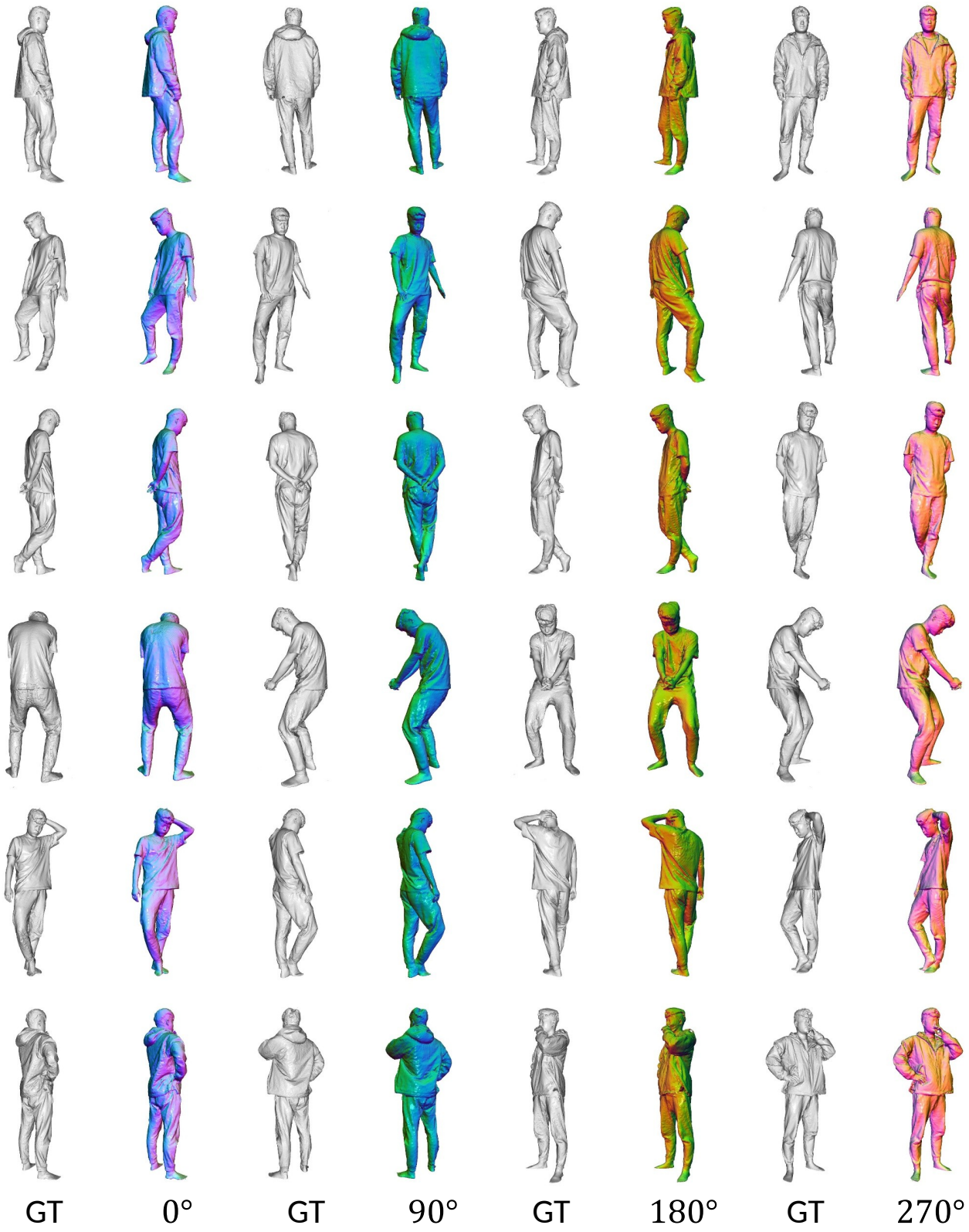

Figure 11. Demonstration of our method with 4 camera inputs. From left to right, viewpoints are 0°, 90°, 180°, and 270°. Colors indicate surface normals.

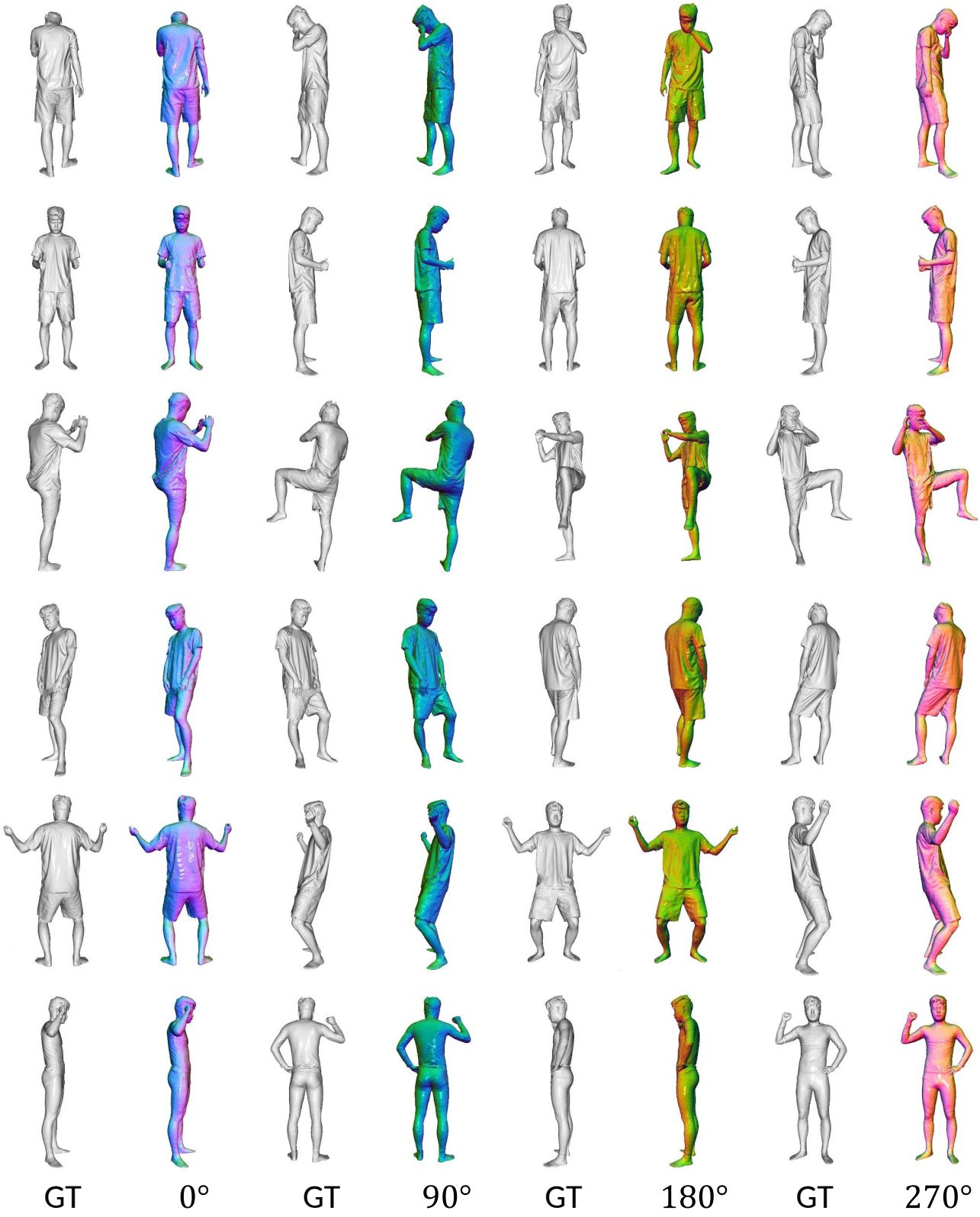

Figure 12. Demonstration of our method with 4 camera inputs. From left to right, viewpoints are 0°, 90°, 180°, and 270°. Colors indicate surface normals.

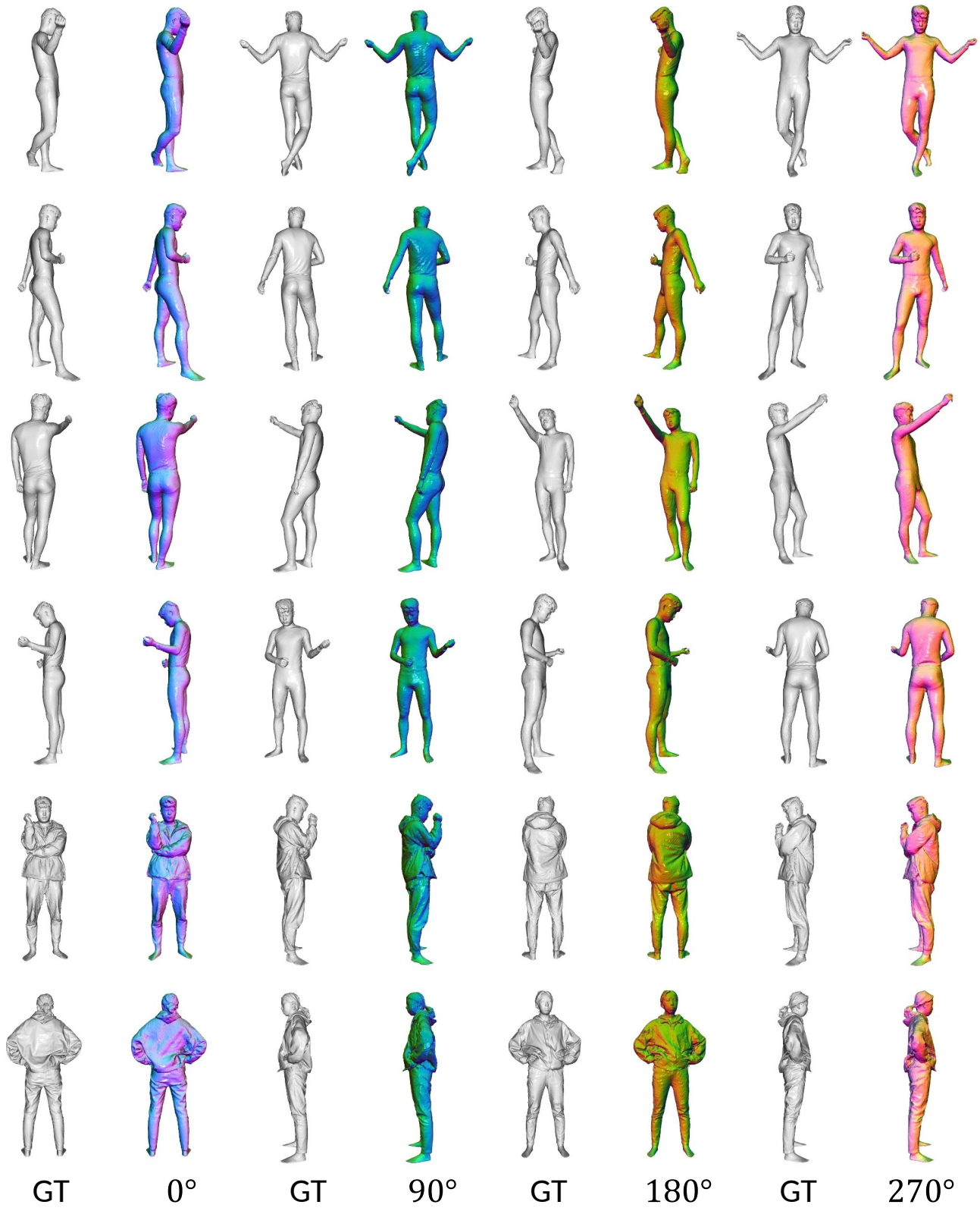

Figure 13. Demonstration of our method with 4 camera inputs. From left to right, viewpoints are 0°, 90°, 180°, and 270°. Colors indicate surface normals.

Figure 14. Demonstration of our method with 4 camera inputs. From left to right, viewpoints are 0°, 90°, 180°, and 270°. Colors indicate surface normals.

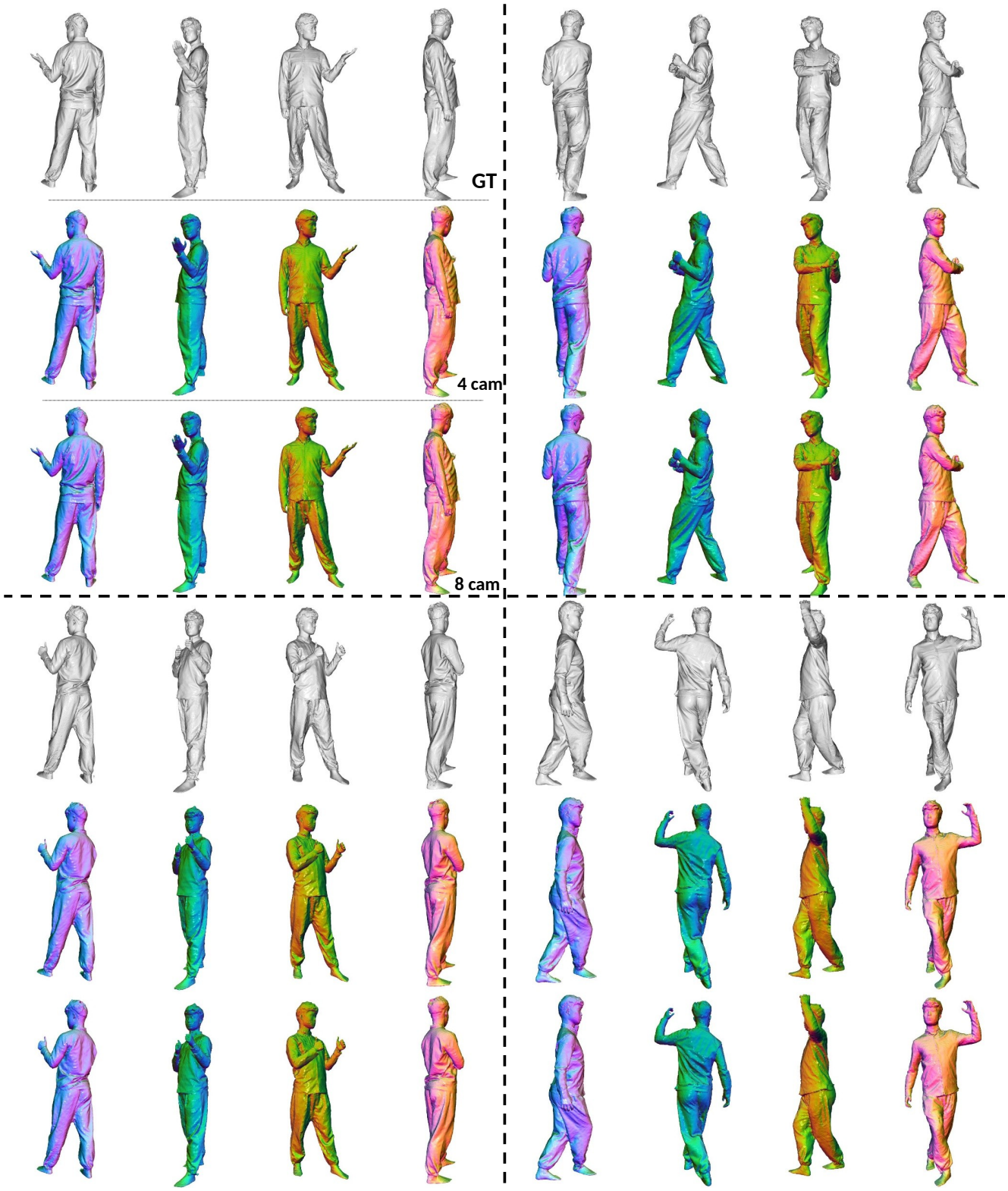

Figure 15. Comparison among reconstructions using four, eight cameras. Each quadrant shows ground truth, reconstruction with four cameras, and reconstruction with eight cameras. Within each row, views are rendered at 0°, 90°, 180°, and 270°.

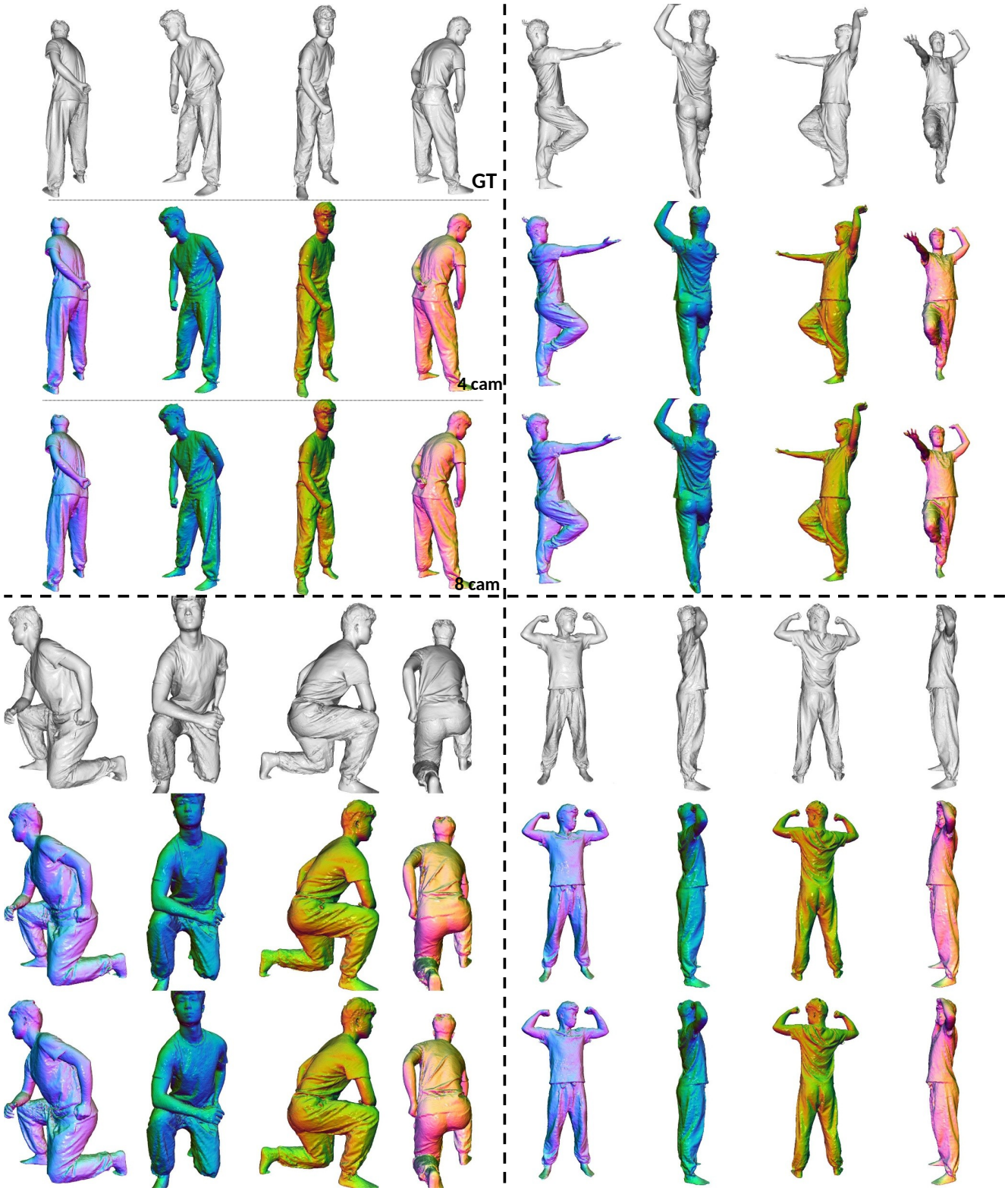

Figure 16. Comparison among reconstructions using four, eight cameras. Each quadrant shows ground truth, reconstruction with four cameras, and reconstruction with eight cameras. Within each row, views are rendered at 0°, 90°, 180°, and 270°.

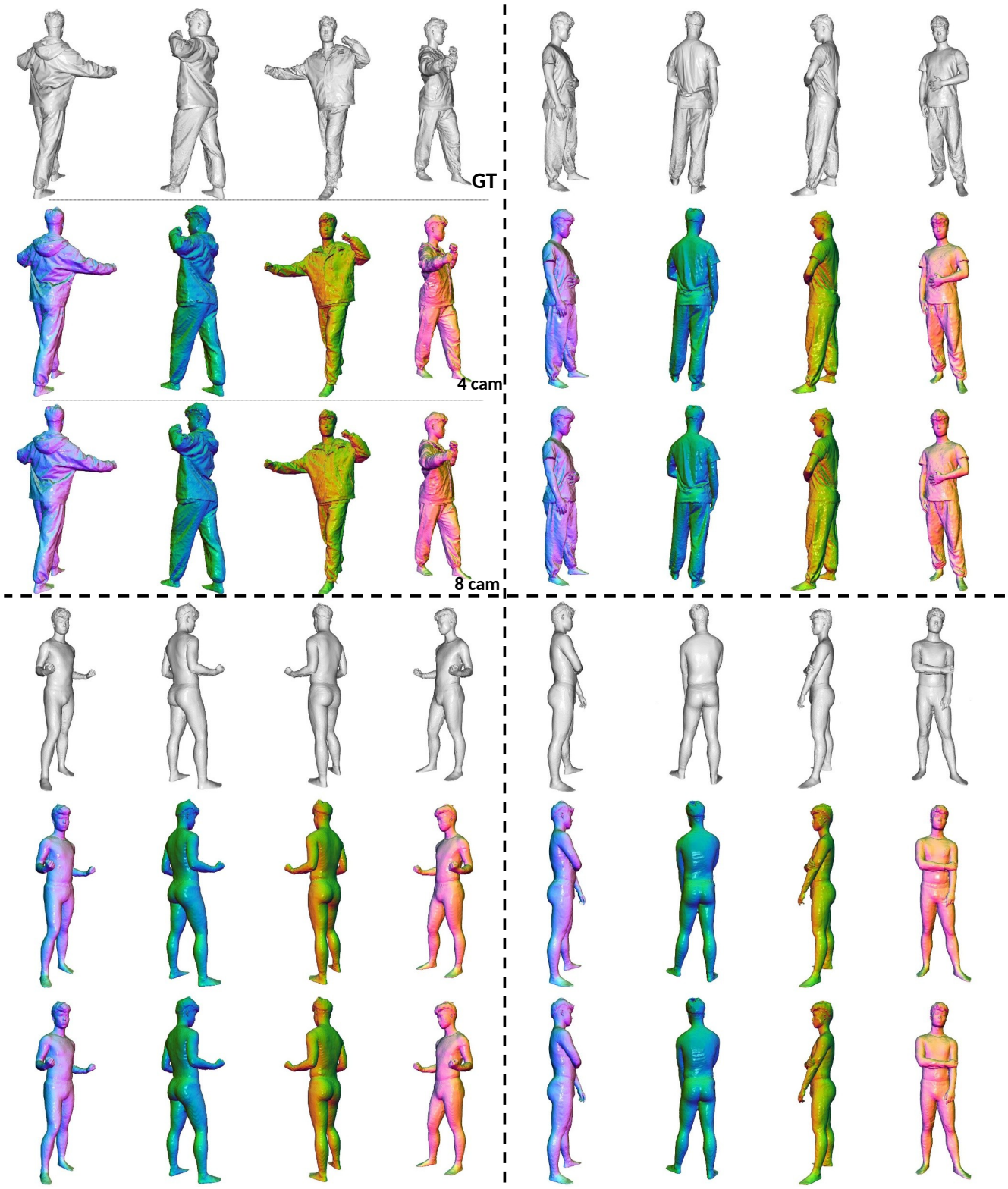

Figure 17. Comparison among reconstructions using four, eight cameras. Each quadrant shows ground truth, reconstruction with four cameras, and reconstruction with eight cameras. Within each row, views are rendered at 0°, 90°, 180°, and 270°.

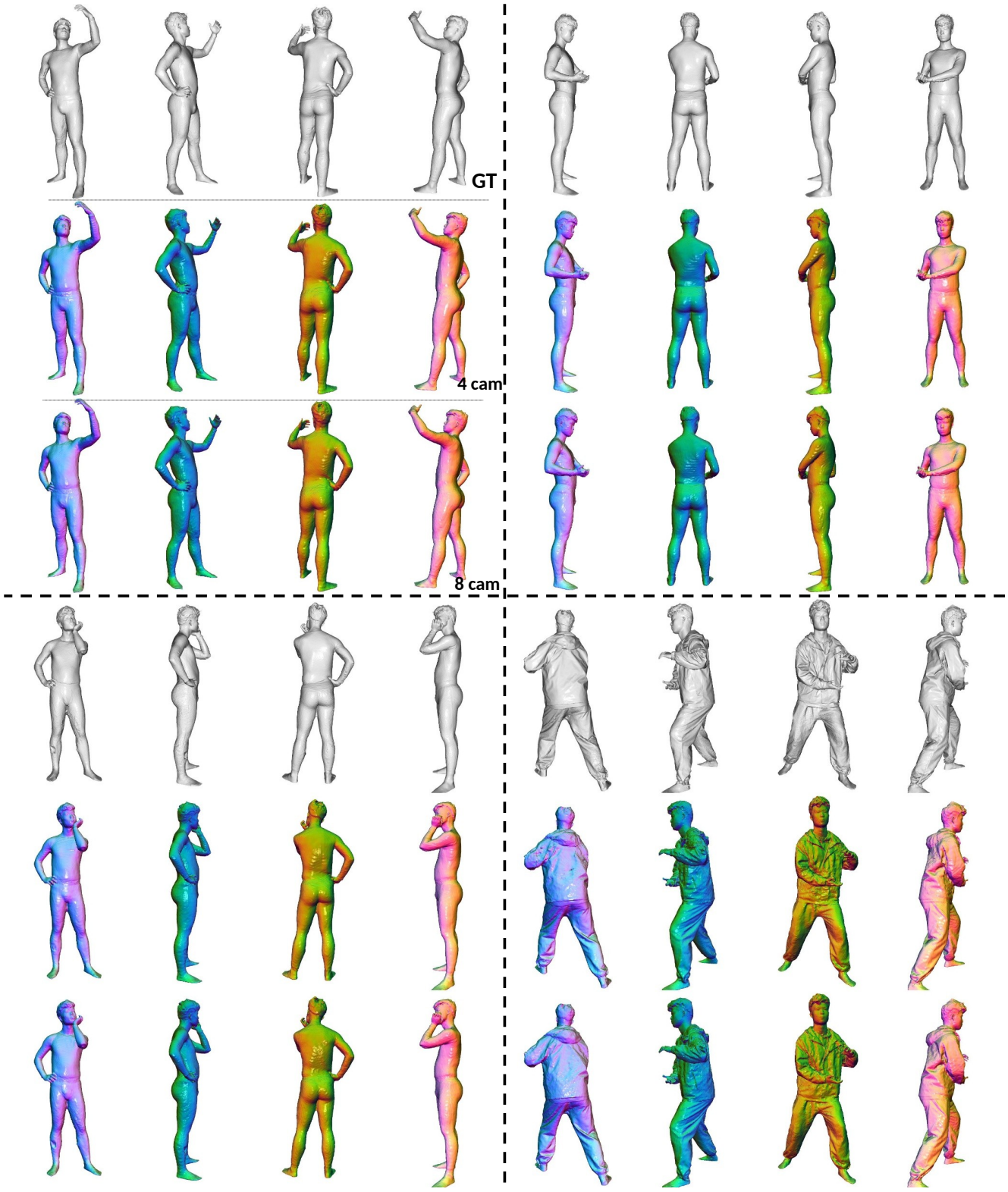

Figure 18. Comparison among reconstructions using four, eight cameras. Each quadrant shows ground truth, reconstruction with four cameras, and reconstruction with eight cameras. Within each row, views are rendered at 0°, 90°, 180°, and 270°.

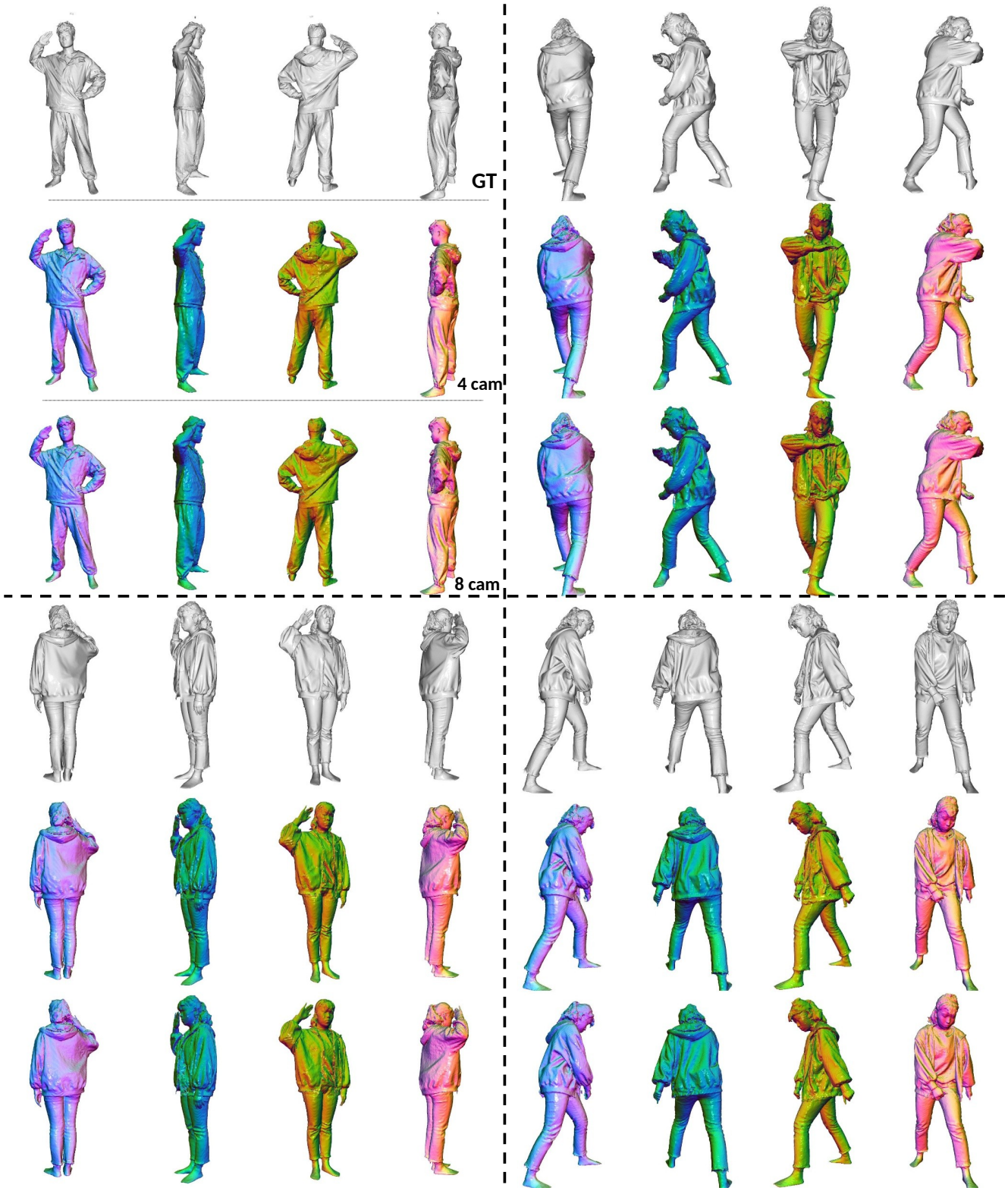

Figure 19. Comparison among reconstructions using four, eight cameras. Each quadrant shows ground truth, reconstruction with four cameras, and reconstruction with eight cameras. Within each row, views are rendered at 0°, 90°, 180°, and 270°.

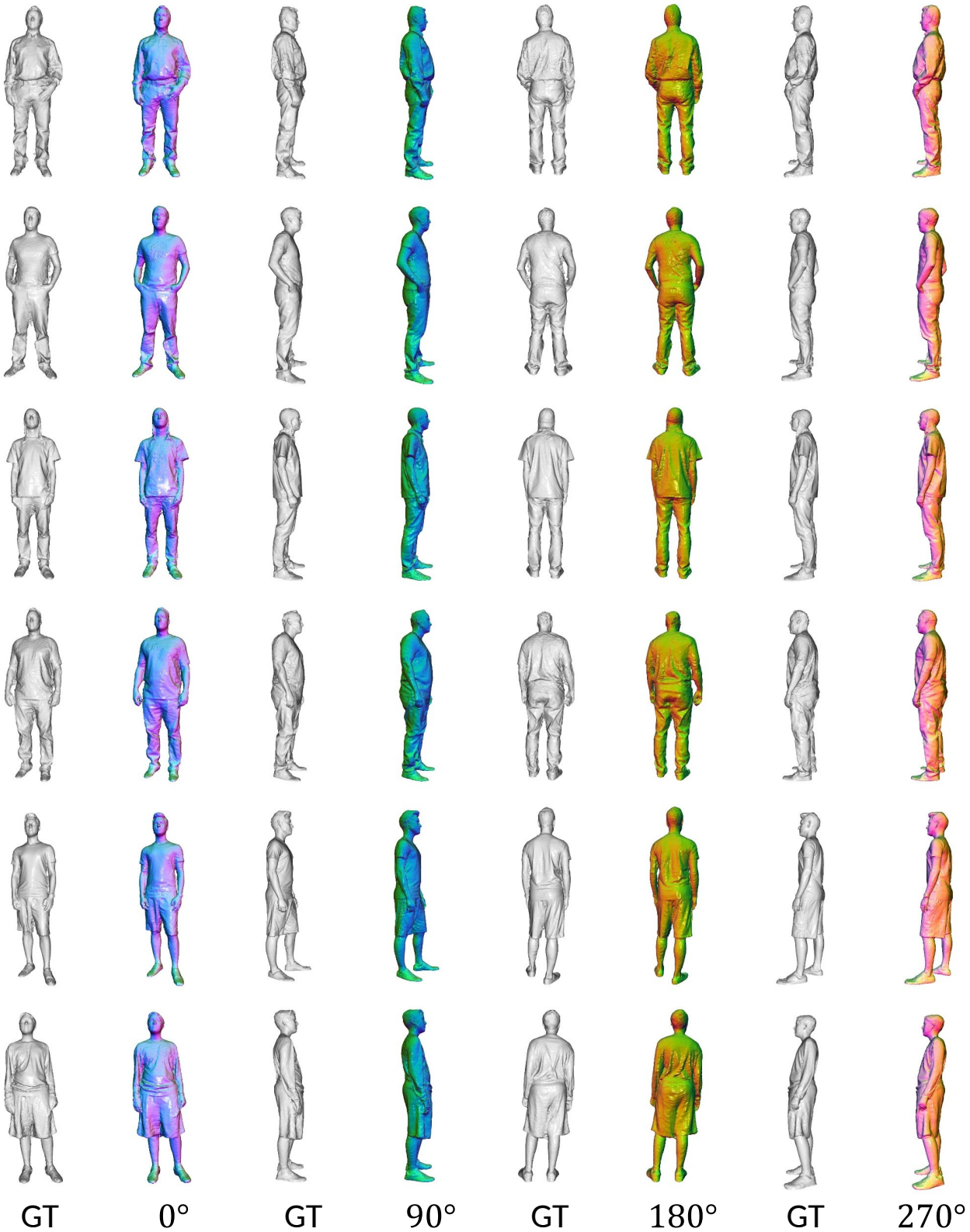

Figure 20. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

Figure 21. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

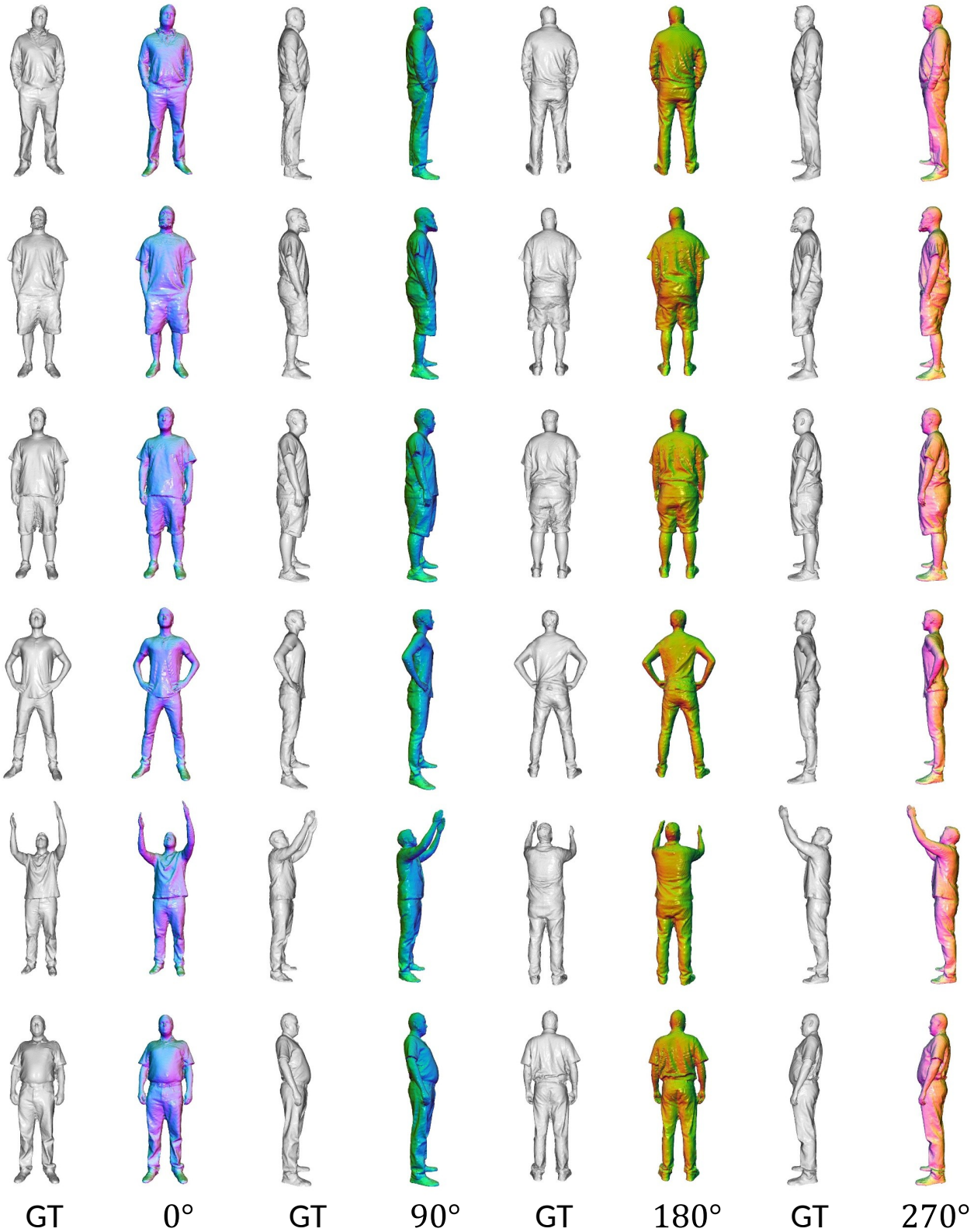

Figure 22. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

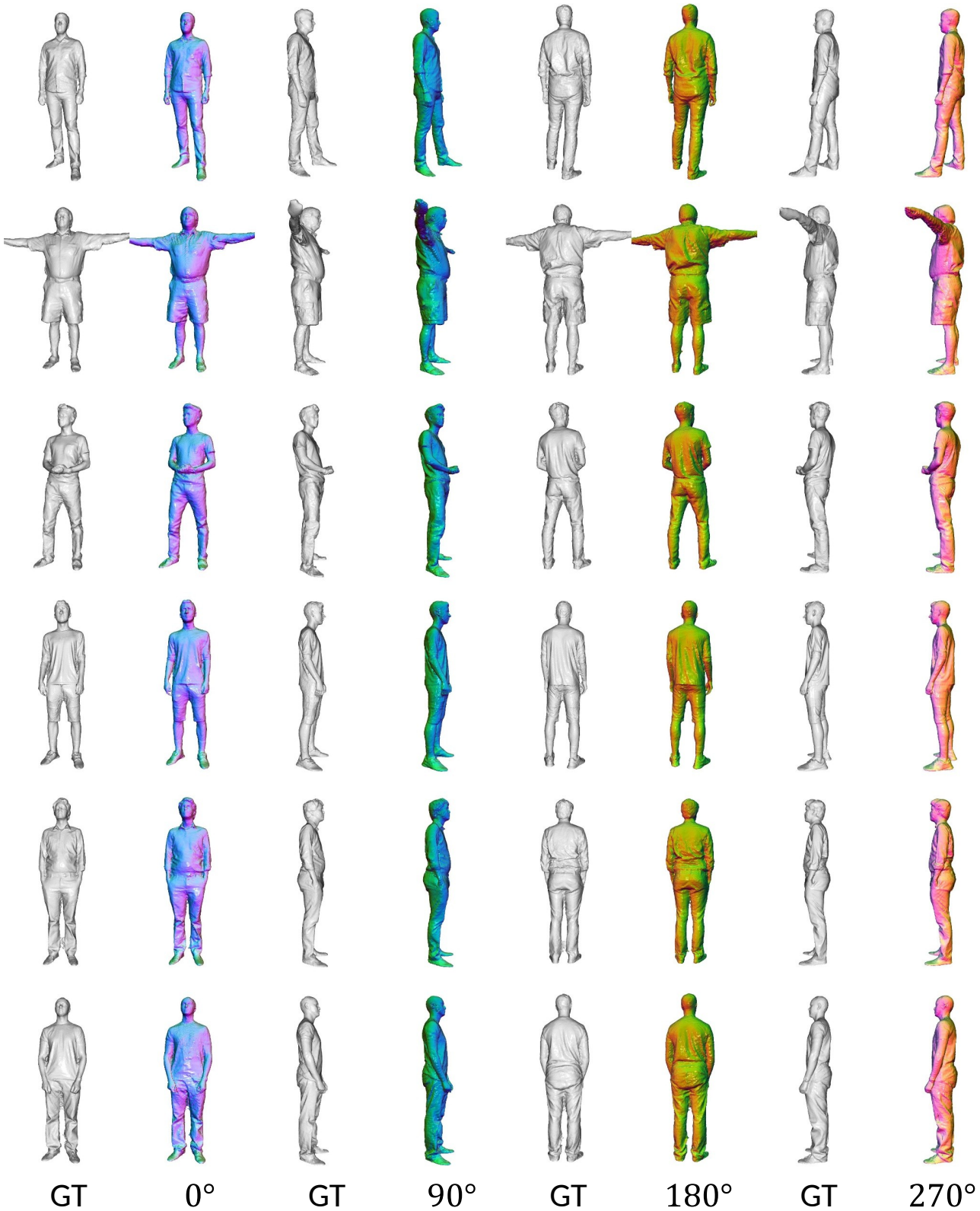

Figure 23. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

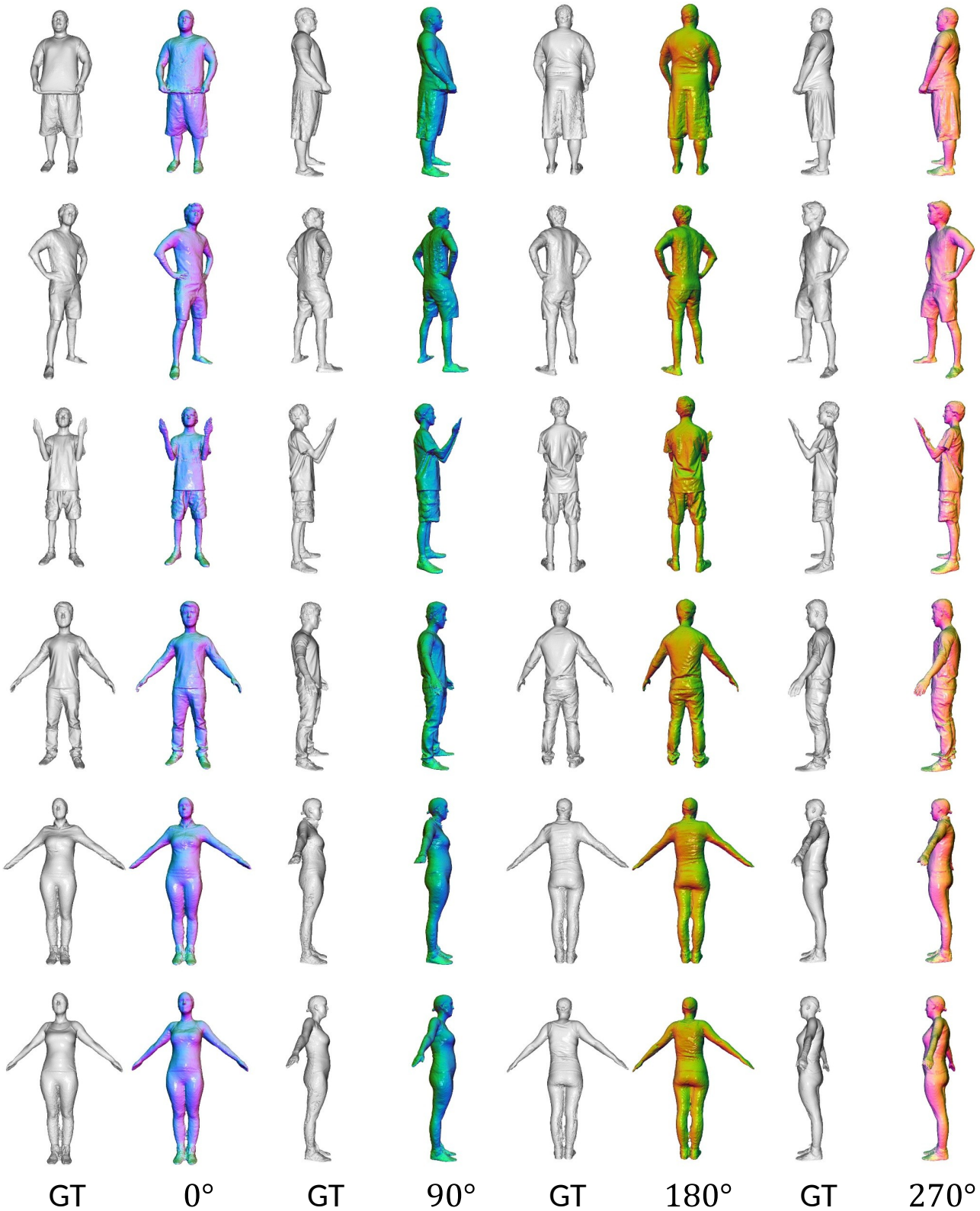

Figure 24. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

Figure 25. Zero-shot results of our pretrained model applied to the MultiGarment [1] dataset.

Figure 26. Zero-shot results of MultiHuman [9] dataset.

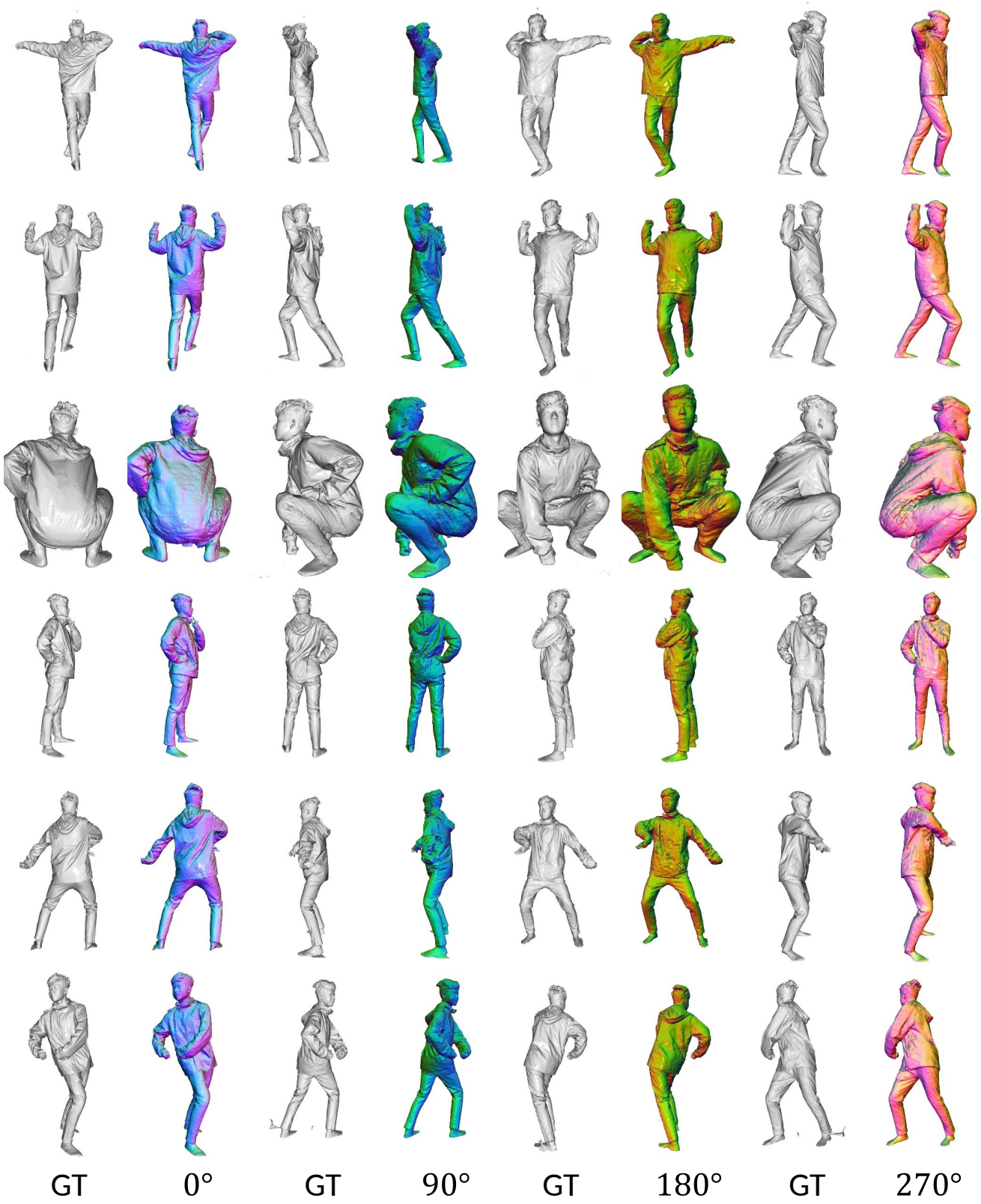

Figure 27. Zero-shot results of MultiHuman [9] dataset.

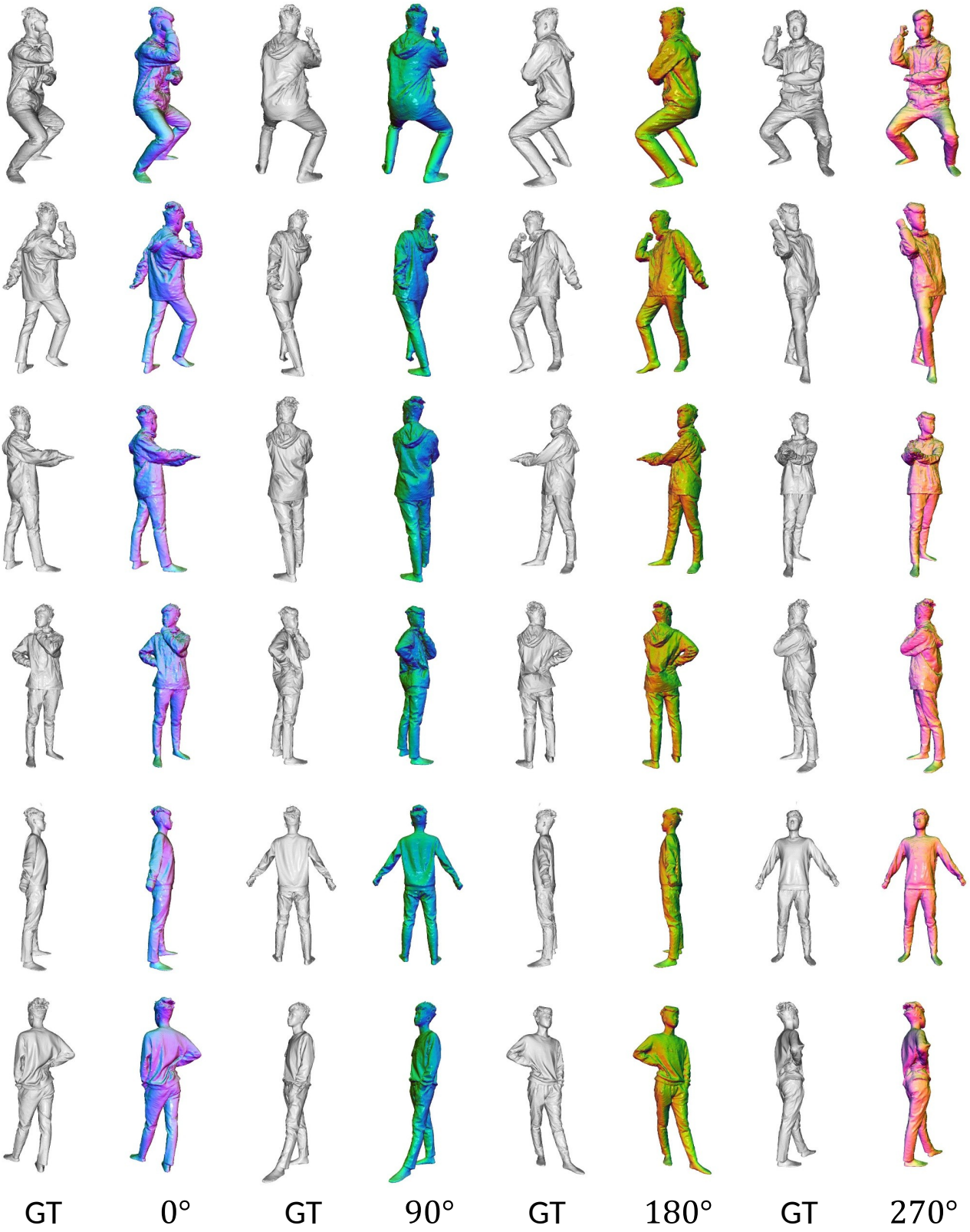

Figure 28. Zero-shot results of MultiHuman [9] dataset.

Figure 29. Zero-shot results of MultiHuman [9] dataset.

Figure 30. Zero-shot results of MultiHuman [9] dataset.

References

[1] 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. 3, 21, 22, 23, 24, 25, 26

[2] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6970–6981, 2020. 1

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 2

[4] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 3

[6] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 1

[7] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2304–2314, 2019. 3, 4, 5, 6, 7, 8, 9, 10

[8] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 3

[9] Yang Zheng, Ruizhi Shao, Yuxiang Zhang, Tao Yu, Zerong Zheng, Qionghai Dai, and Yebin Liu. Deepmulticap: Performance capture of multiple characters using sparse multiview cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6239–6249, 2021. 3, 27, 28, 29, 30, 31

[10] Pierre Zins, Yuanlu Xu, Edmond Boyer, Stefanie Wuhrer, and Tony Tung. Data-driven 3d reconstruction of dressed humans from sparse views. In *2021 International Conference on 3D Vision (3DV)*, pages 494–504. IEEE, 2021. 3, 4, 5, 6, 7, 8, 9, 10