LoRD: Local 4D Implicit Representation for High-Fidelity Dynamic Human Modeling Supplementary Material

Boyan Jiang[1] Xinlin Ren[1] Mingsong Dou[2] Xiangyang Xue[1†]
Yanwei Fu[1†] Yinda Zhang[2†]

[1]Fudan University [2]Google

In this supplementary material, we provide implementation details, additional experimental results, visualization of the surface deformation within a local part, additional qualitative results, and discussions about limitations and future work of our approach.

1 Implementation Details

1.1 Network Architecture

Motion model We adapt the architecture of the IMNet [1] for our motion model. As shown in Fig. 1 (a), the input is the concatenation of: the motion code $c_m \in \mathbb{R}^{128}$, 3D query point $\mathbf{x} \in \mathbb{R}^3$ and normalized time value $T \in \mathbb{R}^1$. The network is based on multi-layer perceptrons with the skip connection to copy the input to concatenate with the output feature of the first 4 layers, each layer has the nonlinear activation of LeakyReLU ($\alpha = 0.2$) [8] except the last layer. We follow LIG [4] to reduce the feature dimension of each hidden layer by 4 fold to obtain an efficient motion model. The output of our motion model is a deformation vector $\mathbf{x}^* \in \mathbb{R}^3$ that transforms the given point to its position in the space of the canonical frame, i.e. $T = 0$.

Canonical shape model The canonical shape model uses the auto-decoder network proposed in DeepSDF [13], which is shown in Fig. 1 (b). The input is the concatenation of the canonical shape code $c_s \in \mathbb{R}^{128}$ and 3D query point $\mathbf{x} \in \mathbb{R}^3$, and the network predicts a signed distance value $\mathbf{s} \in \mathbb{R}^1$ for the given point. We reduce the number of hidden layers from 8 to 6 and the feature channels from 512 to 256 for the efficiency. And following IGR [2], the softplus activation ($\beta = 100$) and geometric initialization are also used.

Texture model The texture model is used to produce the colored results in Fig. 1 and 5 of the main paper. We modify the decoder architecture proposed in Texture Fields [11] for our 4D texture inference, and the architecture is shown in Fig. 1 (c). The texture model is fed with the texture code $c_t \in \mathbb{R}^{128}$ and the concatenation of a 3D point $\mathbf{x} \in \mathbb{R}^3$ and a normalized time value $T \in \mathbb{R}^1$, and predicts the RGB values $\mathbf{c} \in \mathbb{R}^3$ for the given point. There are five residual blocks

Boyan Jiang, Xinlin Ren and Xiangyang Xue are with School of Computer Science, Fudan University. Yanwei Fu is with School of Data Science, Fudan University.

† Corresponding authors. E-mail: Yanwei Fu (yanweifu@fudan.edu.cn).

in the texture model network, each of which consists of two fully connected layers with a skip connection from the input to the second layer. The input of each block is summed up with the feature encoded from the texture code $c_t \in \mathbb{R}^{128}$.

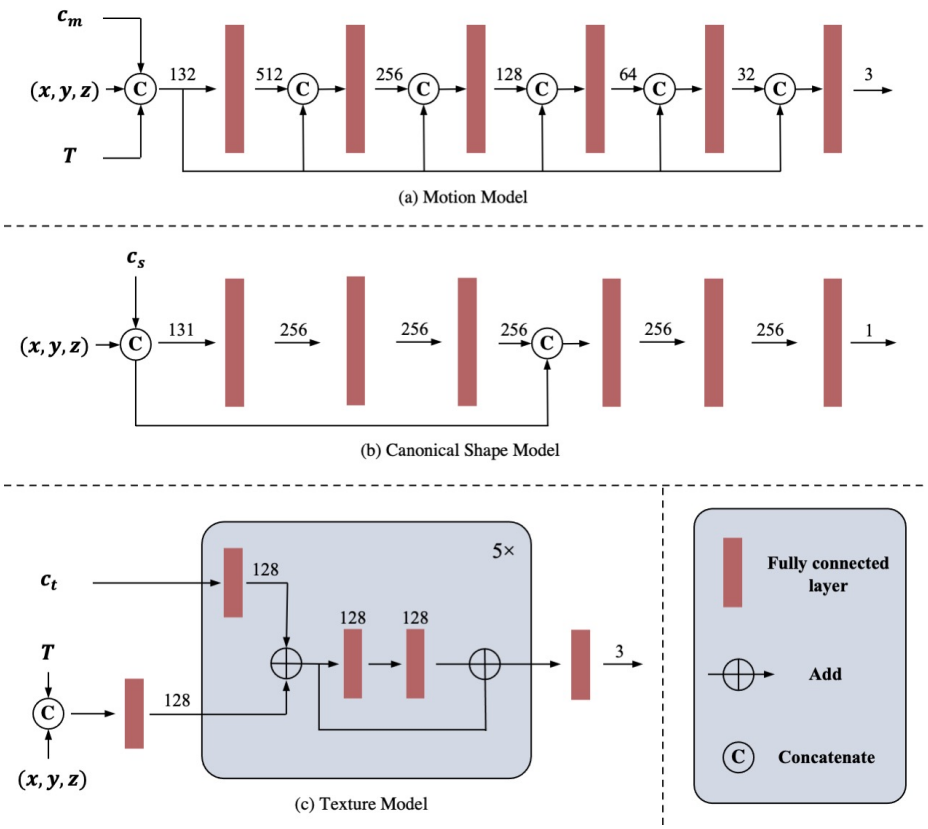

Fig. 1. Detailed network architectures in our framework.

1.2 Local Part Coordinates

In this section, we introduce how to select part centers on SMPL meshes and define the local coordinate system for each part.

Part center sampling Start from a template mesh of SMPL topology in the rest-pose, we first uniformly sample a point set P ($|P| = 100K$) on its surface, and then perform the following steps recurrently to maintain a set of the selected part centers C: 1) randomly choose a point $\mathbf{x}$ from P; 2) remove all the points whose Euclidean distance from $\mathbf{x}$ is less than or equal to $0.5\mathbf{r}$ from P, where $\mathbf{r} = 5cm$ is the part radius we pre-defined; 3) add $\mathbf{x}$ to C. The loop terminates when $|P| = 0$, and we finally obtain 2127 part centers.

Local coordinate frame Given a point $\mathbf{c}_k$ on a triangle face as the part center, we use the face normal as the up-axis $\vec{a}_{k_1}$, the direction vector from point c_k to a vertex of the triangle as another axis $\vec{a}_{k_2}$, and finally the last axis $\vec{a}_{k_3} = \vec{a}_{k_1} \times \vec{a}_{k_2}$. The rotation matrix of part k is then define as $\mathbf{R}_k = [\vec{a}_{k_1}, \vec{a}_{k_2}, \vec{a}_{k_3}]$. Now, a 3D point P_{glo} in the world coordinate frame can be transformed to P^k_{loc} in the local coordinate frame of part k with $P^k_{loc} = \mathbf{R}^T_k (P_{glo} - \mathbf{c}_k)$.

1.3 Other Details

Point sampling The query points used for training and test-time optimization come from three sources: 1) surface; 2) near surface space; 3) free space in the bounding box. During training, we sample $M = 10000$ surface points on the ground truth mesh, while at the test time, the points are randomly chosen from the input point cloud. Given the on-surface points, we obtain the near surface points by adding a displacement vector sampled from a Gaussian distribution $N(0, 0.01)$ to each on-surface point. And $M/8$ free space points are uniformly sampled within the human bounding box. We compute the initial bounding box with the inner body mesh, and pad $10cm$ on each axis as the sampling region.
Auto-decoding During the test time, we use the trained model to fit complete or partial point clouds via the auto-decoding manner [13] (main paper Sec. 3.4). Specifically, we fix the parameters of the local implicit network and optimize the latent codes with back-propagation by minimizing the objective function introduced in Sec. 3.3 of the main paper. We initialize the latent codes for each local part with the random vectors sampled from a Gaussian distribution $N(0, 0.01)$ and use the Adam optimizer [6] with learning rate $1e^{-3}$ to perform backward optimization for 3000 iterations. We use the same objective function and loss weights as training (Sec. 3.3 of the main paper). The optimization process takes around $15min$ for each sequence of $L = 17$ frames on a single GeForce RTX 2080Ti GPU card.
Mesh postprocessing As mentioned in the main paper (Sec. 3.4), the original extracted mesh of our method contains interior back-faces and some tiny floating components in the outside. This possibly because that for the off-surface points, the Eikonal term [2] only constrains the $L2$-norm of their gradients rather than specific SDF values, which may confuse the prediction of gradient direction on the points that far from the part centers. We remove these artifacts by using the post-processing algorithm introduced in LIG [4]. Specifically, we first compute the centroid and surface normal for each face of the original mesh reconstructed by the network, and find its k nearest points on the input point cloud to calculate the mean normal consistency as the normal alignment score. Then a Laplacian kernel is used to smooth the normal alignment score, all faces with the score below a certain normal alignment threshold n and disconnected components with area below a are discarded. We used the same postprocessing parameters as LIG, except that we only preserve the most biggest connected component rather than use the area threshold a as we focus on reconstructing single object.
Mesh surface extraction Different from LIG [4], which only evaluates the occupied grid cells and assumes all empty ones to be "exterior" space to extract

the isosurface, we construct a SDF volume and evaluate every grid point with our local implicit function, ensuring to reconstruct the outer surface not covered by any local parts defined on the inner body mesh. Similar to the strategy illustrated in the main paper Sec. 3.3, we evaluate a given point with n parts that covered it ($n = 32$ in the surface extraction process), and get the final prediction via average-pooling. And if not covered by any parts, n nearest parts are used. To make the inference more efficient, instead of cubic volume used in previous work [9,13], we resort to a rectangular volume that defined as the bounding box of inner body mesh with $10cm$ padding on each axis. It takes around $15s$ to extract a mesh of resolution 256.

2 Additional Experimental Results

2.1 Non-Rigid Fusion

Real-world performance To further demonstrate the value of our method in the actual application scenario, we perform extended evaluation on the real data. Specifically, we capture a human motion depth sequence with a static Azure Kinect sensor, and use our model pre-trained on 100 sequences to conduct non-rigid reconstruction based on the point cloud from raw depth images (note that the background is filtered out). The qualitative results are shown in Fig. 2, DynamicFusion produces fine-grained surface details on cloth but contains noisy (body edge) and incomplete (face, arms and legs) areas. In contrast, our model recovers smoother and more complete geometry with plausible temporal deformation thanks to the local 4D formulation. We use part radius $r = 10cm$ for more stable results. Note that the experiments in here and Sec. 4.3 of the main paper, we only reconstruct the partial surface observed by the static camera in the time span, which cannot be quantitatively evaluated, thus we only show the qualitative comparisons.

Comparisons with DoubleFusion Qualitative results are shown in Fig. 3. We add the same SMPL body mesh used in LoRD into DoubleFusion as body prior, which generally makes the fusion reliable, but it still struggles with aligning finer motion. Also, there are some missing parts, e.g. arm or leg, caused by fitting error. In contrast, LoRD employs a temporal model to provide global geometry consistency between frames, which is more robust to fitting error.

Moving monocular camera In addition to the static camera, we also conduct non-rigid depth fusion in the setting of a moving monocular camera, that is, a person performing some motions during a time span with a depth camera rotating around him concurrently. In this case, the observation of each time step is still partial, but almost every body part can be observed during the time span. Our goal is to compensate each frame based on the geometry information observed in other frames. Specifically, given a dynamic mesh sequence of $L = 17$ frames, we make the camera rotates around the up-axis (Y-axis in our setup) and render a 512×512 depth image every $360/17$ degrees. In this experiment, we assume the camera poses are known, and use the same 10 novel sequences chosen in the generalizable 4D reconstruction task for evaluation. We choose PTF [15],

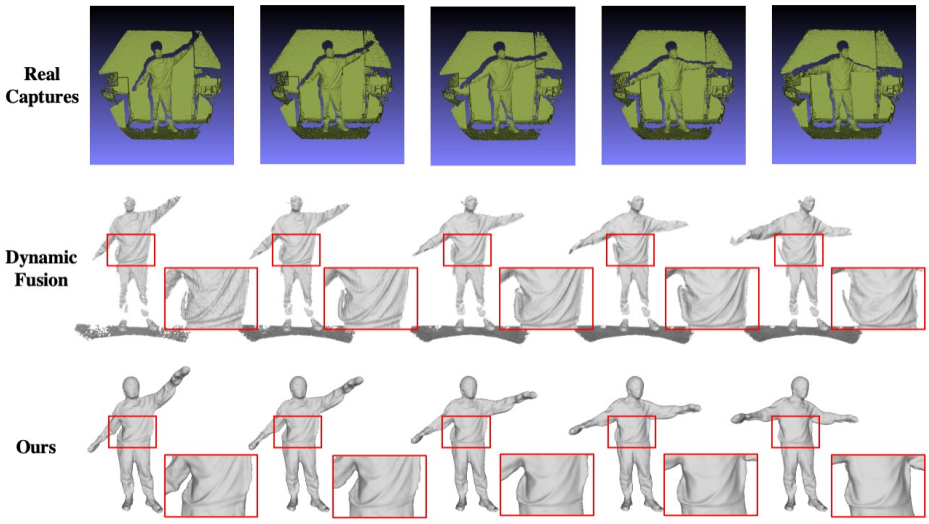

Fig. 2. Monocular depth fusion on the real-world depth captures. The depth images are captured with a static Azure Kinect sensor. The first row shows the point cloud from raw depth captures. Our method is capable to recover the plausible geometry with detailed surface deformation, e.g. clothing wrinkles in the zoomed in part, in such challenging scenario.

NPMs [12] and CAPE [7] as our baseline. For PTF, we input the partial point cloud to the pretrained single-view model, and obtain the reconstructions with feed-forward fashoin. And for NPMs and CAPE, we use the auto-decoding manner to optimize the latent codes in their formulation based on each frame partial observation. Note that we provide the ground truth SMPL pose to CAPE, and only optimize the cloth latent code. All these methods are working in the frame-wise manner to produce the sequence results. The quantitative results in Tab. 1 show that our LoRD representation outperforms all the baseline methods by a large margin. The qualitative comparisons are shown in Fig. 8. Thanks to the local 4D representation, the proposed method achieves high-quality completion results, and produces the detailed geometry and plausible temporal deformation in the invisible areas, while PTF and NPMs fail to hallucinate accurate geometry from partial observations as they do not utilize the temporal information, and CAPE cannot model the high-fidelity surface details.

2.2 Interpolation of Latent Codes

Like many other local implicit representations, our model does not have a compact global latent space to sample from, so the representation ability is often measured by the capability of fitting observations, validated in Sec 4.1 of main paper. Nevertheless, we can still interpolate between two human sequences by interpolating representation between the corresponding local parts (as shown

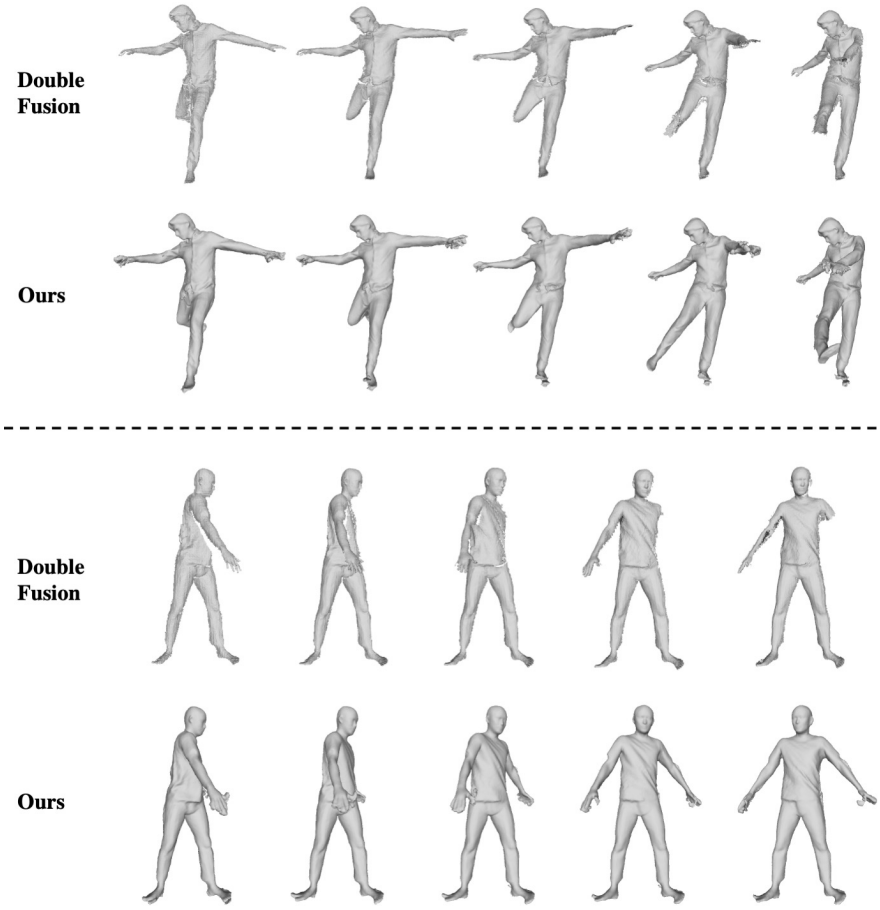

Fig. 3. Qualitative comparisons with DoubleFusion. We choose two examples and uniformly sample 5 out of 17 frames for visualization.

Table 1. Quantitative comparisons on monocular depth fusion with a moving camera. Our method outperforms all the baseline methods by a large margin.

	Ch.-L_2 ↓	Normal ↓	F-Score ↑
PTF [15]	37.936	0.810	0.145
NPMs [12]	0.981	1.933	0.429
CAPE [7]	1.010	0.356	0.377
Ours	**0.395**	**0.226**	**0.750**

in Fig. 4). We first linearly interpolating c_s and SMPL poses showing smooth change between two subjects. Then we interpolate c_m, c_s and per frame SMPL pose jointly to show the representation ability of motion space, since the local deformation is related to global motion. Note that the texture model is optimized per sequence without continuous latent space.

2.3 Ablation Study

Effect of sequence length L The key novel capability of our model is to represent the temporal deformation of 3D shape, so that setting $L = 1$ makes this infeasible, and our model degenerates to framewise method LIG [4] and thus achieves similar performance. Additionally, we test different L on instance-level reconstruction from sparse points task and show the results in Tab. 2. Note that longer sequence demands stronger network capacity while more geometry information can be exchanged between frames through our motion model. The results show that LoRD is able to work with different sequence lengths. We choose $L = 17$ following 4D-CR [3] during training to learn our 4D representation. For longer sequence during test-time, we can utilize the sliding window strategy similar to HMMR [5] to recurrently recover the whole sequence.

Table 2. Evaluations of LoRD on different sequence lengths.

L	Ch.-$L2$ ↓	Normal ↓	F-Score ↑
5	0.175	0.160	0.914
10	0.207	0.176	0.880
17	0.192	0.158	0.945
20	0.159	0.173	0.875
30	0.389	0.221	0.718

Generalization Besides the results shown in Sec. 4.2 of the main paper, we also choose 10 sequences from our training set, and get the performance (Ch.-L_2=0.317, Normal=0.170, F-Score=0.926) on 4D reconstruction task. The results are in general comparable with the performance on testing set (last row of Fig. 6 (a, II) in the main paper), which reflects that thanks to the local part formulation, our model has strong generalization ability with the prior of local surface deformation learned from the training sequences. We further verify this by training our model on even 1 motion sequence of 17 frames, which also can produce the high-quality reconstructions on novel sequences. We show the qualitative results in Sec. 4.2 of Supp. Mat.

Comparison with only test-time training In Fig. 5, we show the loss curves of optimization w/ and w/o pretraining. It can be seen that the pretraining helps optimization converge faster and more stable. By taking about 50 mins per sequence, optimizing from scratch obtains slightly better performance than optimizing with pretraining.

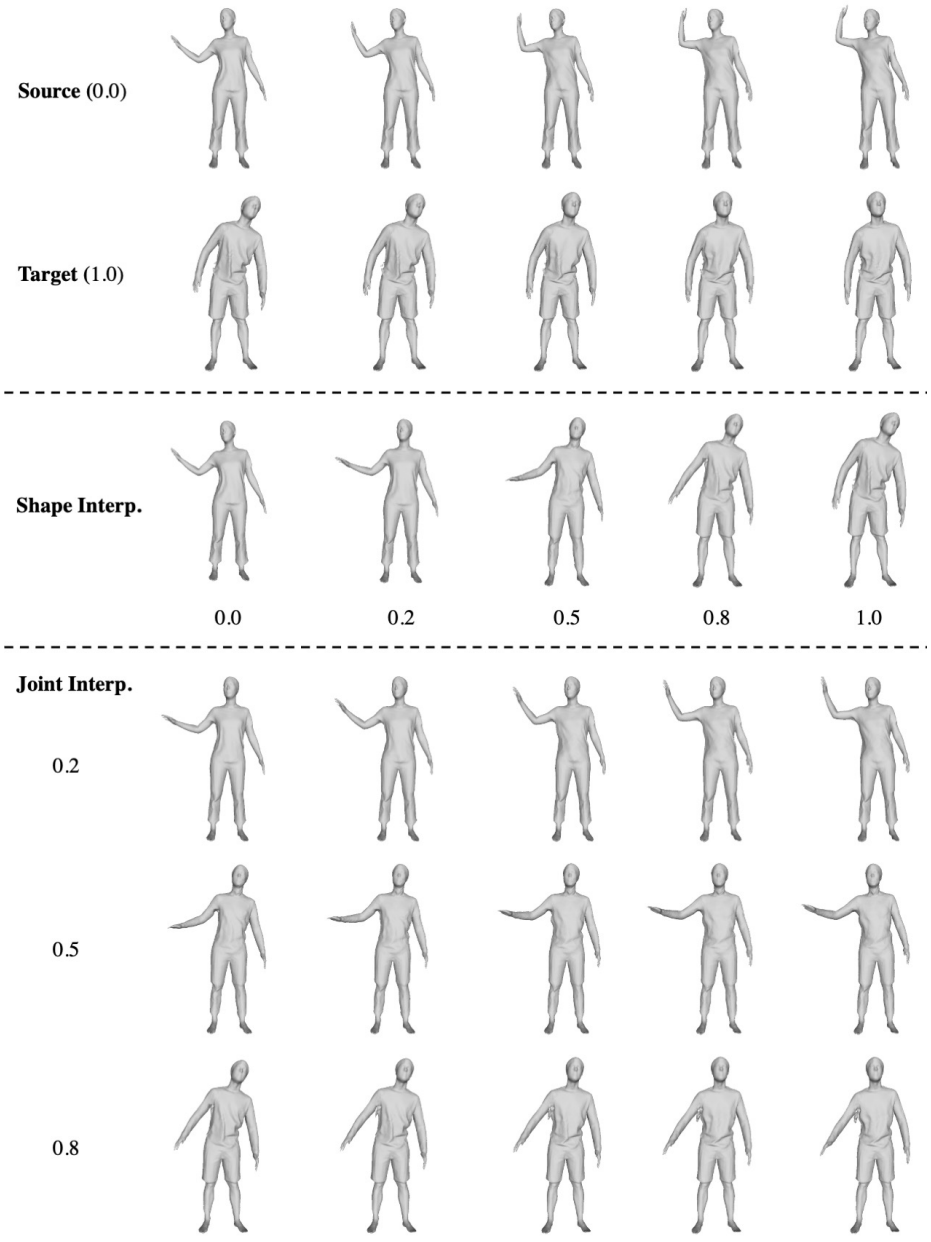

Fig. 4. We choose two sequences (Source and Target) to perform interpolation of latent codes. "Shape Interp." means we interpolate latent shape codes and SMPL poses of the first frames to show the evolution of shape. "Joint Interp." indicates we interpolate shape code, motion code and per frame pose jointly. The decimal values denote the interpolation coefficients

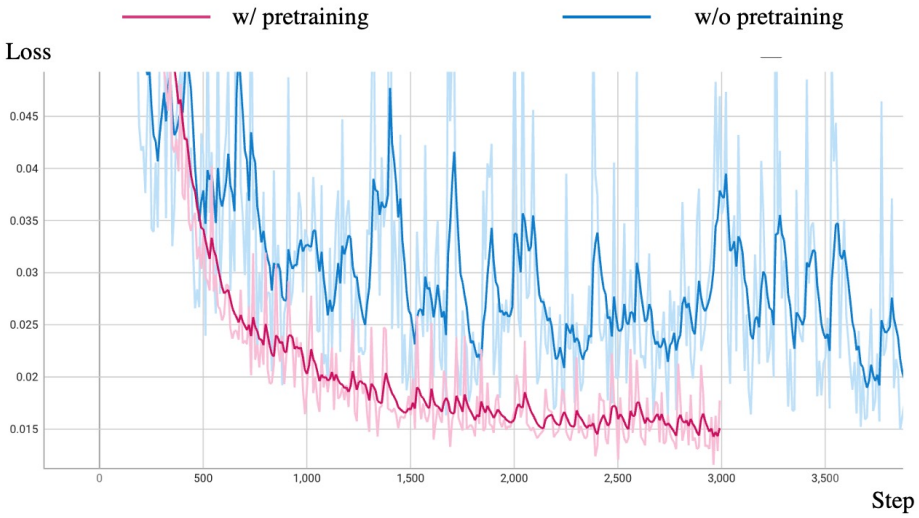

Fig. 5. Loss curves during optimization w/ and w/o pretraining.

3 Local Surface Visualization

Our LoRD representation aims to use a local part-level network to model the detailed temporal deformation of surface patches. To show this, we visualize the temporal deformation of a local patch in Fig. 6. Specifically, we choose a point cloud sequence of 17 frames, each of which has $10K$ points, and perform auto-decoding to optimize the latent codes of each local part. After that, we select a part and extract the surface patch with the local implicit network (main paper Sec. 3.2) conditioned on its latent codes. We show the temporal deformation of this local patch under the global (above) and local (below) coordinate frames respectively. It can be seen that the temporal changing within the local part is smooth and coherent, and our method successfully models the detailed deformation, e.g. changing of the clothing wrinkle.

4 Qualitative Results

4.1 4D Reconstruction

Shape quality Fig. 9, 10 and 12 are the extended figures of Fig. 5 in the main paper, which show more qualitative comparisons with the SoTA methods on 4D reconstruction from sparse points. And Fig. 13 shows some additional reconstruction results of our model trained on 100 sequences.
Textured results We show more textured results in Fig. 7. We choose raw scan mesh sequences from CAPE dataset (containing holes and noises) and use our pretrained LoRD model in this experiment. The results above are obtained with

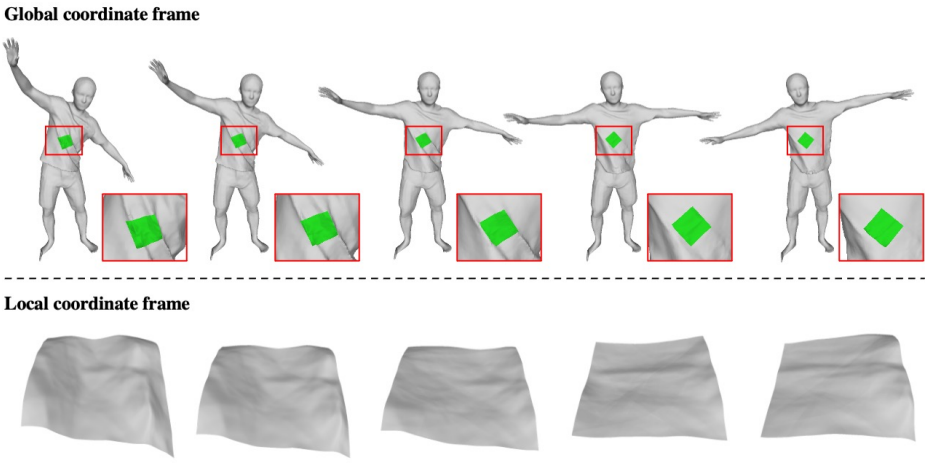

Fig. 6. We select a local part and visualize the surface patch within it. The results above show the temporal deformation of this patch (green) under the global coordinate frame, while below under the local coordinate frame.

colored sparse point clouds sampled from the scan meshes as input. And the results below are obtained from the rendered RGB-D sequences.

4.2 Ablation study

Imperfect body tracking Fig. 11 (before refining) shows some challenging cases that LoRD produces artifacts with inaccurate body tracking. We provide the reconstructions after refining that demonstrate the effectiveness of our inner body refining method. For each example, we stack the ground truth clothed mesh (green) together with the initialized inner body mesh (gray) on the left to reflect the inaccurate estimation, and show the reconstructions on the right. The results show our refining process successfully corrects the noisy inner body, which facilitates the reconstruction.

Local part size Fig. 14 shows the reconstruction results of different part radii. We can observe that the part size slightly affects the quality of reconstructions, the over-small ($\mathbf{r} = 3cm$) part produces some artifacts around body and hands and the reconstructed surface is under-smooth, whereas the larger parts ($\mathbf{r} = 8cm\&\mathbf{r} = 10cm$) tend to recover overly smooth results for some frequency details such as clothing wrinkles and fingers.

Generalization As mentioned in the main paper Sec. 3.3, the training of our LoRD representation is very data-efficient. We show the results of our model trained on 100 sequences (Fig. 5, 6 (b) in the main paper and Fig. 12, 13 in the Supp. Mat). Additionally, we train our model on one motion sequence of length $L = 17$ from the training set, then test on the novel sequences, and show the qualitative results in Fig. 15. As shown, our model trained on even one sequence

still gains generalization ability and produces the high-fidelity reconstructions, which demonstrates the generalization power of our local 4D representation.

5 Limitations and Future work

The proposed LoRD representation shows the powerful capability and achieves the state-of-the-art performance on various tasks. Now we discuss a few limitations of our method which also points to the future directions.

First, we now rely on the SMPL body model to temporally track local parts, though existing methods can produce accurate body in many cases, it is still challenging to work in the complex real scenario. Extending our representation to cooperate with more general tracking methods such as scene flow, deformation graph, would make our method stronger and capable of modeling non-human objects, e.g. animals.

Second, the current experiments mainly focus on 4D reconstruction from 2.5/3D data, e.g. sparse point clouds, RGB-D. Combining the recent neural rendering techniques [10,14,16] with our LoRD representation to support 4D reconstruction from pure RGB videos would be a promising future direction.

Third, we currently use a unified part radius in our formulation. However, different body parts contain various levels of detail, which may be suitable to model by different sizes of parts. Defining the part radii according to the body part label would be one solution.

We believe that the proposed representation could potentially be a building block for various applications, e.g. Metaverse, Robotics, animation, and provides some insights for future research directions.

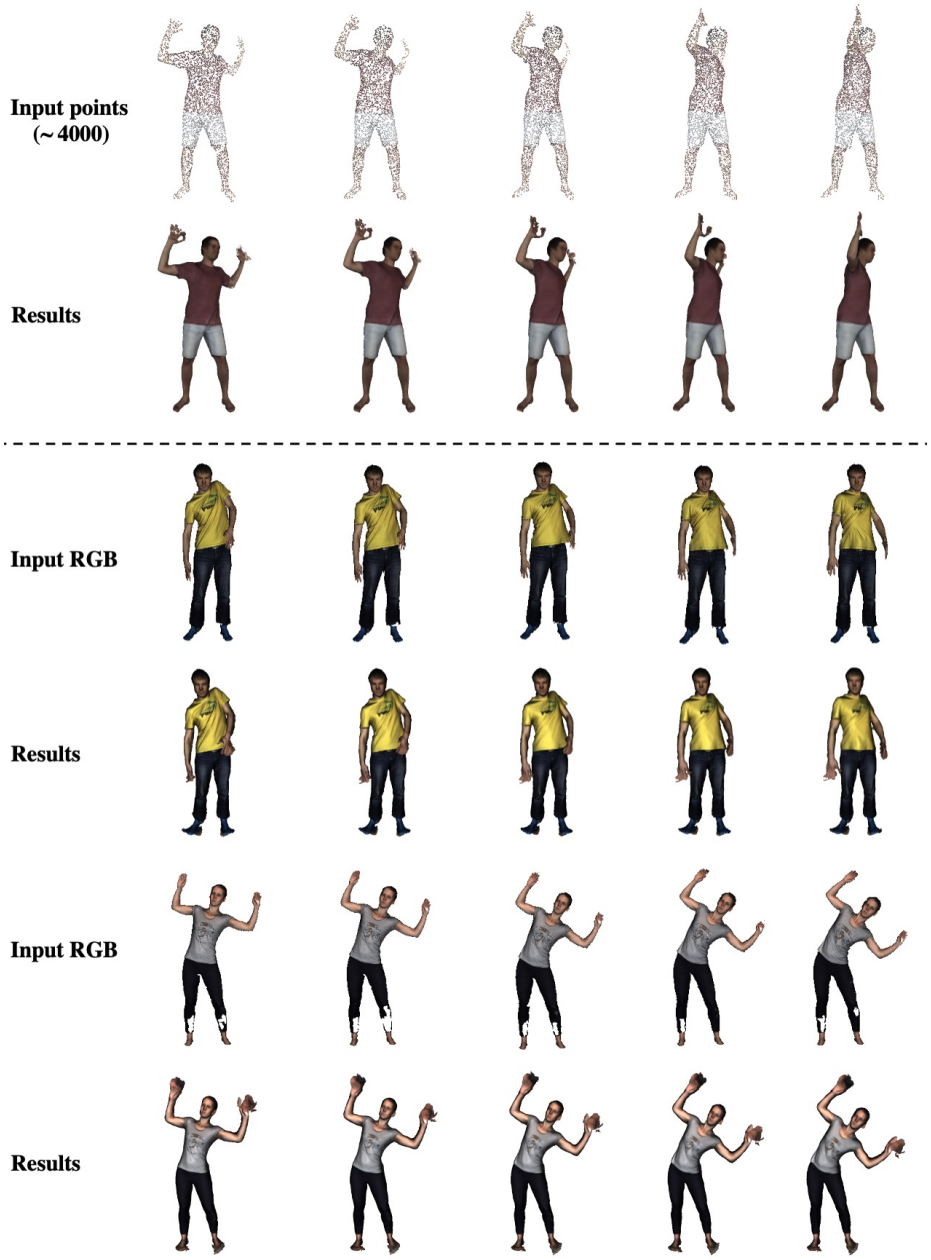

Fig. 7. More textured results achieved by our method. Note that the results below are obtained from RGB-D inputs, we only show color images for visualization.

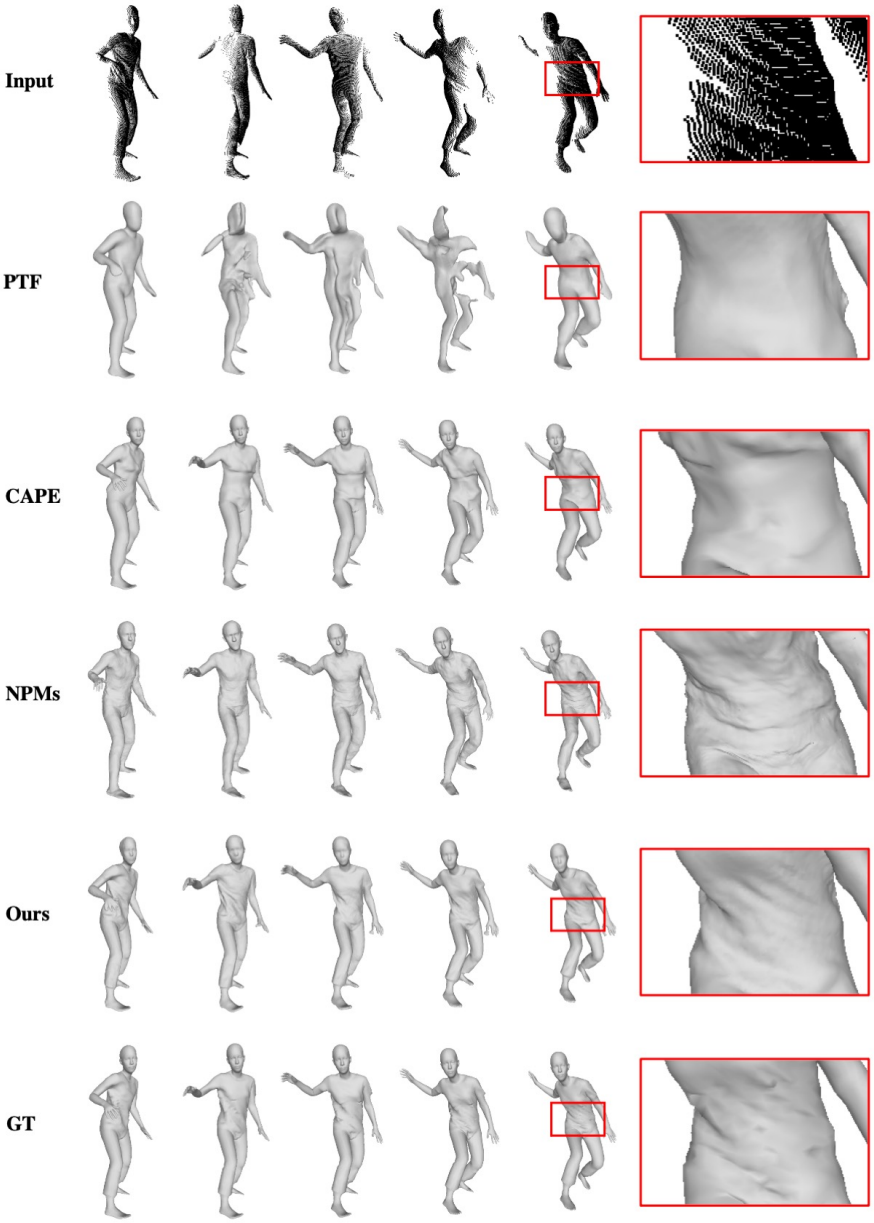

Fig. 8. Qualitative results on monocular depth fusion with a moving camera. The reconstructed sequences have $L = 17$ frames, and we uniformly choose 5 frame for visualization. We assume a moving camera that rotates around the performer, and render a depth image every 360/17 degrees. The partial point clouds are obtained by back-projecting the depth images with the camera intrinsics, and rotate according to the known camera extrinsics.

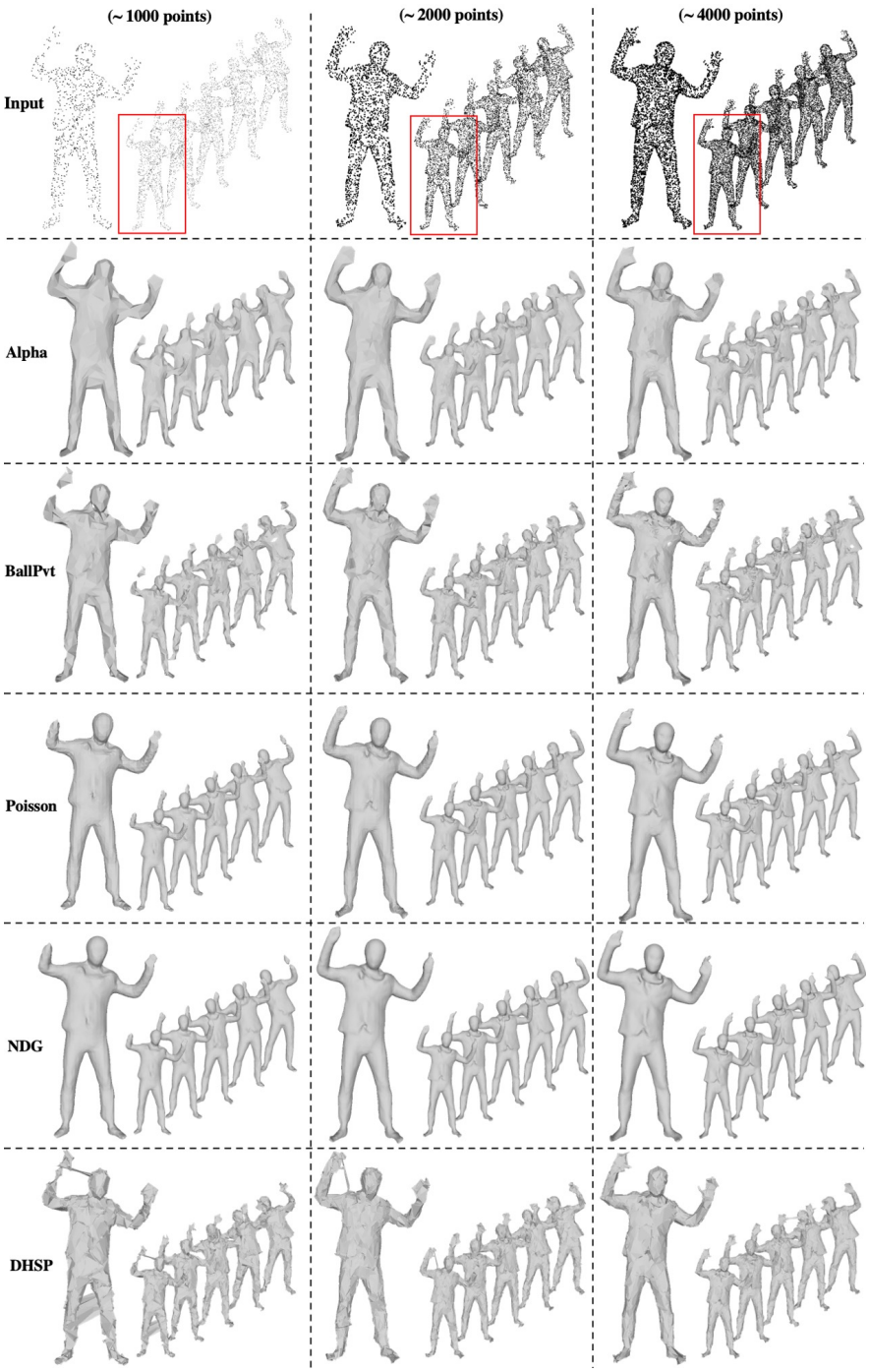

Fig. 9. 4D reconstruction from sparse points (instance-level) (1).

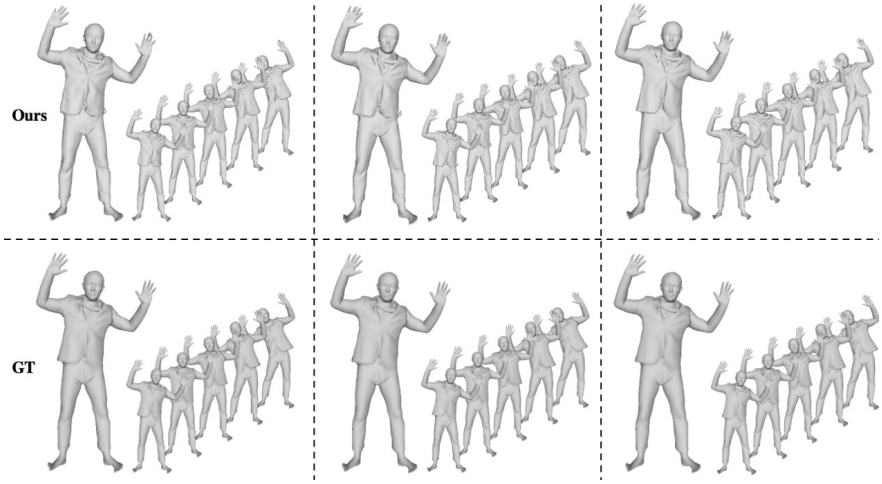

Fig. 10. 4D reconstruction from sparse points (instance-level) (2). The reconstructed sequences have $L = 17$ frames, and we uniformly choose 5 frame for visualization. We zoom in the first frame to show the surface details clearer.

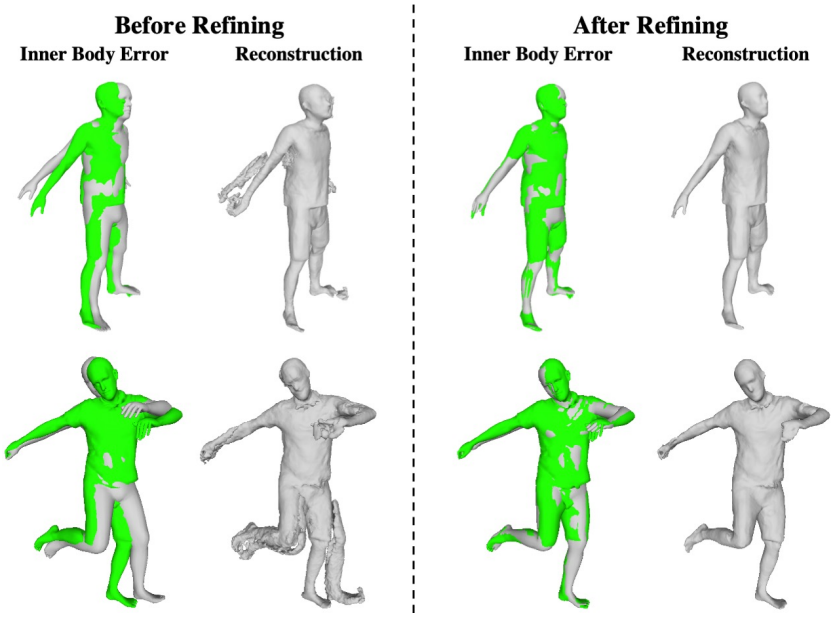

Fig. 11. Effectiveness of the inner body refining. We show the inner body and the reconstructed meshes before (left) and after (right) our inner body refining process. Note that we stack the ground truth clothed mesh (green) with the inner body estimation (gray) to show the inaccurate parts.

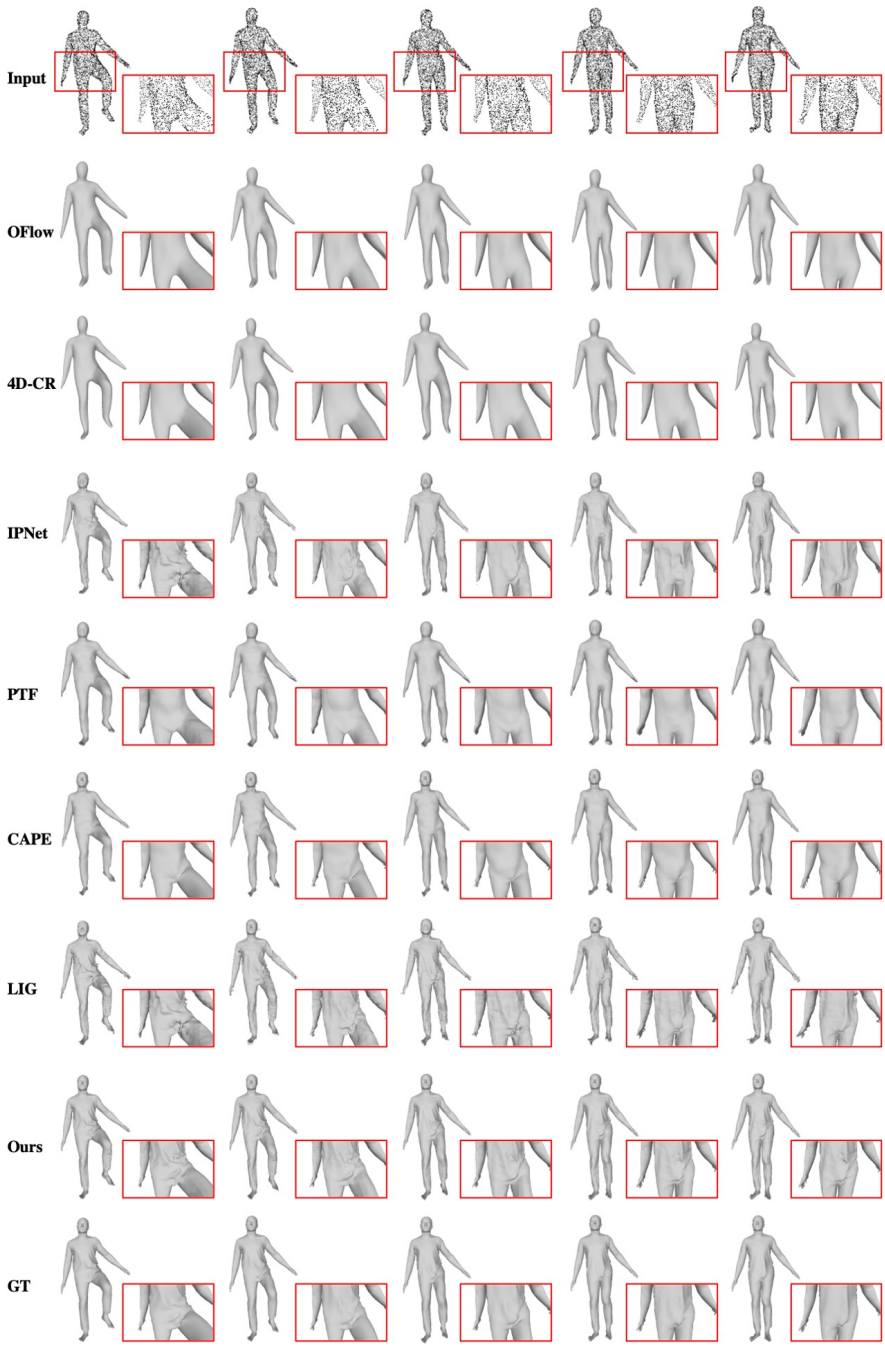

Fig. 12. 4D reconstruction from sparse points (generalization). The reconstructed sequences have $L = 17$ frames, and we uniformly choose 5 frame for visualization.

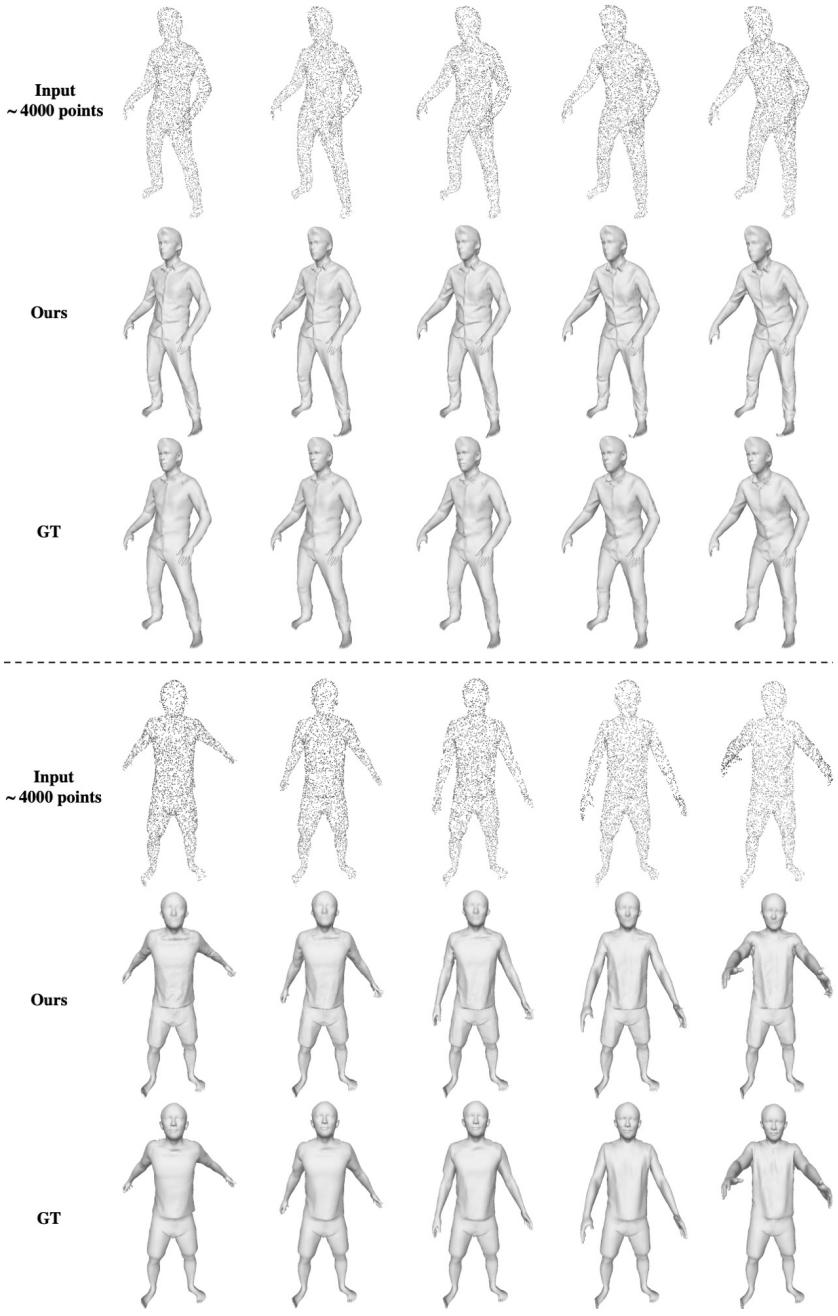

Fig. 13. 4D reconstruction from sparse points (generalization). Here we show more qualitative results achieved by our model trained on 100 sequences. The reconstructed sequences have $L = 17$ frames, and we uniformly choose 5 frame for visualization.

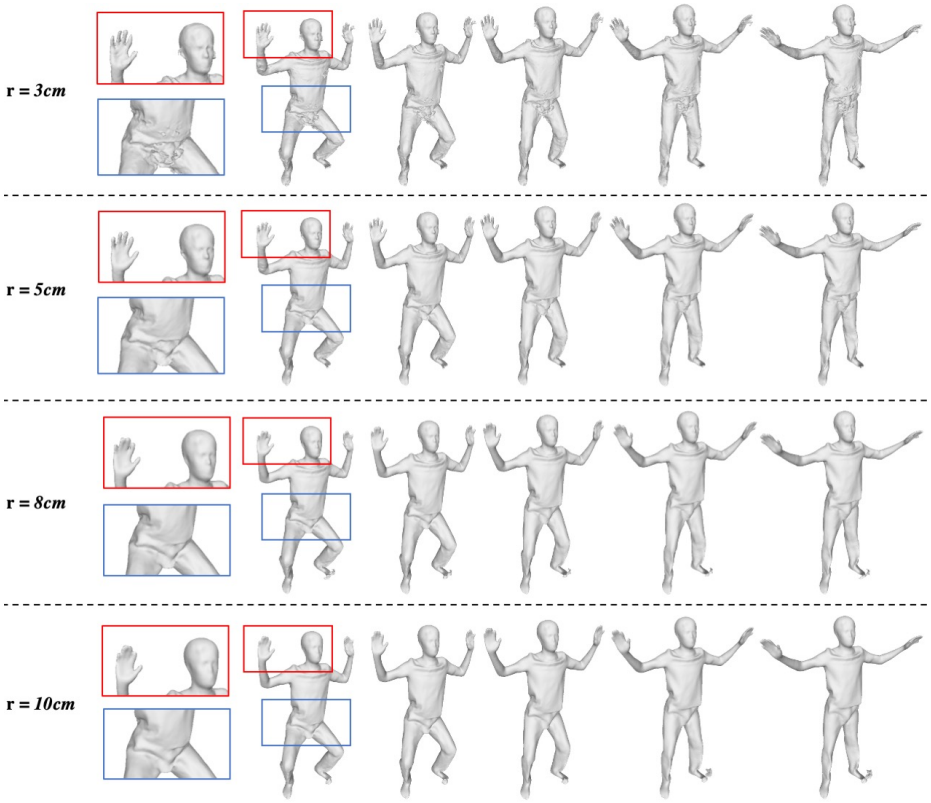

Fig. 14. Effect of the part radius. Different sizes of local parts affect the reconstruction performance but not very heavily. We choose part radius $\mathbf{r} = 5cm$ in our experiments as it in general produces better results.

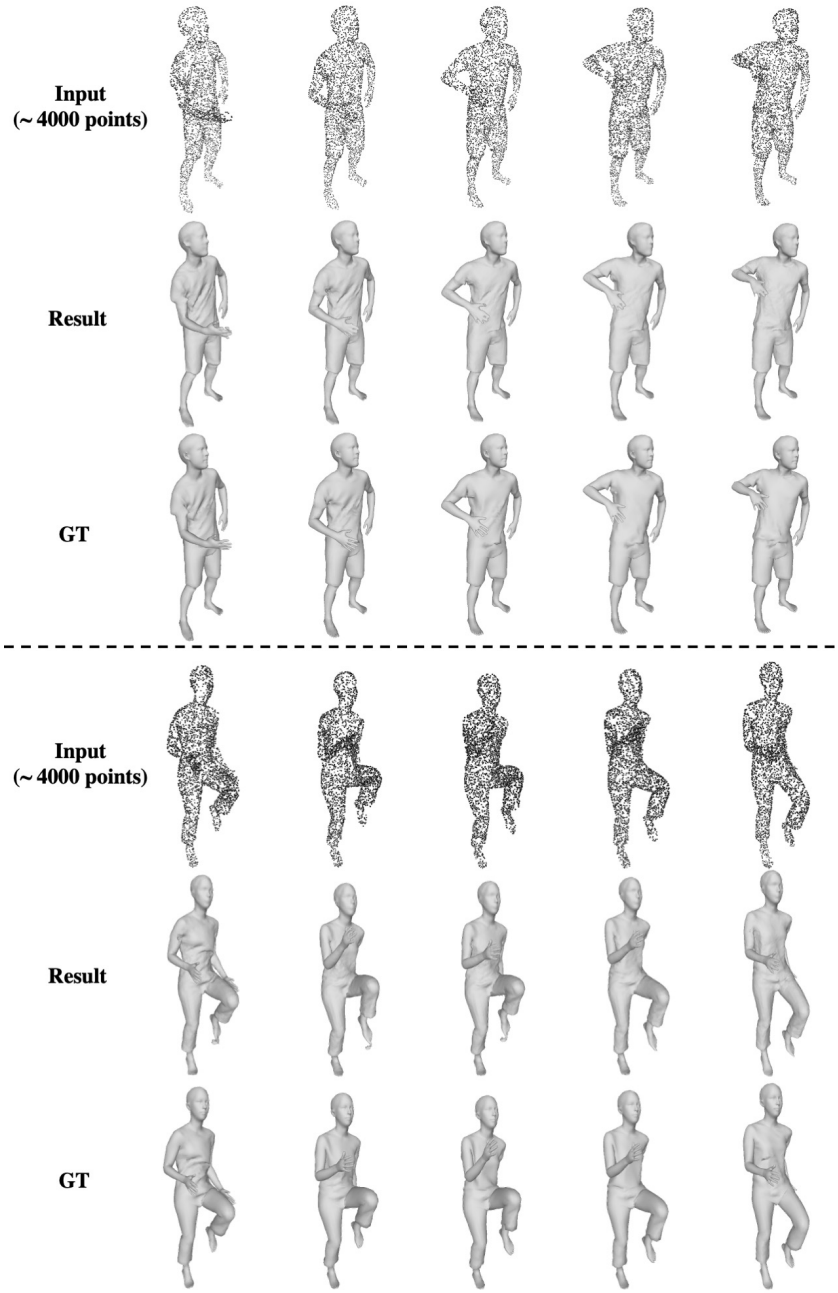

Fig. 15. Qualitative results from our model trained on only 1 motion sequence of length $L = 17$, which show that our LoRD representation can learn local deformation prior from very few data and generalize to novel sequences with high-quality geometry and temporal deformation.

References

1. Chen, Z., Zhang, H.: Learning implicit fields for generative shape modeling. In: CVPR. pp. 5939–5948 (2019)
2. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. arXiv preprint arXiv:2002.10099 (2020)
3. Jiang, B., Zhang, Y., Wei, X., Xue, X., Fu, Y.: Learning compositional representation for 4d captures with neural ode. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5340–5350 (2021)
4. Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T.: Local implicit grid representations for 3d scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6001–6010 (2020)
5. Kanazawa, A., Zhang, J.Y., Felsen, P., Malik, J.: Learning 3d human dynamics from video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5614–5623 (2019)
6. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
7. Ma, Q., Yang, J., Ranjan, A., Pujades, S., Pons-Moll, G., Tang, S., Black, M.J.: Learning to Dress 3D People in Generative Clothing. In: Computer Vision and Pattern Recognition (CVPR) (2020)
8. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3. Citeseer (2013)
9. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4460–4470 (2019)
10. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020)
11. Oechsle, M., Mescheder, L., Niemeyer, M., Strauss, T., Geiger, A.: Texture fields: Learning texture representations in function space. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4531–4540 (2019)
12. Palafox, P., Božič, A., Thies, J., Nießner, M., Dai, A.: Npms: Neural parametric models for 3d deformable shapes. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 12695–12705 (2021)
13. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 165–174 (2019)
14. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021)
15. Wang, S., Geiger, A., Tang, S.: Locally aware piecewise transformation fields for 3d human mesh registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7639–7648 (2021)
16. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems **34** (2021)