Supplementary Matarial
Dynamic Mesh Recovery from Partial Point Cloud Sequence

Hojun Jang[1] Minkwan Kim[1] Jinseok Bae[1] Young Min Kim[1,2]

[1] Dept. of Electrical and Computer Engineering, Seoul National University

[2] Interdisciplinary Program in Artificial Intelligence and INMC, Seoul National University

{j12040208, mkjjang3598, capoo95, youngmin.kim}@snu.ac.kr

A. Additional Implementation Details

In this section, we provide additional details of our method that could not be included in the main paper. The order of the description follows that of Section 3 in the main paper.

A.1. Kinematics Learner

Figure A.1 shows the network architecture related to generating pose parameters, namely the feature encoder and the pose generator, which is part of the parameter estimator. The feature encoder receives the point cloud input and outputs the encoded feature, which is passed to the pose generator. Note that the sequence of motion is further processed with a transformer encoder and decoder. The VAE architecture of the pose generator fits a Gaussian distribution for the latent distribution such that the input pose parameters can be closely reproduced by the encoder-decoder pair. Specifically, the VAE architecture generates the posterior and the prior distribution of the latent space z conditioned on the encoded feature. To form the posterior distribution, we concatenate the ground truth pose parameters to the feature and pass it to the MLP layer. We concatenate the latent vector z_t, which is sampled from the distribution, with the encoded feature F_t and the positional encoding information. Finally, the concatenated vectors are passed to the transformer decoder to form a sequence of pose parameters.

We also use the encoded features $\mathbf{F}$ to estimate the root translations and the shape parameter of the mesh. Each F_t passes through the translation estimator and the shape estimator, which consists of an MLP layer. To make the shape parameter to be constant throughout the sequence, we output the average of the estimated parameters.

A.2. Feature Follower

The network architecture of the feature follower is identical to the feature encoder in the kinematics learner shown in Figure A.1-(a). We train the feature follower to output the feature encoding similar to the output of the kinematics learner.

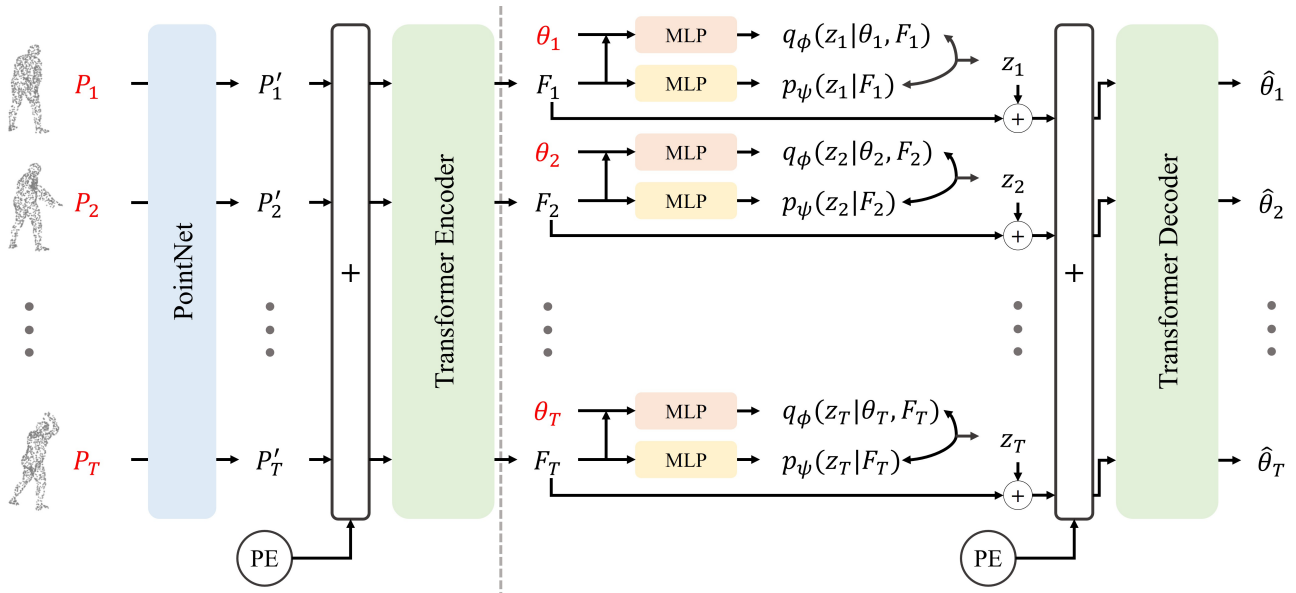
B. Dataset Details

B.1. Human Motion

We use AMASS dataset [4] to train the kinematics of the human motion. Since the dataset used for training and testing the feature follower is SMPL [3] parameters from CMU dataset [1], we train our kinematics learner excluding CMU. Also, as we make the input sequence to be 10 fps and the total number of frame to be 40, we exclude the sequence shorter than 4 seconds. From the sequences longer than 9 seconds, we split the sequence in the period of 5 seconds from the beginning and if the last split has the sequence shorter than 4 seconds, it is merged to the former split. For example, if the length of the sequence is 16 seconds, the frames from the beginning to 5 second will be the first split and the frames from 5 second to 10 second will be the second split. The last split will have the frames from 10 second to the last, 16 second. When training, the sequences longer than 5 seconds are cropped randomly to generate 10 fps, 40 frames input sequence. Finally, the total number of sequences we use to train the kinematics learner is 17,481.

Now, we outline the dataset used to train the feature follower. After generating the depth image by projecting the SMPL model of CMU, using the same process of making SURREAL dataset [8], we obtain single-view point clouds which are made to seem as if they are obtained from a single depth camera. We also split the sequence in the period of 5 seconds similar to the process explained above. The number of generated single-view point cloud sequences is 2,759. Then, we spare 200 sequences to the test set randomly and the rest to the train set of the feature follower.

To generate the motion sequence to train our model for the spatially partial sequence input, we manipulate the depth image. We erase the depth information which is in a random box, which is randomly sized and randomly placed



(a) Feature Encoder (b) Pose Generator

Figure A.1. The process to generate pose parameters from the input point cloud sequence. (a) shows the feature encoding process and (b) shows the network architecture of the pose generator. The variables which are written in red letters are the given inputs.

in image. By making a point cloud sequence from that randomly masked depth image, it is possible to obtain a spatially partial point cloud sequence.

B.2. Hand Motion

Since hand motion has less diversity than that of human motion, we use the dataset having a smaller scale than the scale of the human motion dataset. We utilize two MANO [7] fitted hand motion datasets, HanCo [11, 10] and InterHand2.6M [5]. To train the kinematics learner for the hand motion input, we use the given train and validation split of the both datasets. Also, we make the input sequence of a hand motion to be 5 fps with 40 frames total in a similar way we make the dataset for human motion. As a result, we use 1,753 sequences to train the kinematics learner for hand motion. The sequences, which are not used when training the kinematics learner, are used to train and test the feature follower.

We make use of Open3D library [9] to generate single-view point cloud of a hand. We first render 3D mesh of a hand and then obtain the depth image from a viewpoint, which is randomly chosen from an orbit rotating around the hand. Then, we make the single-view point cloud of a hand sequence using the Open3D internal function. Since the point clouds obtained from the Open3D library are too smooth, we add random noise with σ_{noise} 0.01 to the point clouds. We set the sequences from `capture id: 0` in InterHand2.6M as the test split of the feature follower, which includes 75 hand motion sequences. The remaining sequences are being the training split of the feature follower, and the number of the sequences in the training split is 172.

C. Hyperparameter Setup

We show the hyperparameter settings of our method for training the kinematics learner and the feature follower, respectively. The weights for the loss terms do not vary through the input type. That is, loss weights for the human motion and the hand motion are the same. Also, the weights for the noisy sequence or the partial input are set equally.

The total epoch of the training is written as N_{epoch}. When the number of epoch reaches $N_{\text{first\_decay}}$ and $N_{\text{second\_decay}}$, the learning rate lr reduces to $lr/4$ and $lr/10$, respectively. $\texttt{max}_{\text{block\_rate}}$ denotes the maximum masking ratio of the sequence, the masking ratio is set randomly below $\texttt{max}_{\text{block\_rate}}$. Block rate 0.4 means $T_{\text{Tot}} \times 0.4$ input time steps are randomly masked. D_P refers to the dimension of the PointNet [6] feature and D_F is the dimension of the Transformer feature, both shown in Section 3.1 in the main paper. $D_{TF,L}$ and $D_{TF,ff}$ are the latent dimension and the dimension of the feedforward network model of the Transformer layer, respectively. $N_{TF,\text{Layer}}$ is the number of layers in the Transformer network and $N_{TF,\text{Head}}$ is the number of heads in the multi-head attention models.

C.1. Kinematics Learner

We set the the hyperparameters for the kinematics learner as in Table C.1. Additionally to the hyperparame-

Train-related	N_{epoch}	4,000
	$N_{\text{first\_decay}}$	3,000
	$N_{\text{second\_decay}}$	3,500
	lr	0.0005
Input-related	# input point cloud	1,024
	T_{Tot}	40
	N_{β}	10
	$\texttt{max}_{\text{block\_rate}}$	0.4
Network-related	D_P	512
	D_F	512
	$D_{TF,L}$	256
	$D_{TF,ff}$	1,024
	$N_{TF,\text{Layer}}$	4
	$N_{TF,\text{Head}}$	8
Loss-related	w_{θ}^{aux}	0.5
	w_{θ}	0.5
	w_J	0.5
	w_{vol}	1.0
	w_{β}	0.1
	w_{KL}	1.0

Table C.1. Hyperparameter setups to train the kinematics learner.

ters written in the given table, the number of pose parameters for the human motion datasets is 66 and the number of joints is 24. Different from the human motion datasets, the number of pose parameters for the hand motion datasets is 48 and the number of joints is 21.

C.2. Feature Follower

We report the hyperparameter setups to train the feature follower in Table C.2. The number of pose parameters and the joints are equivalent to the training of the kinematics learner.

Train-related	N_{epoch}	2,000
	$N_{\text{first\_decay}}$	1,200
	$N_{\text{second\_decay}}$	1,600
	lr	0.0001
Input-related	# input point cloud	1,024
	T_{Tot}	40
	N_{β}	10
	$\texttt{max}_{\text{block\_rate}}$	0.4
Network-related	D_P	512
	D_F	512
	$D_{TF,L}$	256
	$D_{TF,ff}$	1,024
	$N_{TF,\text{Layer}}$	4
	$N_{TF,\text{Head}}$	8
Loss-related	w_F	1.0
	w_{θ}^{aux}	0.5
	w_{θ}	0.5
	w_J	0.5
	w_{β}	0.1

Table C.2. Hyperparameter setups to train the feature follower.

D. Additional Qualitative Results

D.1. Full Sequence Input

We show the mesh recovery results of the pretrained kinematics learner. The kinematics learner reconstructs mesh from the full point cloud sequence input.

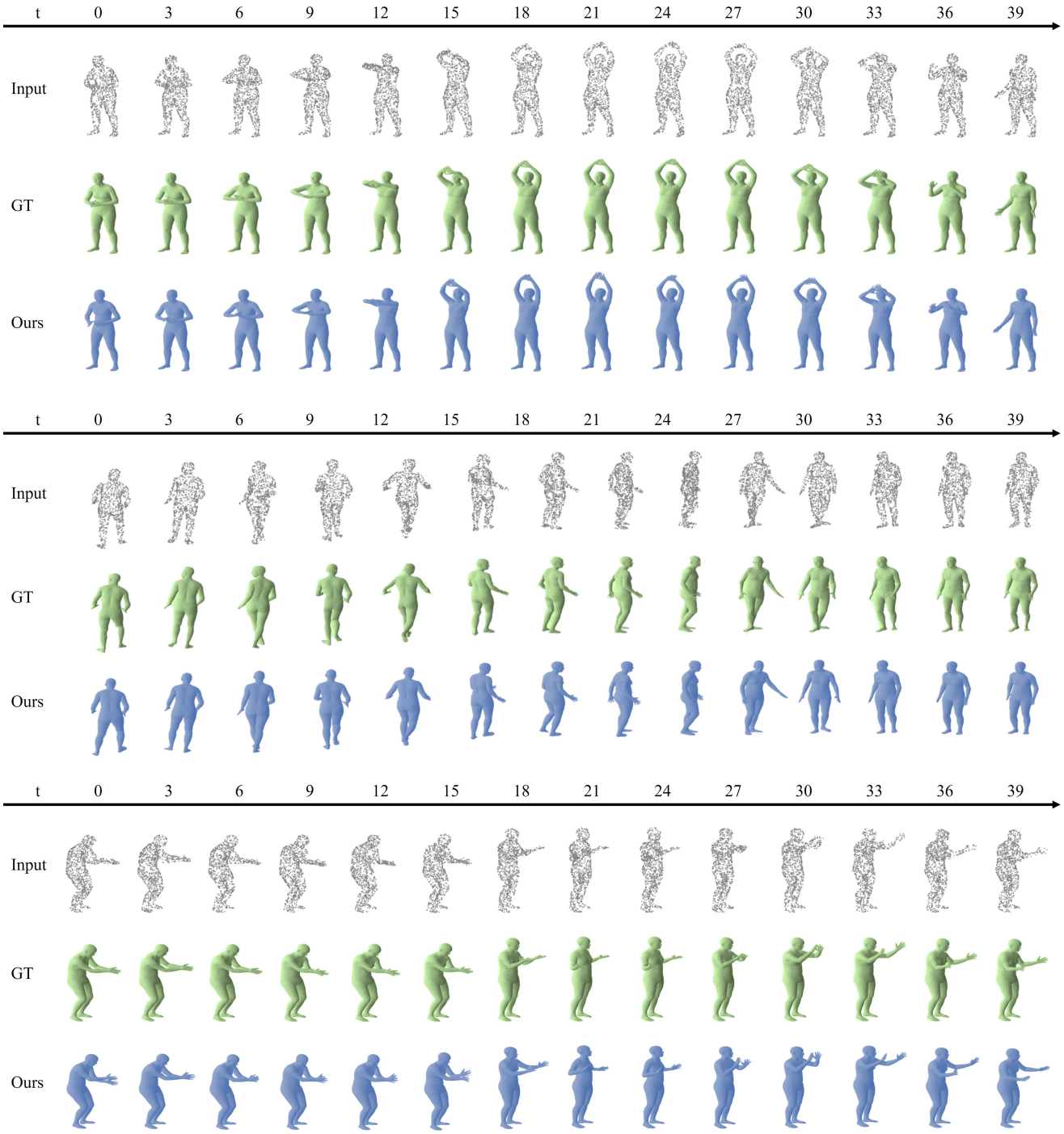


Figure D.1. The result of the kinematics learner reconstructing the mesh sequence from the full point cloud sequence input.

D.2. Noisy Sequence Input

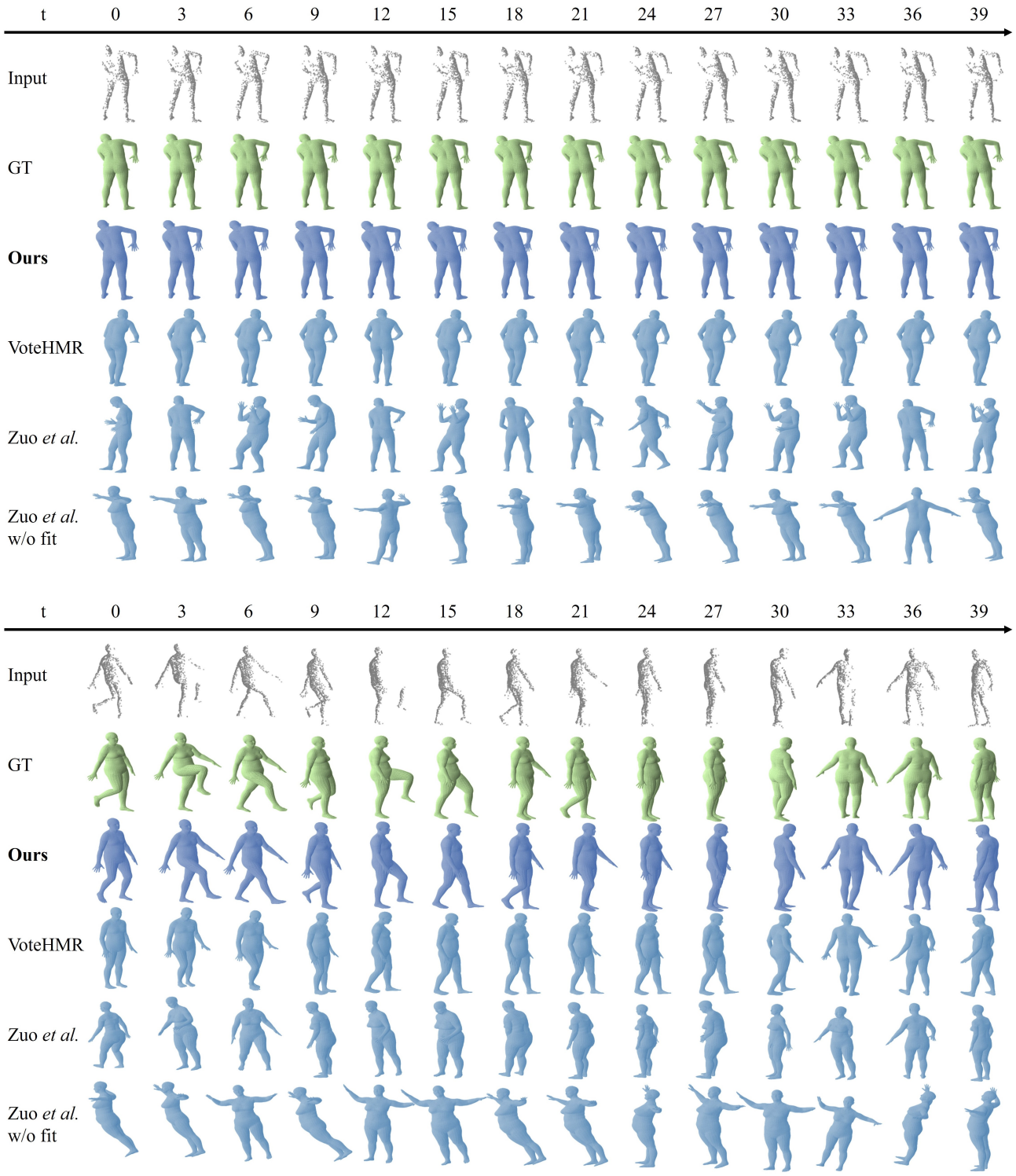


Figure D.2. The results of the feature follower against the baselines in noisy single-view point cloud sequence input. Our method shows the best qualitative performance over the baselines, VoteHMR [2], Zuo *et al*. [12] without optimization, and Zuo *et al*. with additional optimization.

D.3. Spatially Partial Sequence Input

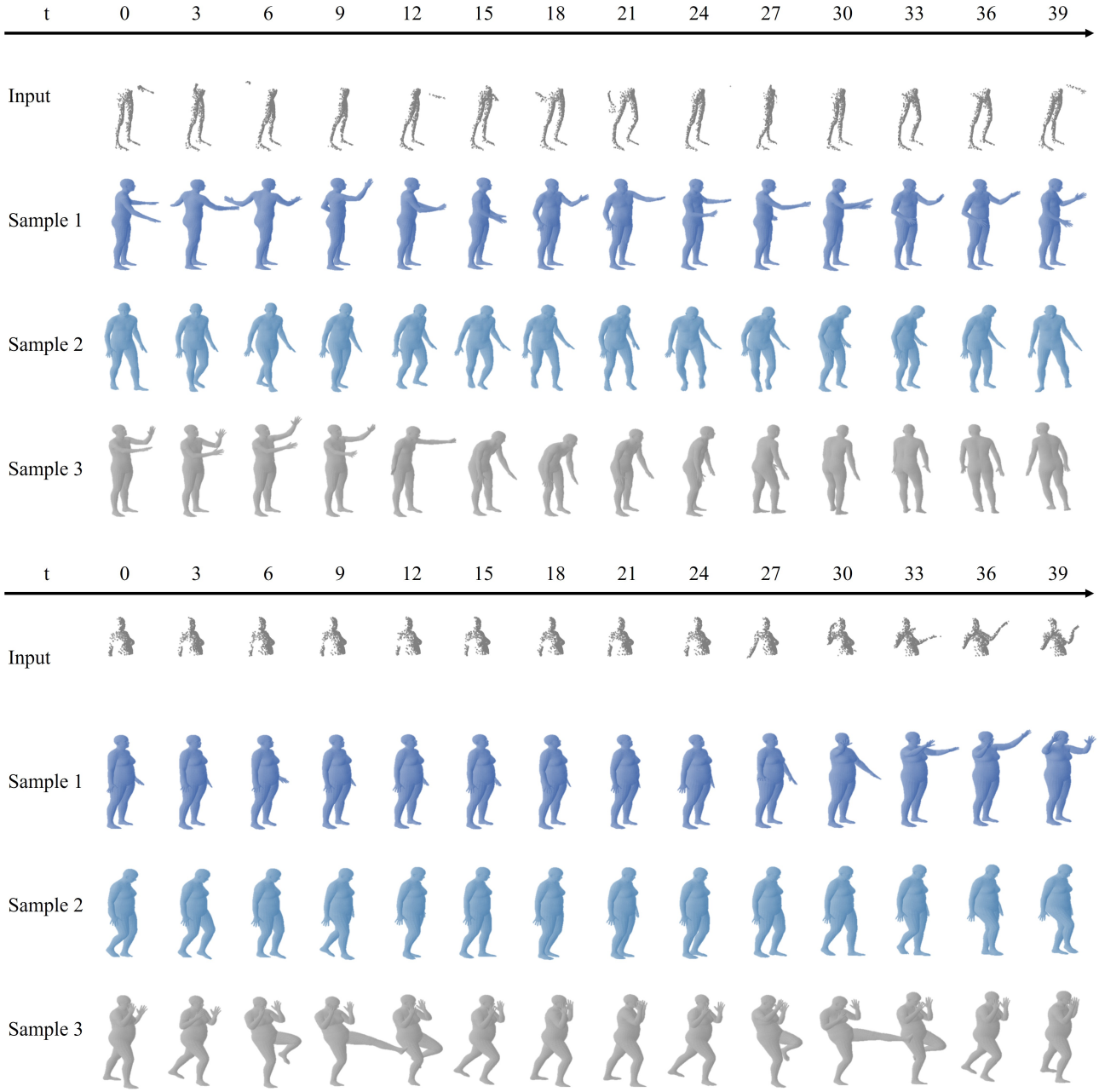


Figure D.3. The results showing our model generating multiple plausible poses under severely sparse observations. We show sample 3 latent vectors for the pose generation and decode them to generate plusible mesh sequences.

D.4. Temporally Partial Sequence Input

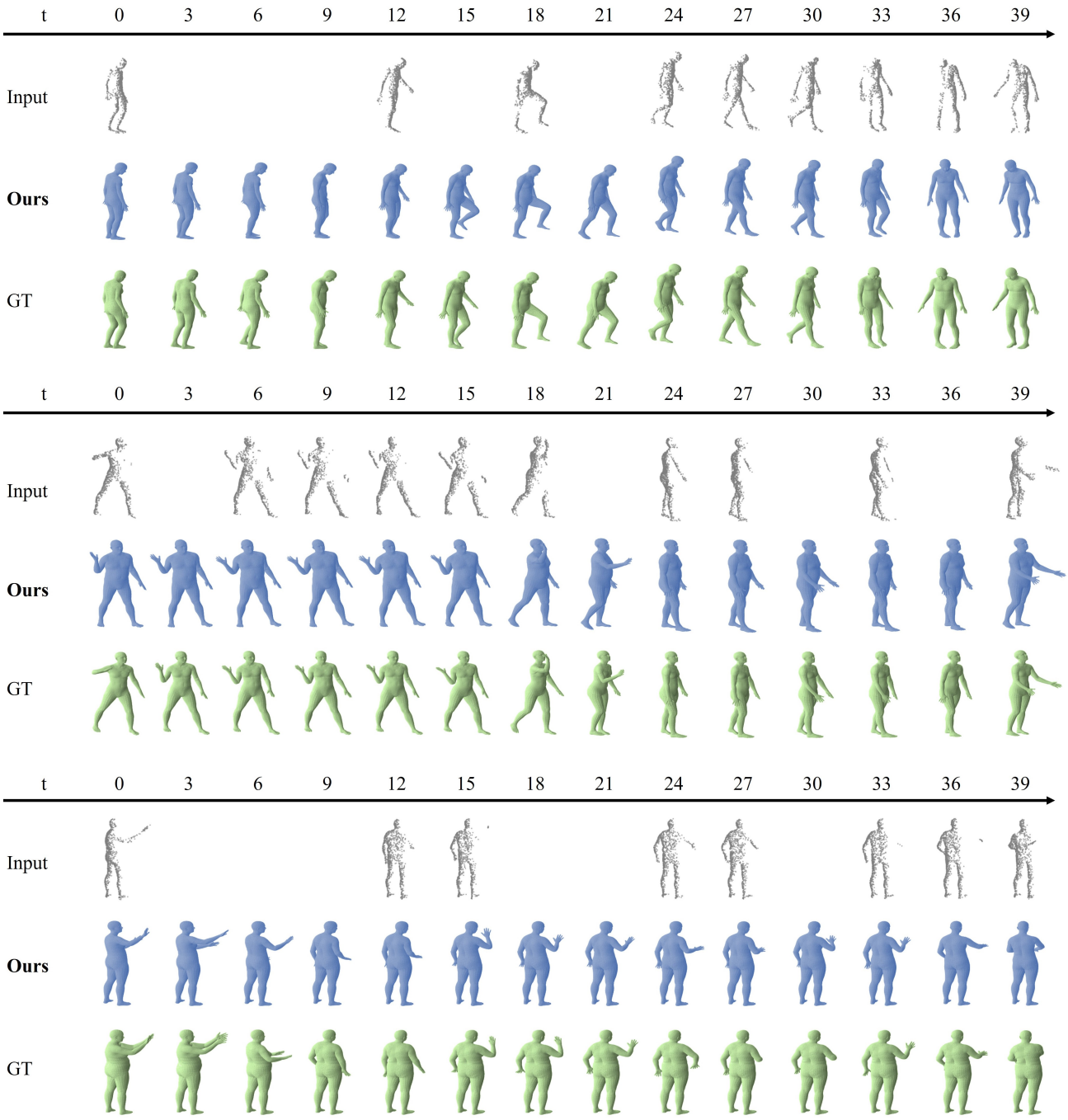


Figure D.4. The result of our model reconstructing mesh sequence even on sequence with empty observations.

D.5. Real Sequence Input

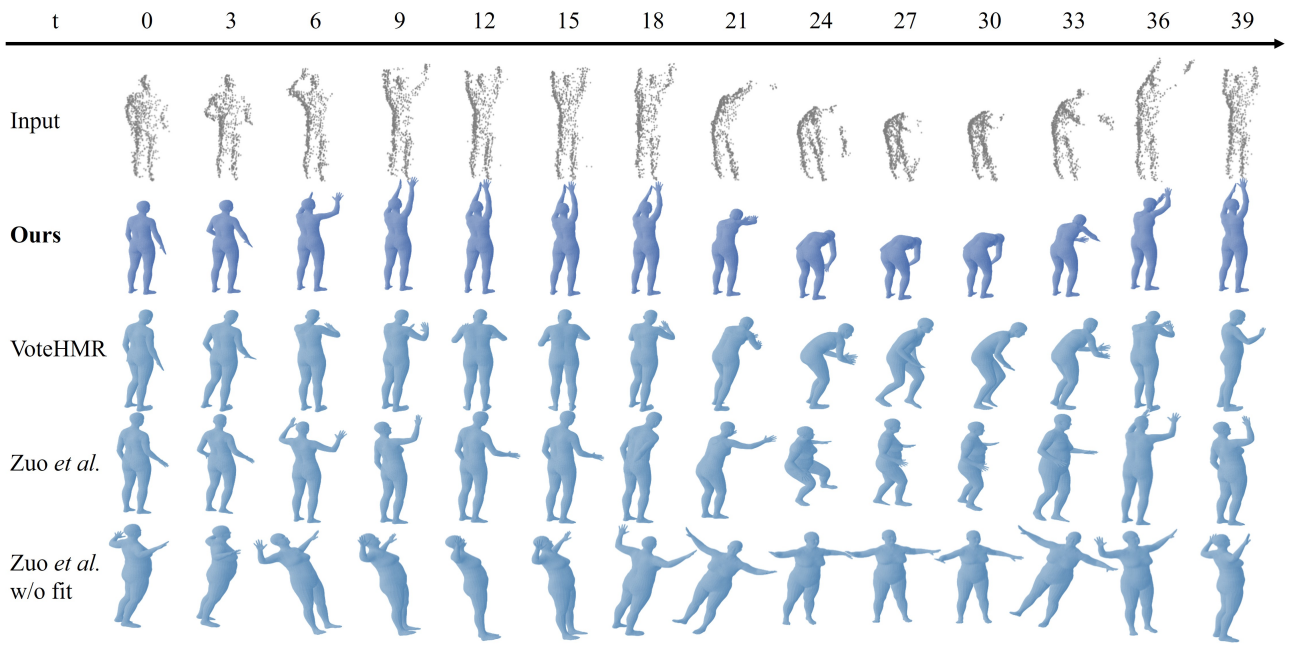


Figure D.5. Mesh recovery results of our method and the baselines. Zuo *et al*. [12] without additional optimization cannot really match the point cloud and is inconsistent over the sequence. VoteHMR [2] outputs better result than the Zuo *et al*. [12] without fitting but the performance is still weak. The mesh sequence from our model seems to match the best to the point cloud sequence over the baselines.

References

[1] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *The IEEE International Conference on Computer Vision (ICCV)*, 2015. 1

[2] Guanze Liu, Yu Rong, and Lu Sheng. Votehmr: Occlusion-aware voting network for robust 3d human mesh recovery from partial point clouds. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 955–964, New York, NY, USA, 2021. Association for Computing Machinery. 5, 8

[3] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015. 1

[4] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision (ICCV)*, pages 5442–5451, Oct. 2019. 1

[5] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. Interhand2.6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image. In *European Conference on Computer Vision (ECCV)*, 2020. 2

[6] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 2

[7] Javier Romero, Dimitrios Tzionas, and Michael J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6), Nov. 2017. 2

[8] Gül Varol, Javier Romero, Xavier Martin, Naureen Mahmood, Michael J. Black, Ivan Laptev, and Cordelia Schmid. Learning from synthetic humans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[9] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 2

[10] Christian Zimmermann, Max Argus, and Thomas Brox. Contrastive representation learning for hand shape estimation. *arXiv preprint arXiv:2106.04324*, 2021. 2

[11] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 813–822, 2019. 2

[12] Xinxin Zuo, Sen Wang, Minglun Gong, and Li Cheng. Unsupervised 3d human mesh recovery from noisy point clouds. *CoRR*, abs/2107.07539, 2021. 5, 8