H4D: Human 4D Modeling by Learning Neural Compositional Representation Supplementary Material

Boyan Jiang[1*] Yinda Zhang[2*] Xingkui Wei[1] Xiangyang Xue[1] Yanwei Fu[1]
[1] Fudan University [2] Google

In this supplementary material, we provide implementation details, results about the generalization ability of our motion model, extended comparisons to related works, visualization of principal components, additional qualitative results, run-time comparison, and discussions about limitations, future work and broader impact of our approach.

1. Implementation Details

In this section, we first provide network architectures used for the compositional encoder, Motion-Comp and Shape-Comp networks in our framework. Next, we explain the strategy of choosing the number of principal components for our linear motion model. Finally, we discuss more details in our experiments.

1.1. Network Architecture

Compositional Encoder Both the shape encoder and the initial pose encoder take point cloud of the first time step as input, and we adopted the same architecture as the spatial encoder in Occupancy Flow (OFlow) [19]. The network is a variation of PointNet [21] which has five residual blocks as show in Fig. 1a. Each of the first four blocks has an additional max-pooling operation to obtain the aggregated feature of size $(B, 1, C)$ where C denotes the dimension of hidden layers, and an expansion operation (repeat the pooled feature to the size (B, N, C)) to make it suitable for concatenation. The output of the fifth block is passed through a max-pooling layer and a fully connected layer to get the final outputs of dimension 10 for shape parameter and 72 for initial pose parameter.

Our temporal encoder, for the purpose of learning motion and auxiliary codes, is composed of a point feature extractor (shallow PointNet) and a double layers GRU [4], as shown in Fig. 1b. The shallow PointNet extracts spatial features for each input point cloud, which has 3 hidden layers with hidden sizes equal to 128. We use the same max-pooling and concatenating operations as the spatial encoder. Then the per-frame features are processed sequentially by the GRU layer to provide the latent vector of dimension 90 for motion code and 128 for auxiliary code.

Motion-Comp Network We design a conditional GRU for our Motion-Comp network to learn the compensation of the input motion sequence. Specifically, we use the motion code c_m and auxiliary code c_a as conditions, copy and concatenate them with the pose parameter of each time step estimated by our linear motion model. The detailed architecture is shown in Fig. 1c. The output of the conditional GRU is the motion compensation, and we apply a residual connection to obtain the refined motion sequence, *i.e.* per-frame poses. We can recover body mesh sequences with the predicted shape and per-frame pose codes by using SMPL decoder, here we use the neutral shape model as in previous work [6, 9, 10].

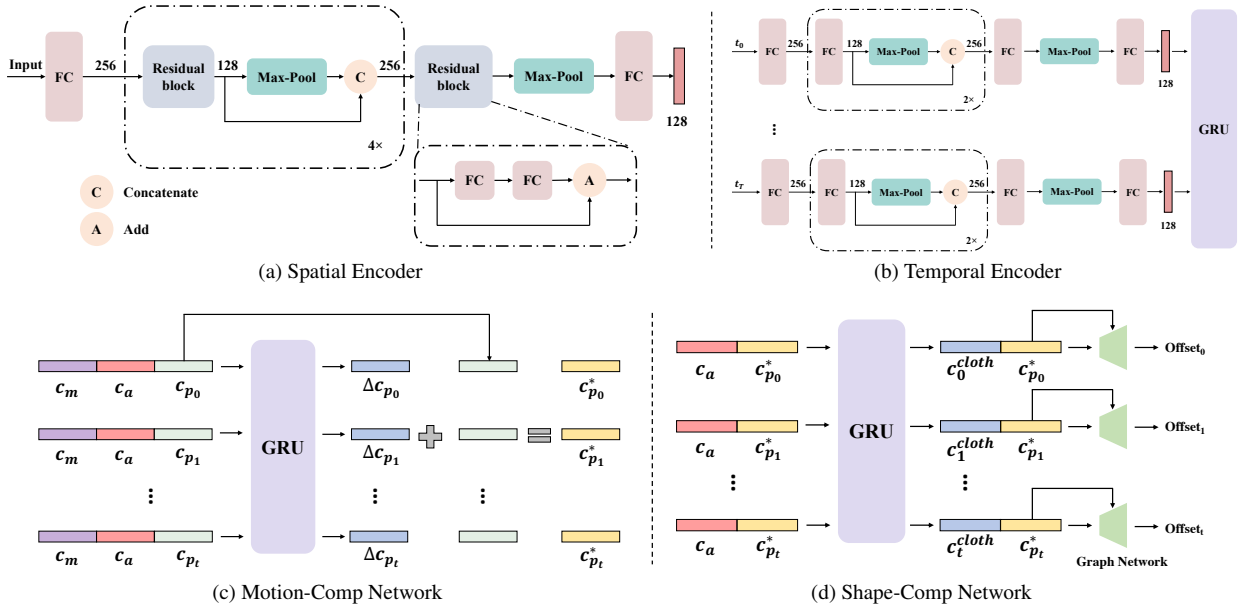
Shape-Comp Network We propose a Shape-Comp network, in which a conditional GRU takes the auxiliary code c_a as input and predicts a new latent vector for each temporal frame conditioned on the predicted pose (we follow CAPE to represent each joint with the flattened rotational matrix and filter the joints that are not related to clothing). The latent vector of each frame is then fed into the graph network to predict per-vertex offsets, which is similar to the CAPE decoder. We remove the one-hot vector of clothing type and only use the predicted pose as condition since we do not focus on the generative task. The architecture is shown in Fig. 1d.

Implementation of GRUs We use the standard API of GRU provided by PyTorch. All the GRUs in our framework share the same architecture, which has 2 layers with the hidden size of 512, except we apply an additional linear layer for each GRU to transform the output dimensions for different modules.

1.2. Linear Motion Model

For the linear motion model (Section 3.2 in the main paper), we employ the Principal Component Analysis (PCA)

* indicates equal contributions.
Boyan Jiang, Xingkui Wei and Xiangyang Xue are with the School of Computer Science, Fudan University.
Yanwei Fu is with the School of Data Science, Fudan University, and Fudan ISTBI—ZJNU Algorithm Centre for Brain-inspired Intelligence, Zhejiang Normal University, Jinhua, China.



(a) Spatial Encoder (b) Temporal Encoder

(c) Motion-Comp Network (d) Shape-Comp Network

Figure 1. **Detailed network architectures in our framework.**

to model the per-frame difference of the pose parameter regarding the first frame in a sequence. As stated in Sec. 3.2 of the main paper, we run PCA separately for the global orientation (*i.e.* pelvis) and the remaining body joint rotations. Inspired by Urtasun *et al*. [24], we choose the number of PCA components depending on the fraction of the total variance of the training data that is captured by the subspace, denoted by $Q(m)$:

$$Q(m) = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{M} \lambda_i} \tag{1}$$

where m controls the number of principal components, λ_i are ordered eigenvalues of the data covariance matrix such that $\lambda_i \geq \lambda_{i+1}$, and M is the total number of eigenvalues. In our experiments, we choose $m = 4$ for the global rotation and $m = 86$ for the remaining body joints rotation, which satisfy $Q(m) > 0.9$. We visualize some principal components in Fig. 5 and 6 (Sec. 4).

1.3. Experiment Details

Loss Functions Given an input point cloud sequence, our model generates one shape code c_s and three mesh sequences $\mathbf{X}_{\text{linear}}$, $\mathbf{X}_{\text{motion}}$ and $\mathbf{X}_{\text{shape}}$, which correspond to the outputs of LMM, Motion-Comp network and Shape-Comp network respectively. Each sequence has $L = 30$ mesh frames and each mesh has $K = 6890$ vertices. We also have the ground truth shape parameter c_s^* and posed SMPL body mesh sequence $\mathbf{Y}_{\text{body}}$. Furthermore, we compute ground truth offsets sequence by $\mathbf{Y}_{\text{offset}} = \mathcal{M}_{\text{clothed}} - \mathcal{M}_{\text{SMPL}}$, where $\mathcal{M}_{\text{clothed}}$ and $\mathcal{M}_{\text{SMPL}}$ stand for the vertices of the clothed human mesh and corresponding SMPL body mesh in the canonical pose respectively.

Then we define the reconstruction loss as the per-vertex L_1 error with the ground truth mesh

$$\mathcal{L}_{\text{r}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{LK} \sum_{l=1}^{L} \sum_{k=1}^{K} \left\| \mathbf{X}_{l,k} - \mathbf{Y}_{l,k} \right\|_1 . \tag{2}$$

Furthermore, we apply L_2 penalization on the predicted shape code to further alleviate the ambiguity between body shape and clothing, given by

$$\mathcal{L}_{\text{s}}(\mathbf{c}, \mathbf{c}^*) = \left\| \mathbf{c} - \mathbf{c}^* \right\|_2^2 . \tag{3}$$

Finally, the total loss for training can be formulated as

$$\begin{aligned} \mathcal{L} = \lambda_s \mathcal{L}_{\text{s}}(\mathbf{c_s}, \mathbf{c_s^*}) &+ \lambda_{r_1} \mathcal{L}_{\text{r}}(\mathbf{X}_{\text{linear}}, \mathbf{Y}_{\text{body}}) \\ &+ \lambda_{r_2} \mathcal{L}_{\text{r}}(\mathbf{X}_{\text{motion}}, \mathbf{Y}_{\text{body}}) \\ &+ \lambda_{r_3} \mathcal{L}_{\text{r}}(\mathbf{X}_{\text{shape}}, \mathbf{Y}_{\text{offset}}) , \end{aligned} \tag{4}$$

we set $\lambda_s = \lambda_{r_1} = 1$, $\lambda_{r_2} = \lambda_{r_3} = 0$ for the first training stage and $\lambda_s = \lambda_{r_2} = 1$, $\lambda_{r_3} = 30$ and $\lambda_{r_1} = 0$ for the second stage.

Backward Experiments For our auto-decoding based experiments, *i.e.* completion and prediction, we use the trained model to perform a backward fitting algorithm. Specifically, we remove the encoder, freeze the parameters of the remaining modules and optimize the SMPL parameters and latent codes with back-propagation to produce the outputs as similar to the observations as possible.

We initialize the SMPL parameters and latent codes with the random vector sampled from a Gaussian distribution $N(0, 0.01)$ and use the Adam optimizer [8] with learning rate $3e^{-2}$ to perform back-propagation for 500 iterations. In each iteration, we uniformly sample 8192 points

on the surface of the predicted meshes, and compute Chamfer loss [17,22] w.r.t the observed points for penalizing. Additionally, we follow IPNet [2] to add pose and shape prior terms, which penalize unnatural output bodies during optimization.

Completion We conduct two different types of motion completion experiments, *i.e*. temporal completion and spatial completion. Given a temporal sequence of $L = 30$ frames, for temporal completion, we randomly select 15 frames as observation and optimize the SMPL parameters and latent codes to complete the missing frames. We choose HMMR [7] and 4D-CR-SMPL (an extension we implement for 4D-CR [5]) as baselines. To implement 4D-CR-SMPL, we replace their implicit decoder with the SMPL decoder, and set the dimensions of their identity code and initial pose code to 10 and 72 respectively. Then we can obtain the pose code for each time step with the Neural ODE conditioned on the motion code, and input it to the SMPL decoder with the identity code to produce the reconstructed mesh frame. For 4D-CR-SMPL, we use the model trained on our dataset, and for HMMR, we use the official pretrained model.

The goal of spatial completion is to complete the temporal sequence with partial spatial observation. To this end, we use the raw scanned mesh sequences of CAPE [15], and render the depth images of resolution 512×512 with the approach illustrated in the main paper (Sec. 4.2) to simulate the real world scenario. We assume the camera poses are known and back project the depth images to obtain partial point clouds. We initialize our codes with the random vectors sampled from a Gaussian distribution $N(0, 0.01)$ with no requirement for an additional initialization step like NPMs [20], and use the Adam optimizer with learning rate $3e^{-2}$ to perform back-propagation for 500 iterations. Note that in this experiment, we adopt one-directional point-to-surface loss instead of two-directional Chamfer loss due to the partial geometry. We show some qualitative examples in Fig. 12.

2. Generalization of Our Motion Model

In this section, our goal is to investigate the capacity of our motion model for representing novel motions from another dataset. To this end, we choose some motion sequences from AMASS [16], a large 3D MoCap dataset. And then we use our model trained on the CAPE dataset [15], perform the similar backward algorithm in completion and prediction experiments to fit the whole sequence of $L = 30$. Instead of dense SMPL, we randomly sample 8192 points from the SMPL mesh of each frame as observations, then use the Chamfer loss to the points sampled from the predicted mesh. Prior terms [2] are also used to penalize the unnatural output. Since AMASS only provides SMPL parameters, we disable the Shape-Comp network and use the results from Motion-Comp network for visualization. Fig. 2 shows that the proposed method successfully reconstructs the full sequence from such sparse input, which demonstrates the generalization capability of our model to represent novel motions from another data source.

3. Extended Comparisons to Related Works

Comparison to HuMoR We compare with a SoTA human body estimation method HuMoR [23] on the task of fitting point cloud sequences. Specifically, we choose 100 mesh sequences of 30 frames from our test set and randomly sample 8192 points from each frame. Then we use the pretrained model of both methods to conduct backward optimization with 2 choices of loss functions, *i.e*. Chamfer loss (HuMoR, Ours) or 3D keypoint loss (HuMoR*, Ours*). For both losses, we also enabled prior losses to regularize the predicted shape and motion for H4D. As shown in Tab. 1, our method beats HuMoR in both cases. The qualitative comparisons are shown in Fig. 3, HuMoR can only recover the global movement trend without accurate limbs by using Chamfer loss and gains a significant improvement when using 3D keypoints as supervision. In contrast, our method obtains more accurate results in both cases.

	PA-MPJPE ↓	MPJPE ↓	PVE ↓	Accel ↓
HuMoR	70.7	46.0	45.4	10.5
Ours	32.6	30.0	27.6	4.9
HuMoR*	25.7	27.1	26.1	7.3
Ours*	**16.2**	**14.5**	**11.4**	**4.5**

Table 1. **Quantitative comparisons to HuMoR.** By default, the Chamfer loss between the input point cloud and points sampled from the reconstructed mesh is adopted. And * indicates that we use 3D keypoint as supervision.

Comparison to NPMs We provide the comparisons to NPMs [20] on depth completion task. We choose 100 sequences and use the pretrained model of NPMs to perform completion from partial depth. Specifically, given a depth image sequence of 30 frames, we project the depth values into a 256^3-SDF grid to generate the inputs for NPMs, and then optimize the latent codes frame-by-frame with the default setup. Note that NPMs runs 10 times slower than H4D and uses twice of the GPU memory. Tab. 2 shows that our model outperforms NPMs, either w/ or w/o (NPMs*) encoders for code initialization, on both metrics. As can be seen from the qualitative results shown in Fig. 4, NPMs produces accurate motion but fails to recover fine-grained geometry, while our results are plausible on both shape and motion.

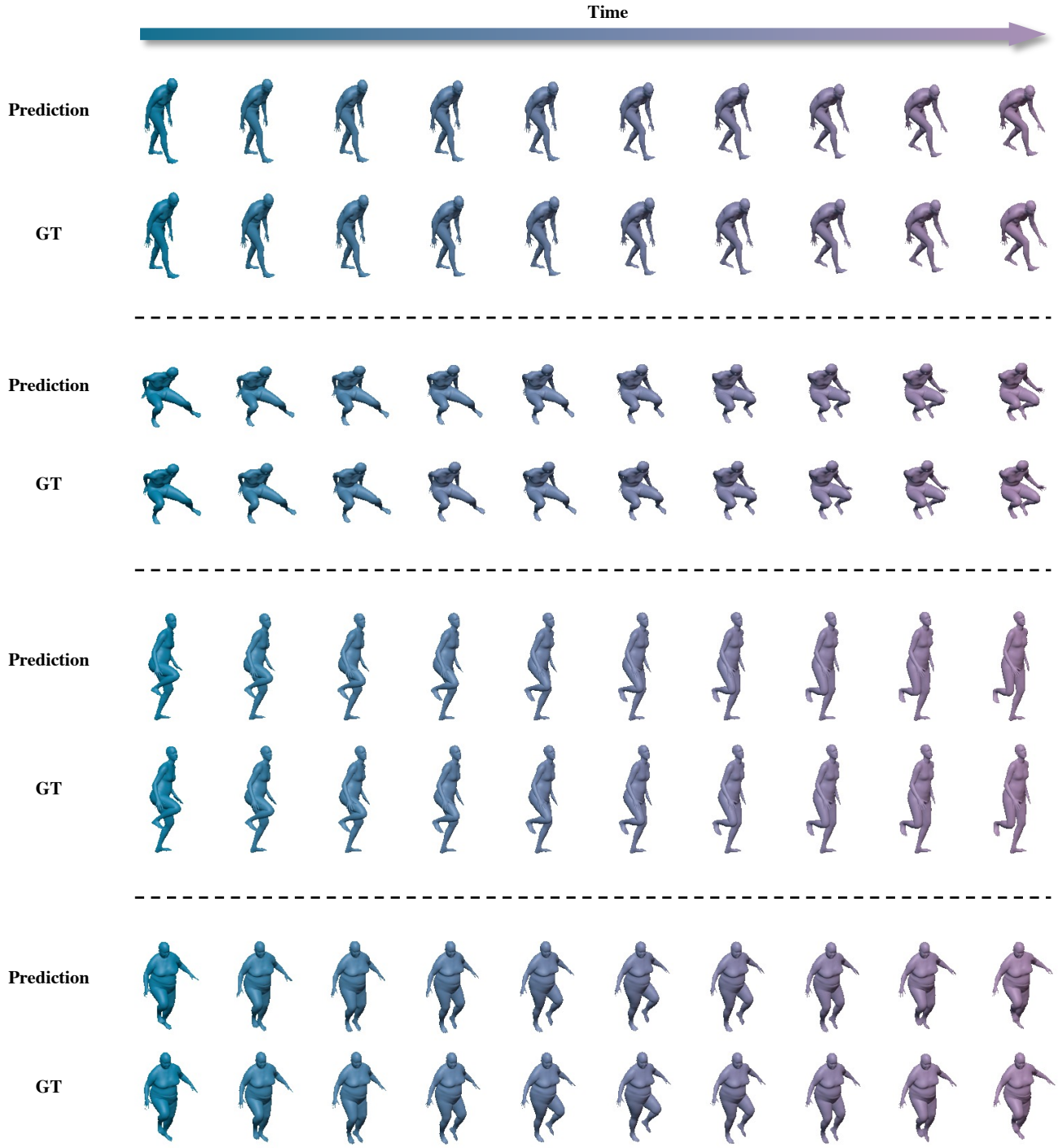


Figure 2. **Results of the novel motions from AMASS dataset.** To investigate the generalization ability of our method, we choose 4 motion sequences from the AMASS dataset, and use our model trained on the CAPE dataset to fit them by using the backward algorithm.

4. Visualization of Principal Components

The linear motion model in our framework totally has 90 principal components, the first 4 components are for global rotation (pelvis joint) and the rest 86 for other body joints. We start from the same rest pose and visualize some principal components in Fig. 5 and 6. Specifically, for each shown component, we select different scaling factors (be-

	IoU ↑	CD ↓
NPMs*	79.3%	0.104
NPMs	85.5%	0.042
Ours	**87.7%**	**0.037**

Table 2. **Quantitative comparisons to NPMs.** We use the random vectors sampled from a Gaussian distribution $N(0, 0.01)$ to initialize the codes for NPMs* and Ours, and use the pretrained encoders to obtain initialization for NPMs.

fore each row) and multiply them with this component to show the motion results. As shown, PC0 roughly controls the global rotation around the vertical axis; PC4 and PC6 affect the opening and closing of the upper arm and forearm, respectively; PC7 is related to the bending of the legs; and PC9 tells the movement of arms and legs at the same time. In general, the positive and negative scaling factors of components correspond to opposite directions of motion, and the absolute value affects the magnitude of the motion.

5. Additional Qualitative Results

We show additional qualitative examples on 4D reconstruction in Fig. 7, shape and motion recovery in Fig. 9, temporal completion in Fig. 10 and 11, spatial completion in Fig. 12, future prediction in Fig. 8, motion retargeting in Fig. 13 and ablation study in Fig. 14.

6. Run-time

In Tab. 3, we show the per sequence run-time of our method and previous 4D representation methods on forward inference for 4D reconstruction and backward optimization for temporal completion. Note that we report the time cost to run a complete backward optimization process for a sequence (500 iterations). The length of the full sequence is $L = 30$, and all models run on a single NVIDIA 2080Ti GPU. Instead of the Neural ODE [3], we model the human motion using the linear model and GRU-based compensation networks. As can be seen, our model runs faster in both cases, especially in backward optimization.

	Forward (s)	Backward (min)
OFlow [19]	1.106 (0.814)	17.600
4D-CR [5]	14.469 (5.861)	14.117
4D-CR-SMPL [5]	0.209	16.817
Ours	**0.175**	**7.303**

Table 3. **Comparisons about the run-time.** We show per sequence run-time of our method and baselines on forward inference and backward optimization. The numbers in the parentheses mean time without Marching Cubes.

7. Limitations and Future work

We now discuss a few limitations of our approach that point to future work. **First**, our motion model can reconstruct the discretized frame w.r.t each input time step, but not the arbitrary time in the continuous whole time span like 4D-CR [5] or OFlow [19], which will be useful in some scenarios requesting higher temporal resolution from inputs. Incorporating a network that takes a time value scalar as input, *e.g.* Neural ODE [3], temporal MLP, would be a solution. **Second**, we currently conduct all experiments on the sequences of 3D data, *e.g.* point clouds or meshes. On the one hand, this is due to the lack of 4D human datasets with color images, *e.g.* pairs of video and 3D human sequences (with clothing and hair). And on the other hand, the focus of this work is to propose a compositional representation and effectively power various 4D human-related applications based on point cloud. Combining our representation with techniques such as neural rendering [13, 18] or photometric-based optimization [12, 14] for image-based full human 4D reconstruction would be a promising future direction. **Third**, we adopt the same clothing representation used in previous work [1, 11, 15], *i.e.* per-vertex offsets upon the body in the canonical pose, and extend it to apply to temporal sequences. However, as discussed in CAPE [15], some loose garments such as skirts and coats are difficult to represent with offsets due to the limited capacity. Modeling clothes and hair as separate layers on the body with meshes or implicit surface is a feasible way, and we leave it to future work. **Forth**, since we have a compact motion representation that uses one single motion code to provide global control upon the whole sequence, future works also include high-level inference applications such as using the motion code learned in an unsupervised fashion to perform action classification with a simple linear classifier.

8. Broader Impact and Social Impact

Learning a compact representation for 3D data is a widely interested problem. However, less attention has focused on the 4D cases, though it is important for various applications to understand time-varying objects, *e.g.* Robotics, VR/AR. This work focuses on 4D human modeling and proposes H4D, a compact and compositional representation, which uses low-dimensional SMPL parameters and latent codes to encode key factors of dynamic humans. We make some attempts and demonstrate our representation has rich capacity and is amenable to many applications. We hope these explorations could provide insights for future research directions. For instance, using our representation for video-based full human reconstruction; exploiting the compositional property to control the outputs for generative tasks; and improving the 4D human representation and make up for the discussed limitations of our method. Broadly, our

approach can serve as an important core tech in achieving the Metaverse. It may enable everyone to produce their own Avatar with their motions, potentially benefiting the Social Welfare.

References

[1] Thiemo Alldieck, Gerard Pons-Moll, Christian Theobalt, and Marcus Magnor. Tex2shape: Detailed full human body geometry from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2293–2303, 2019. 5

[2] Bharat Lal Bhatnagar, Cristian Sminchisescu, Christian Theobalt, and Gerard Pons-Moll. Combining implicit function learning and parametric models for 3d human reconstruction. In *European Conference on Computer Vision (ECCV)*. Springer, aug 2020. 3

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018. 5

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1

[5] Boyan Jiang, Yinda Zhang, Xingkui Wei, Xiangyang Xue, and Yanwei Fu. Learning compositional representation for 4d captures with neural ode. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5340–5350, 2021. 3, 5

[6] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018. 1

[7] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 3

[8] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[9] Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5253–5263, 2020. 1

[10] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2252–2261, 2019. 1

[11] Verica Lazova, Eldar Insafutdinov, and Gerard Pons-Moll. 360-degree textures of people in clothing from a single image. In *2019 International Conference on 3D Vision (3DV)*, pages 643–653. IEEE, 2019. 5

[12] Chen-Hsuan Lin, Oliver Wang, Bryan C Russell, Eli Shechtman, Vladimir G Kim, Matthew Fisher, and Simon Lucey. Photometric mesh optimization for video-aligned 3d object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 969–978, 2019. 5

[13] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. 5

[14] Shaohui Liu, Yinda Zhang, Songyou Peng, Boxin Shi, Marc Pollefeys, and Zhaopeng Cui. Dist: Rendering deep implicit signed distance function with differentiable sphere tracing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2019–2028, 2020. 5

[15] 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 *Computer Vision and Pattern Recognition (CVPR)*, 2020. 3, 5

[16] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5442–5451, 2019. 3

[17] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[18] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 5

[19] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019. 1, 5

[20] Pablo Palafox, Aljaž Božič, Justus Thies, Matthias Nießner, and Angela Dai. Npms: Neural parametric models for 3d deformable shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12695–12705, 2021. 3

[21] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1

[22] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 3

[23] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11488–11499, 2021. 3

[24] Raquel Urtasun, David J Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3d human body tracking. *Computer vision and image understanding*, 104(2-3):157–177, 2006. 2

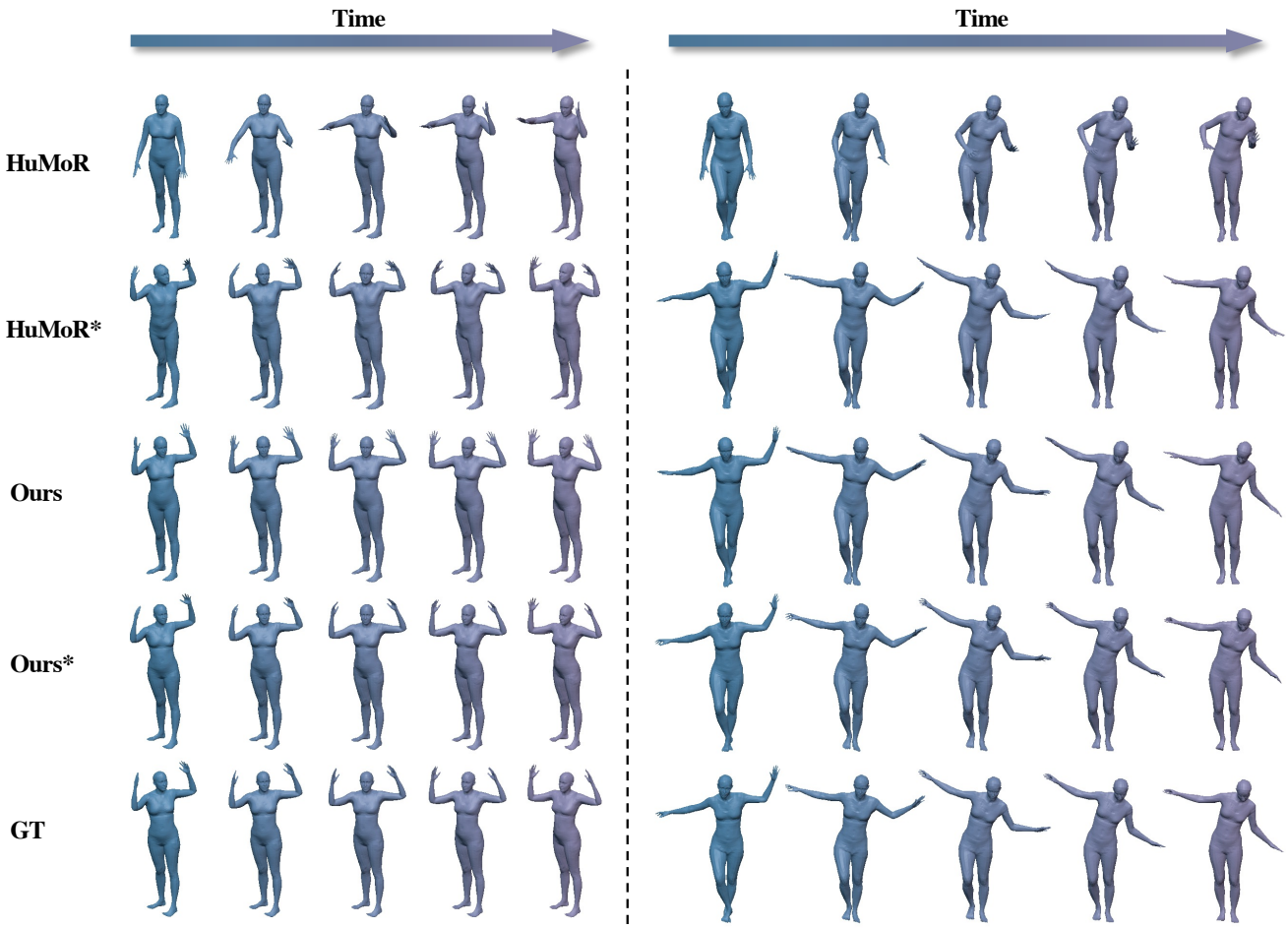


Figure 3. **Qualitative comparisons to HuMoR.** We compare the reconstruction results with HuMoR by using Chamfer loss (HuMoR, Ours) and 3D keypoint loss (HuMoR*, Ours*) for auto-decoding. Our method produces more accurate SMPL sequences in both cases.

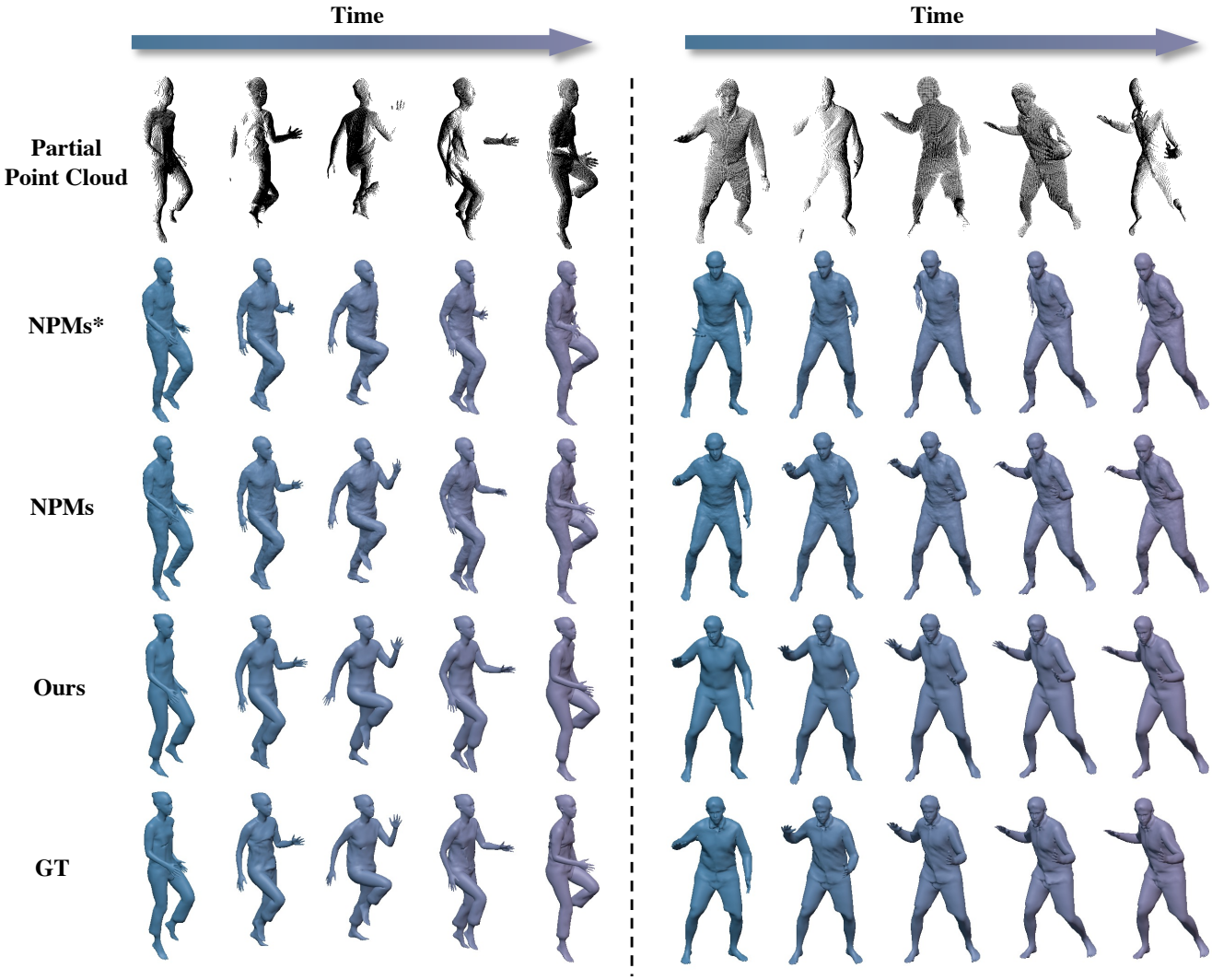


Figure 4. **Qualitative comparisons to NPMs on depth completion task.** We use the partial point clouds back-projected from the depth images rendered by rotating a camera around the performer as observation, and reconstruct the motion sequence with complete geometry. We use the random vectors sampled from a Gaussian distribution $N\left(0, 0.01\right)$ to initialize the codes for NPMs* and Ours, and use the pretrained encoders to obtain initialization for NPMs.

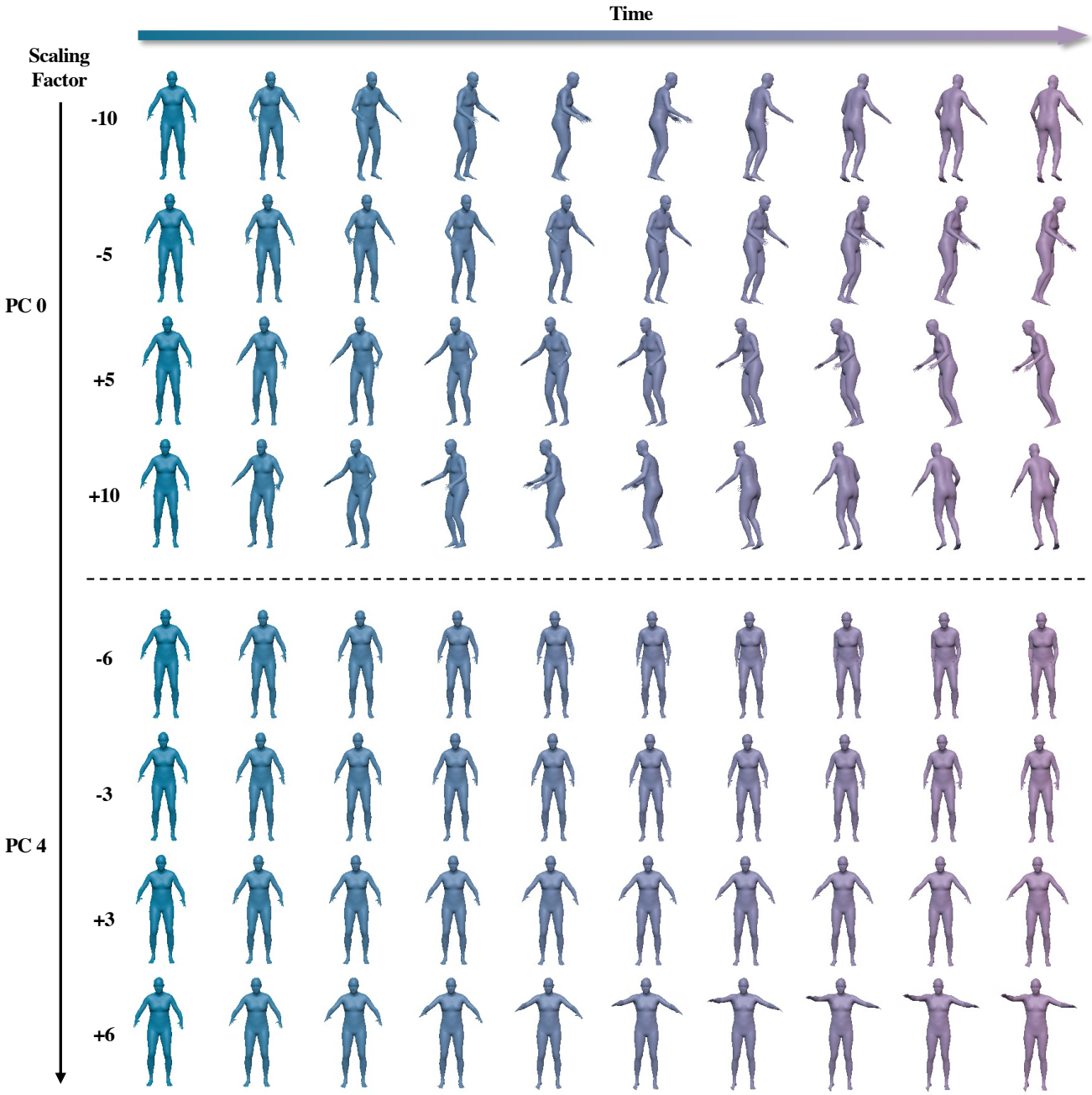


Figure 5. **Visualization of principal components (1).** PC0 and PC4 is the first principal component of global rotation and other body joint rotations respectively. The number before each row is the scaling factor for the corresponding component (multiply it with the eigenvector and show the result motion).

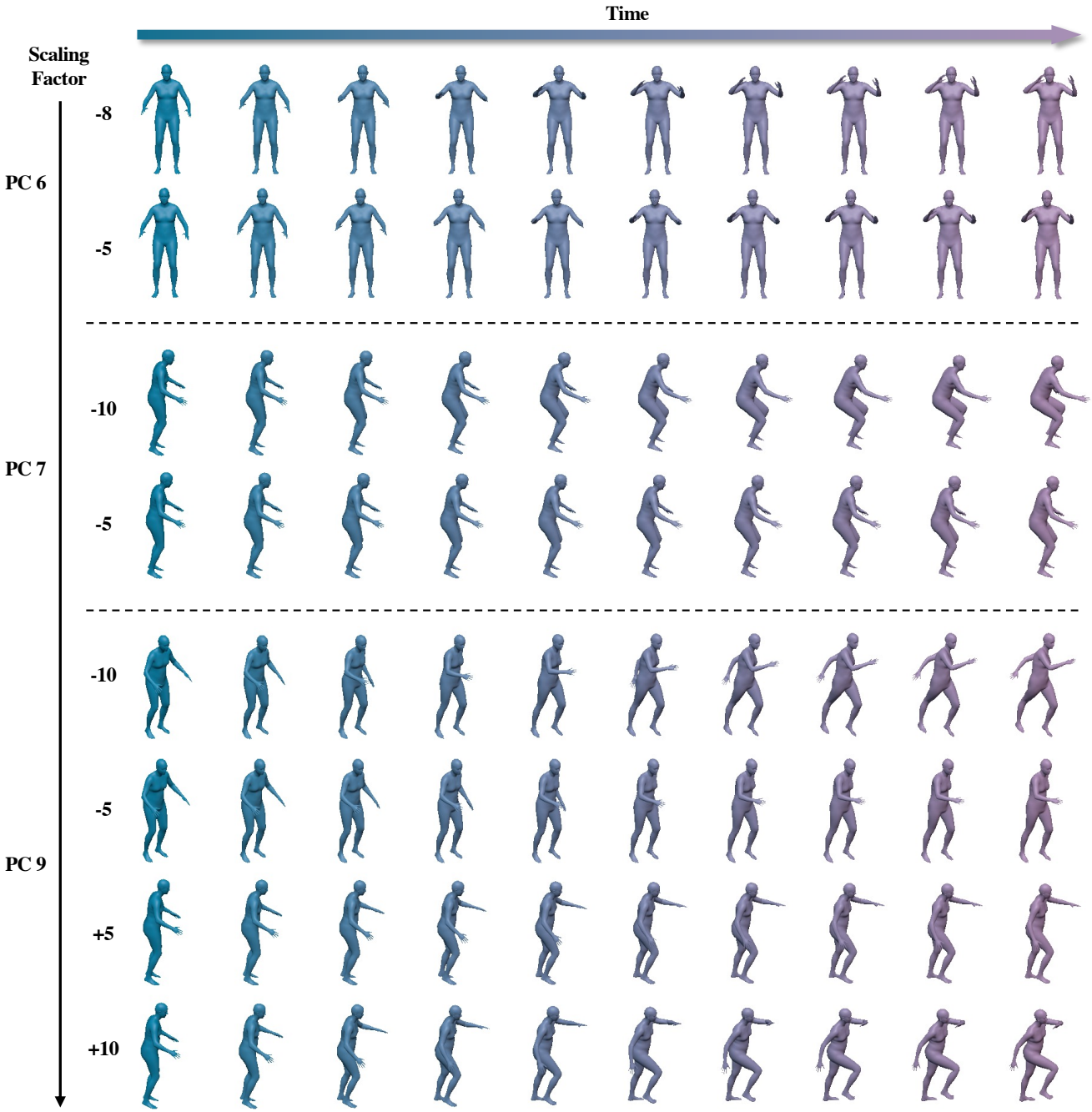


Figure 6. **Visualization of principal components (2).** Here are three principal components of body joint rotations, which in general control the movements of forearms (PC6), the bending of the legs (PC7), and motion similar to running (PC9) respectively.

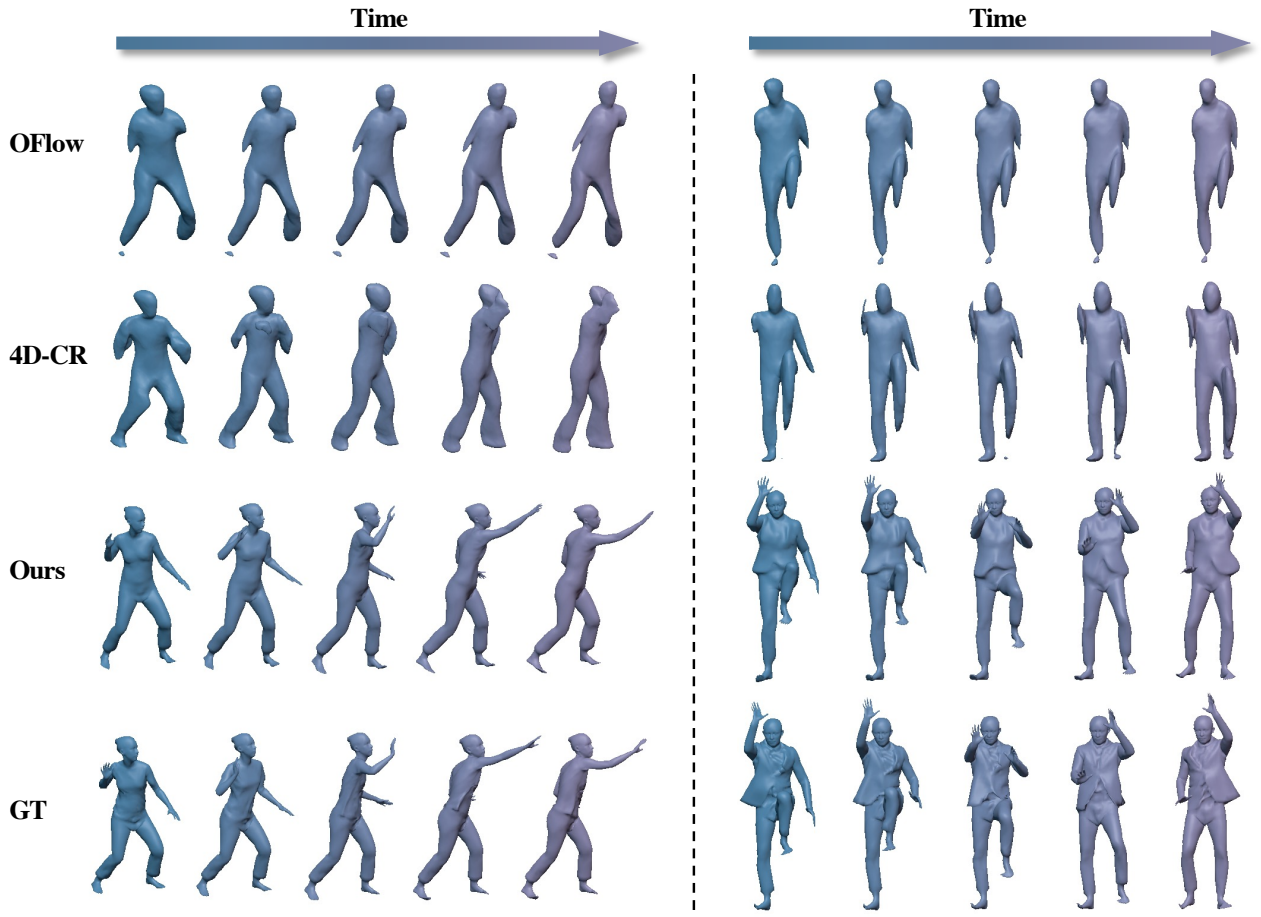


Figure 7. **4D Reconstruction.** Our method produces accurate motion sequence with fine-grained geometry (blazer, long trousers and hairstyle), while the results of baseline methods suffer from incomplete geometry with missing arms or hands, and are overly smooth.

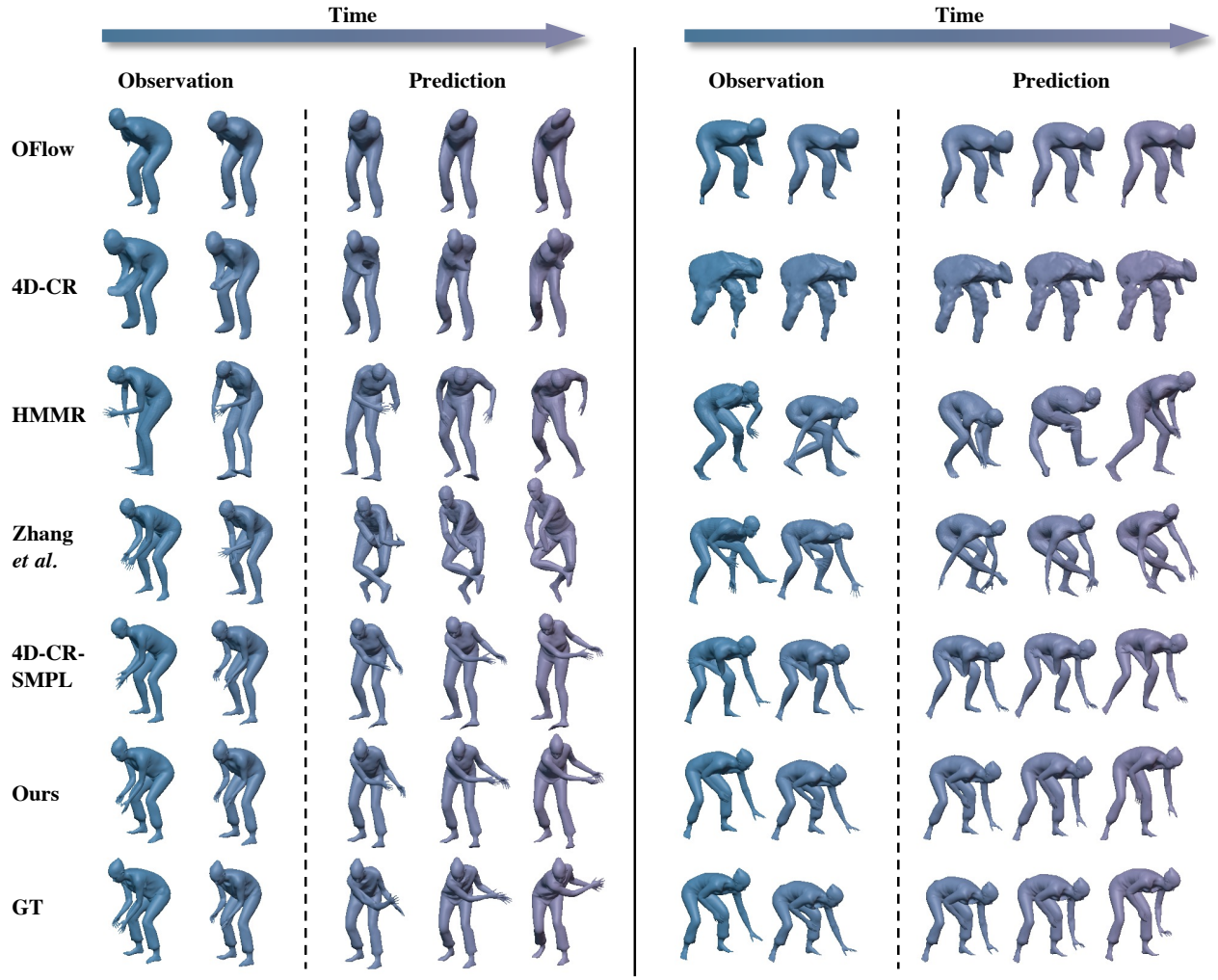


Figure 8. **Future Prediction.** Here are two different sequences on the future prediction task (split by solid line). Each sequence has $L = 30$ frames and we are aiming to extrapolate 10 future temporal frames based on 20 past observed frames. The meshes on the left of dotted line are reconstruction results of the observations, and the meshes on the right are the predictions for future time steps.

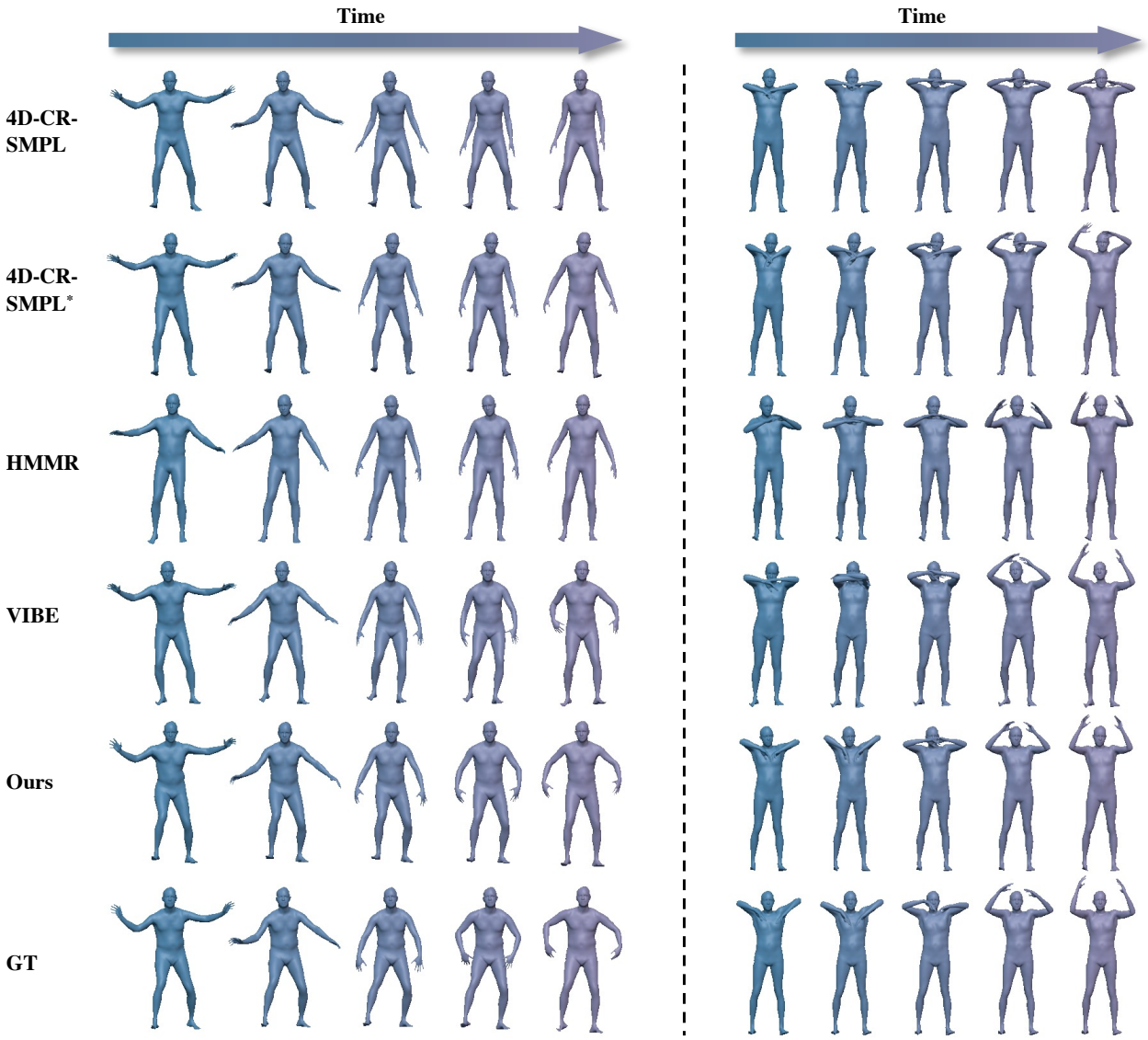


Figure 9. **Shape and Motion Recovery.** Different from the 4D reconstruction task, the goal here is to recover accurate SMPL motion sequence from the input point cloud sequence. We uniformly sample 5 frames (out of 30 frames) for visualization. Our model in general performs best among all the methods.

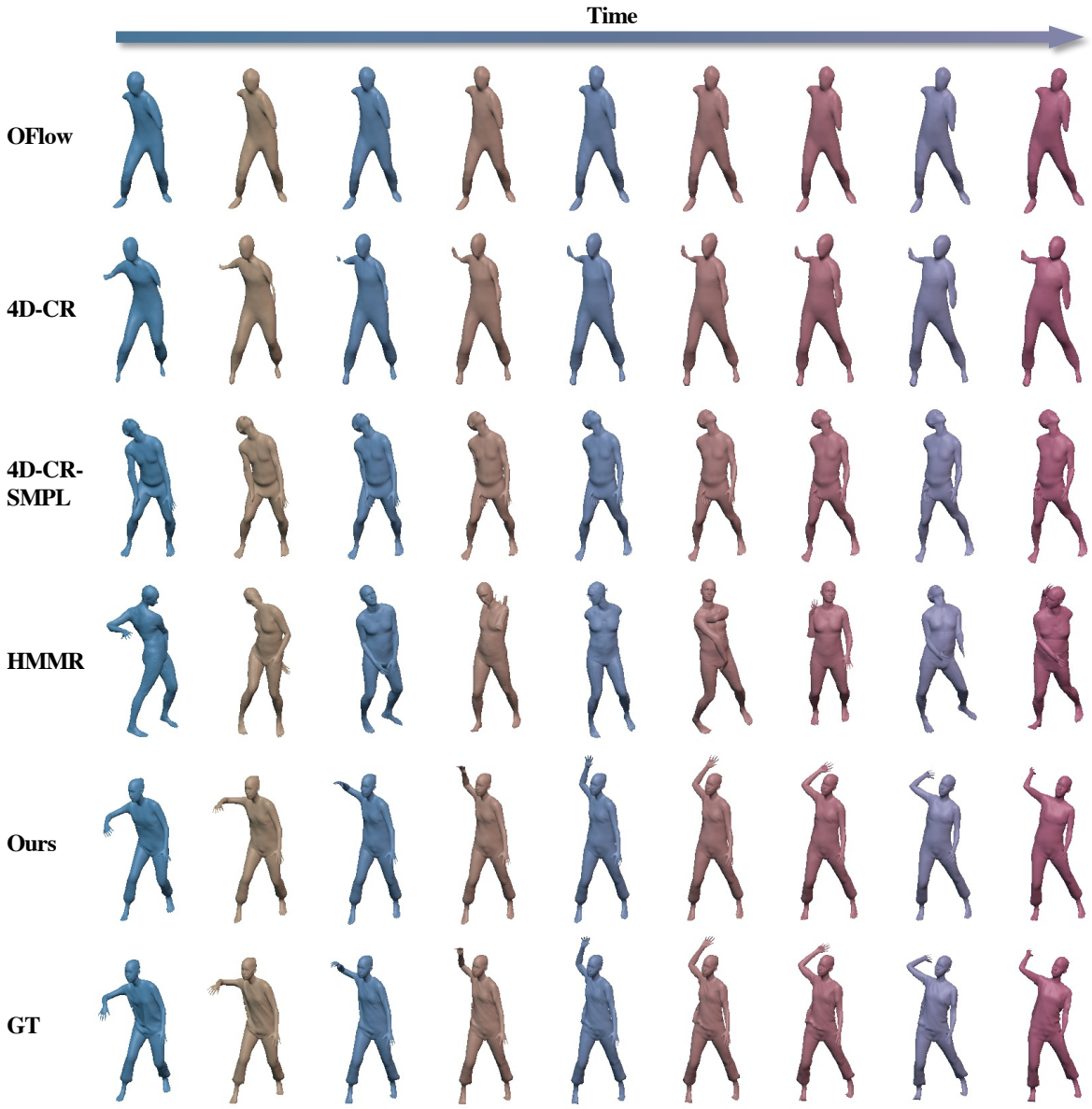


Figure 10. **Temporal Completion.** Given a sequence of $L = 30$ frames, we randomly select 15 frames as observation and perform the backward fitting algorithm to optimize the SMPL parameters and latent codes, and then reconstruct the full sequence to complete the missing frames. The meshes with yellow-red-ish color are completed unseen frames. We find 4D-CR-SMPL and HMMR produce unnatural pose results while our method successfully reconstructs and completes the full motion sequence, possibly because our linear motion model provides regularization and global temporal context for the output motion.

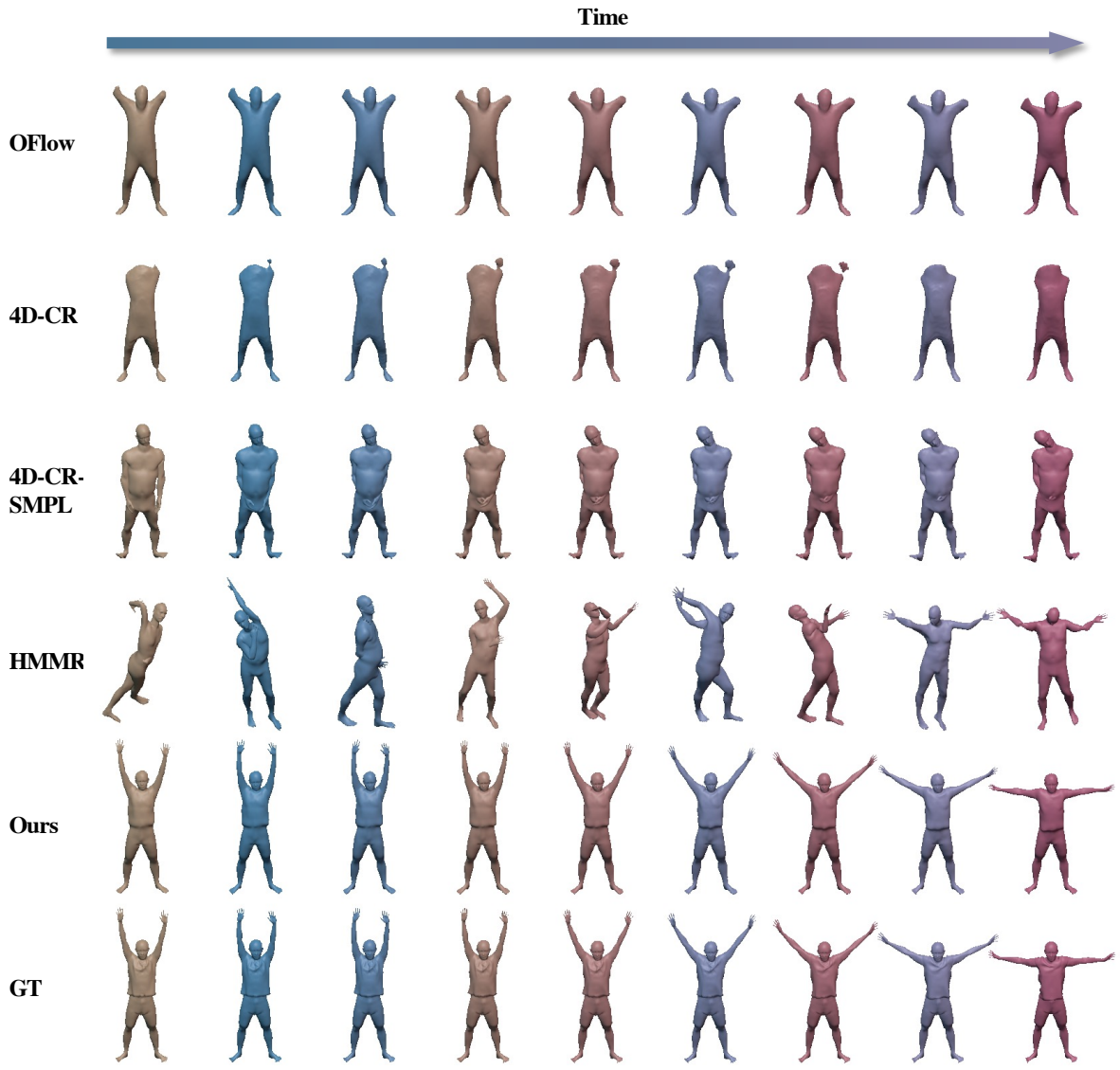


Figure 11. **Temporal Completion.** The meshes with yellow-red-ish color are completed unseen frames.

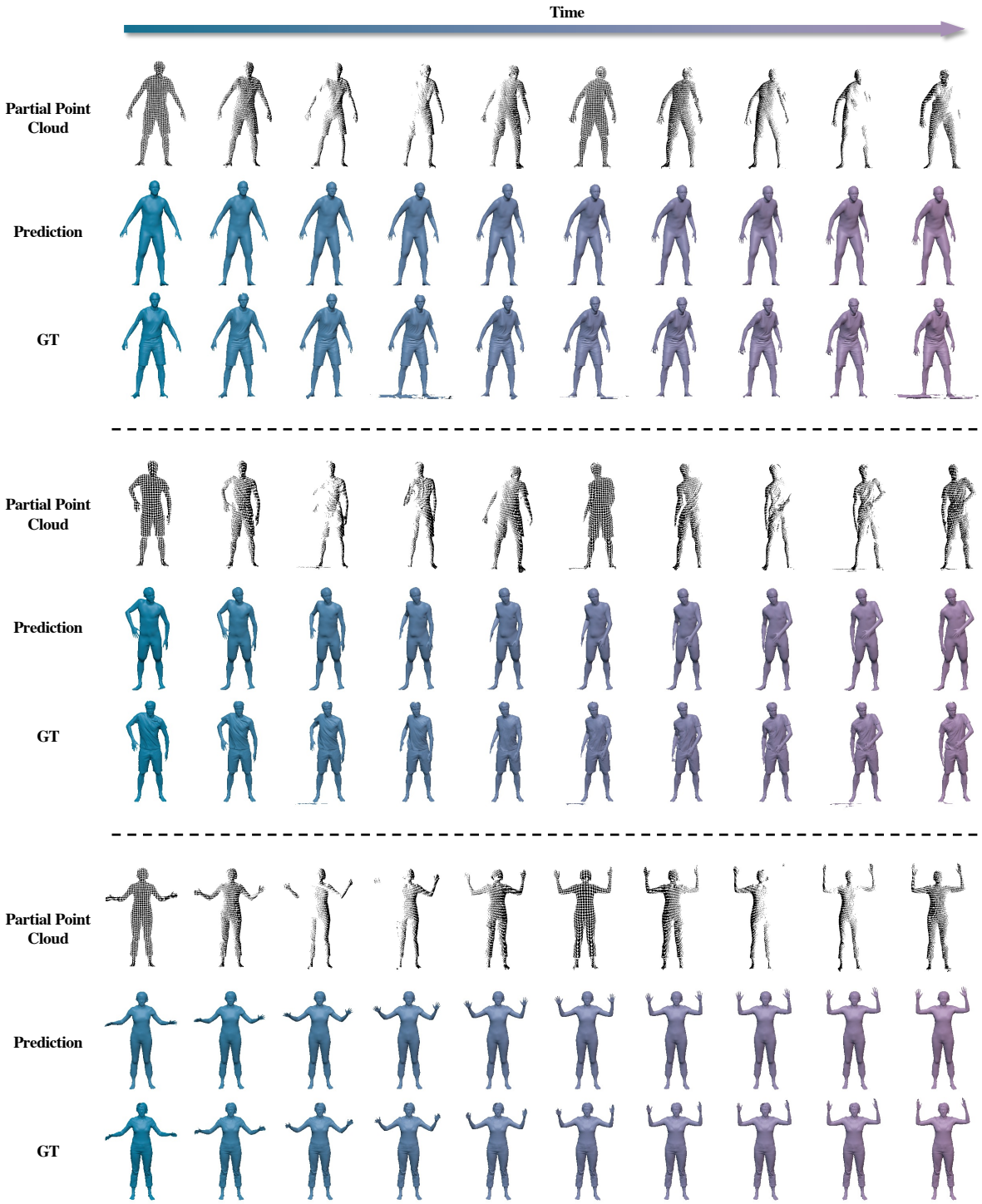


Figure 12. **Spatial Completion.** We use the partial point clouds back-projected from the depth images as observation, and reconstruct the motion sequence with complete geometry. Note that the depth images are rendered from the raw scanned mesh sequence of CAPE dataset, which simulates the real-world scenarios.

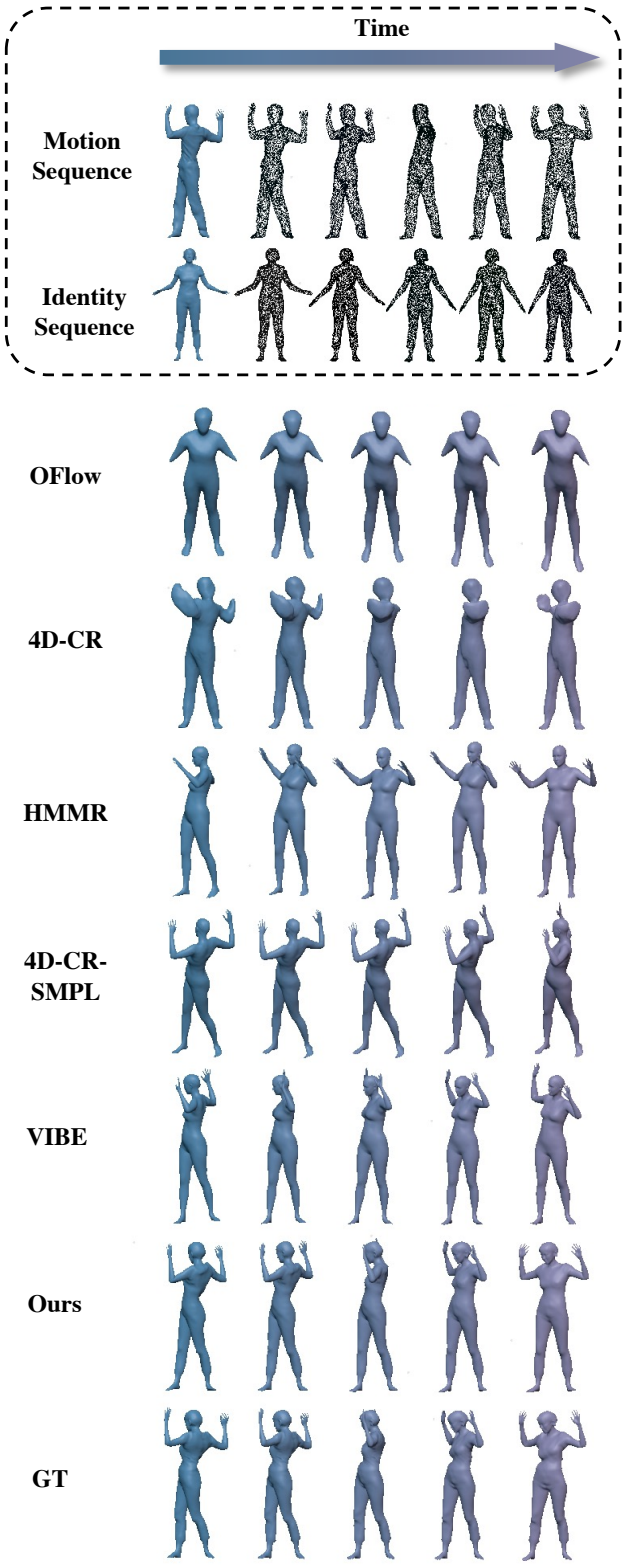


Figure 13. **Motion Retargeting.** Our goal is to transfer the human movements of the motion sequence (Row 1) to the people in the identity sequence (Row 2).

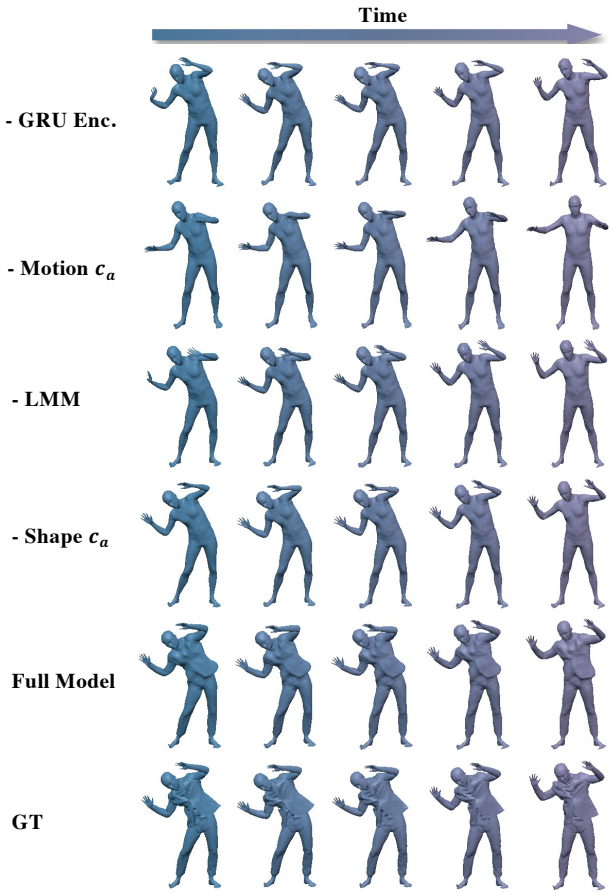


Figure 14. **Ablation Study.**