4D-DRESS: A 4D Dataset of Real-World Human Clothing With Semantic Annotations

Supplementary Material

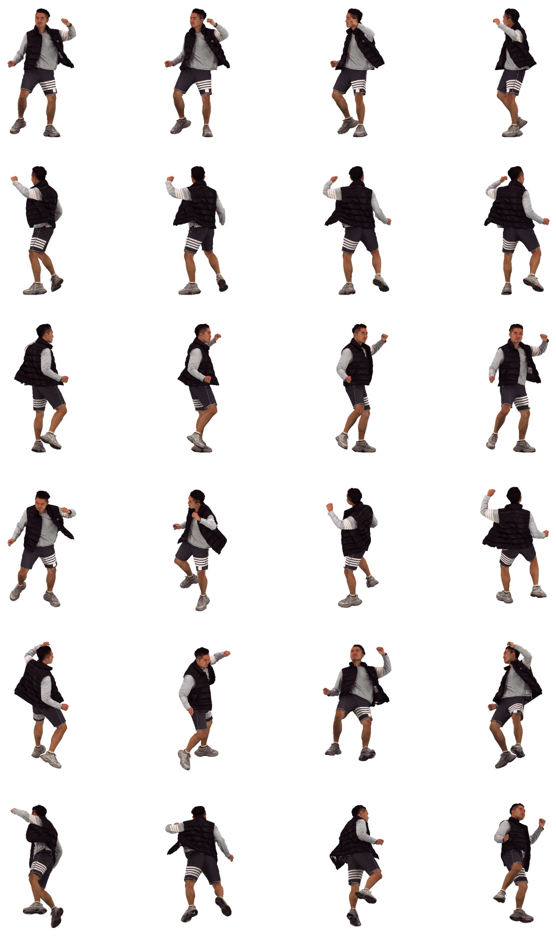
Figure 1. **Example of 24 rendered views.** We render 24 views to ensure the visibility of each scan vertex and consider the computational cost of human parsing.

1. Implementation Details

1.1. Multi-view Parsing

Multi-view rendering. For each frame $k \in \{1, ..., N_{frame}\}$, we render twelve horizontal, six upper, and six lower images $I_{img,n,k}$ that are uniformly distributed on a sphere by rasterizing the textured scan with Pytorch3D [13], where $n \in \{1, ..., N_{view} = 24\}$. Each scan is centralized according to its bounding box center and then placed at the camera sphere center. The rendered images have a resolution of 512×512. Examples of 24-view rendered images are shown in Fig. 1.

4D-DRESS	Graphonomy (LIP)
(-1) other	background
(0) skin	torso-skin, face, glove left-arm, right-arm, left-leg, right-leg
(1) hair	hat, hair, sunglasses
(2) shoe	socks, left-shoe, right-shoe
(3) upper	upper-clothing, dress, scarf
(4) lower	pant, skirt
(5) outer	coat

Table 1. **Label mapping between 4D-DRESS and LIP dataset**. We define 6 label categories based on LIP dataset.

Human image parser (*PAR*). We apply the pre-trained Graphonomy [7] to each rendered image $I_{img,n,k}$ and save the label results as a new image $I_{par,n,k}$. Concretely, we manually classify the 20 classes of Graphonomy labels into 6 classes that are used in our dataset: skin (0), hair(1), shoes(2), upper(3), lower(4), and outer(5) clothes. The corresponding labels between Graphonomy (LIP) and ours are shown in Tab. 1. Specifically, we map the background label from Graphonomy to our setting with a label value -1, and the color code of white. These background labels will return 0 in the vote function $f_{par,n}(p, l)$.

Optical flow transfer (*OPT*). To establish connections with previous frames, we project previous frame vertex labels to multi-view labels $I_{lab,n,k-1}$ using the same rendering cameras and rasterizer from Pytorch3D. Then, we warp these previous multi-view labels to the current frame $I_{opt,n,k}$ using the optical flow vectors predicted by the RAFT [16] model. The vertex labels at the first frame do not involve this process thanks to our first-frame initialization (see Sec. 1.3). Concretely, each pixel label with location p within $I_{lab,n,k-1}$ will be warped to a new pixel location $p + v$ at the current frame, through the optical flow vector $v = RAFT(I_{img,n,k-1}, I_{img,n,k}, p)$. The new labels at the current frame are determined by voting. If there is no corresponding label found in the previous frame, the new label will be set to -1.

Segmentation masks and scores (*SAM*). We use Segment Anything Model [9] to segment each rendered image $I_{img,n}$ into a group of masks $M_{m,n}$ without any ex-

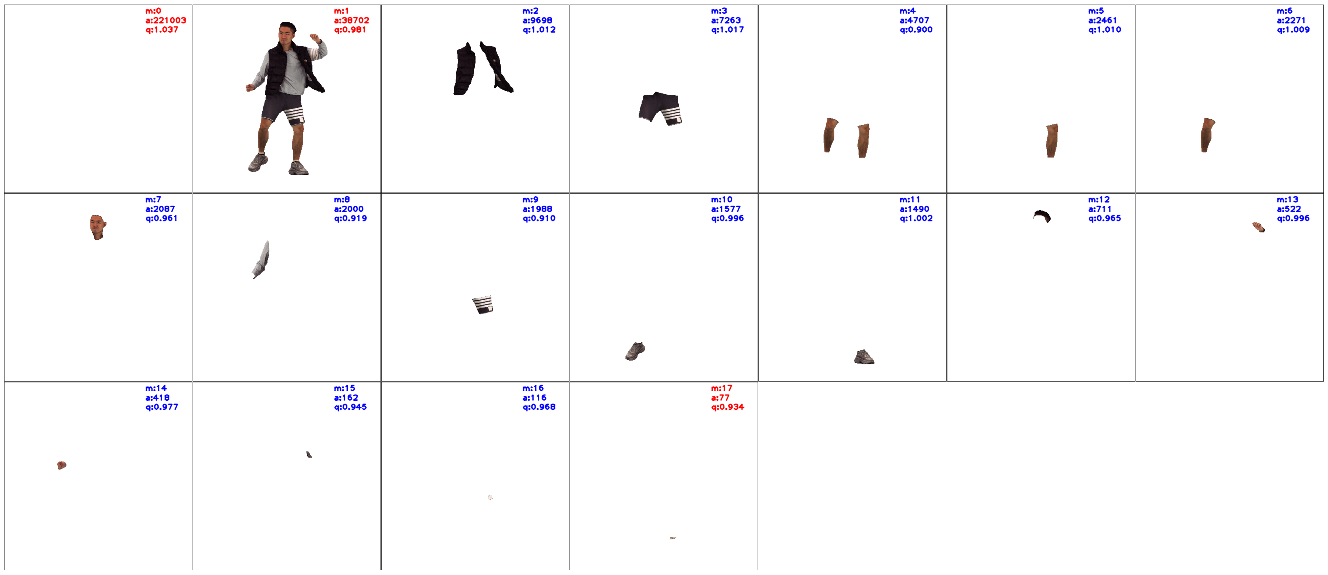

Figure 2. **Example of SAM predictions.** The input image is the first view (upper-left) of Fig. 1. We filter out the segmentation masks that contain background, full body, and only small regions (marked as red).

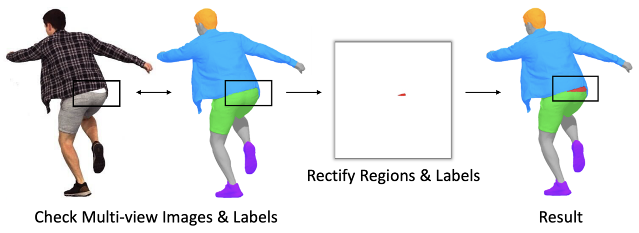

Figure 3. **Example of manual rectification.** An annotator selects a region in the rendered images and gives a correct label. The label is projected to 3D and used for correcting the 3D vertices through a second round of graph cut optimization.

tra prompts, where $m \in \{1, ..., M_{mask,n}\}$. Then we compute the score function $S(l, M_{m,n})$ within each mask for each label by fusing the votes from the image parser and optical flow, normalized by the area of the mask. Fig. 2 depicts the predicted segmentation masks from a rendered image. A pixel p within the rendered image $I_{img,n}$ may belong to multiple segmentation masks. In this case, the SAM vote function $f_{sam,n}(p, l)$ is calculated by summing all the scores of masks that contain this pixel.

1.2. Graph Cut Optimization

The energy Eq. (5) in the main paper is optimized through the graph cut algorithm (alpha-expansion). The vertex-wise unary energy is normalized among all labels and then added to the edge-wise binary energy. The weights are empirically set as $\lambda_p = 0.5$, $\lambda_o = 0.5$, $\lambda_{po} = 1.5$, $\lambda_s = 1$, and $\lambda_b = 1$.

1.3. Manual Rectification Process

Manual rectification on segmentation masks. In our dataset, each scan mesh has around 80k vertices. Manually annotating their vertex labels on the 3D scans is very expensive and time-consuming. Thus, we introduce a manual rectification process within the 2D image space. After the first graph cut optimization, we render vertex labels to multi-view images, from which we let an annotator correct labels with the segmentation masks and a painting tool. More specifically, the annotator is asked to identify an incorrectly labeled region by checking the multi-view images and labels. Once an incorrect labeling is found, the annotator will look for its corresponding segmentation masks for label correction. If such a mask does not exist, the annotator will manually paint the region using a painting tool. Finally, the images with rectified labels are projected to 3D vertices and are formulated as the manual vote function $f_{man,n}(p, l)$. The energy $E_{man,n}$ term will be added to the second round of graph cut optimization, with a large weight $w_{man} = 10$. We note that for each 150-frame 4D sequence, the rectification process takes about 30 minutes on a desktop with an RTX 2080Ti GPU whereas the human parsing and the graph cut optimization take two and one hour, respectively. An example of our rectification process is shown in Fig. 3.

First-frame initialization of vertex labels. To ensure a good label initialization, the motion sequences always start from the A pose, which is easier for human parsing and pose registration. We obtain the first-frame vertex labels

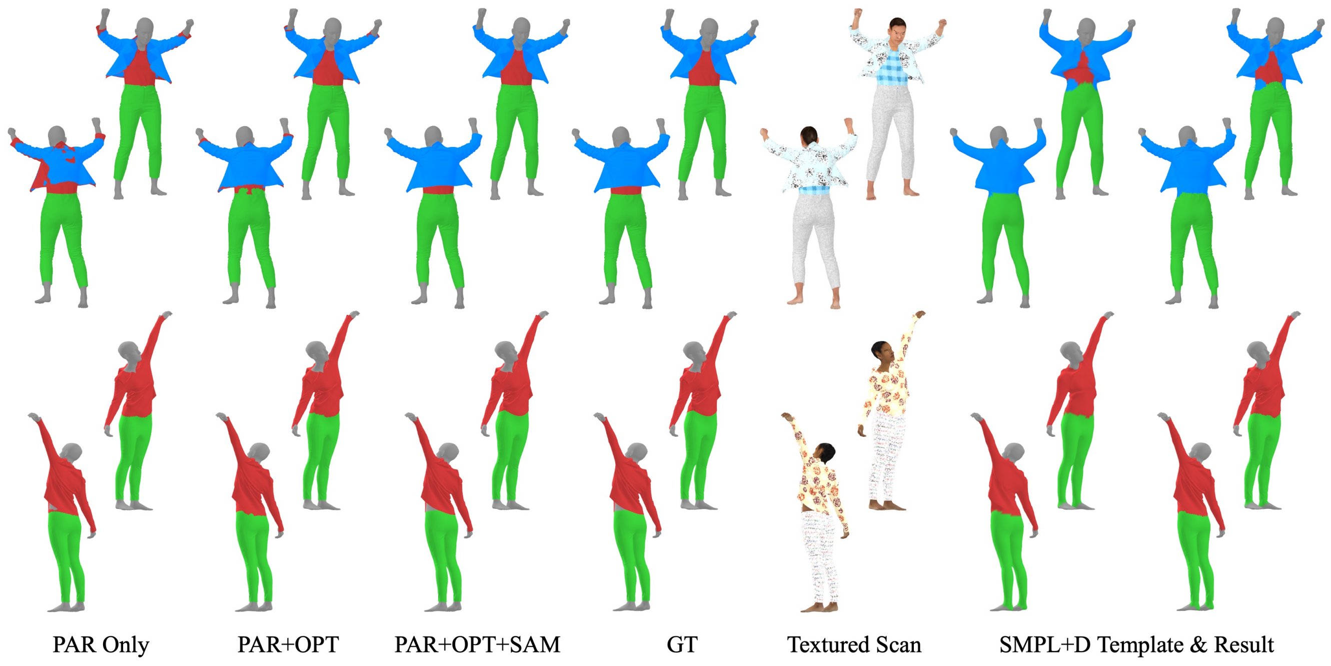

Figure 4. **Ablation study and baseline comparison on the BEDLAM dataset.** We conducted ablative experiments on the synthetic BEDLAM dataset where ground-truth semantic labels are available.

using the edge-wise binary energy and the multi-view unary energy calculated only from the image parser (E_{par}) and manual rectifications (E_{man}).

2. Additional Parsing Experiments

2.1. 4D Parsing on Synthetic Datasets

We conducted controlled 4D parsing experiments on two synthetic datasets, CLOTH4D [17] and BEDLAM [3], where the cloth meshes are simulated from cloth templates on top of the parameterized body models. Since within these synthetic datasets, some inner body and cloth vertices are always invisible from the outside, we report our labeling accuracy only on the vertices that are visible from our 24 views of rendered images.

Baseline comparison. We first compare our 4D human parsing method with a template-based baseline method [12] that utilizes a semantic SMPL+D template to first track the clothed human shape, and then project the template labels to neighboring scan vertices. Since ClothCap [12] didn't release their 4D parsing code, we implemented their parsing method following their descriptions. We first register the SMPL+D model to all frames. Then we initialize the first frame template label using the nearby scan vertex labels obtained through our first-frame initialization process. At each frame, we update the template labels using the body prior, previous frame prior, and the Gaussian Mixture Model trained from the vertex colors of each labeled category. Finally, the scan vertex labels are assigned from the nearest template label. The quantitative parsing results from this baseline method are shown in the main paper. Here, we show more qualitative results in Fig. 4.

The main issue of this template-based baseline method is fitting the SMPL+D template to loose human outfits. The spatial mismatch between template and loose garments generates incorrect labels, especially in the open area of the jackets. Besides this, precisely updating the template labels using the Gaussian mixture model of labeled vertex colors is also difficult, especially in front of garments that have similar colors. The limited template resolution also results in noisy boundary labels at the higher-resolution clothed human meshes. The parsing accuracy from this baseline method is below 90% for all synthetic outfits.

Ablation studies. We then compare our 4D human parsing method (without manual rectifications) with several ablations of the multi-view parsing inputs (PAR Only, PAR+OPT, PAR+OPT+SAM), as shown in Fig. 4. Similar to Fig. 3 in the main paper, we observed similar qualitative results on the synthetic datasets.

2.2. 4D Parsing on Other Datasets

Our 4D human parsing method takes the input as scan mesh sequences and multi-view videos and thus can be applied to the existing real-world 4D human datasets, such as BUFF,

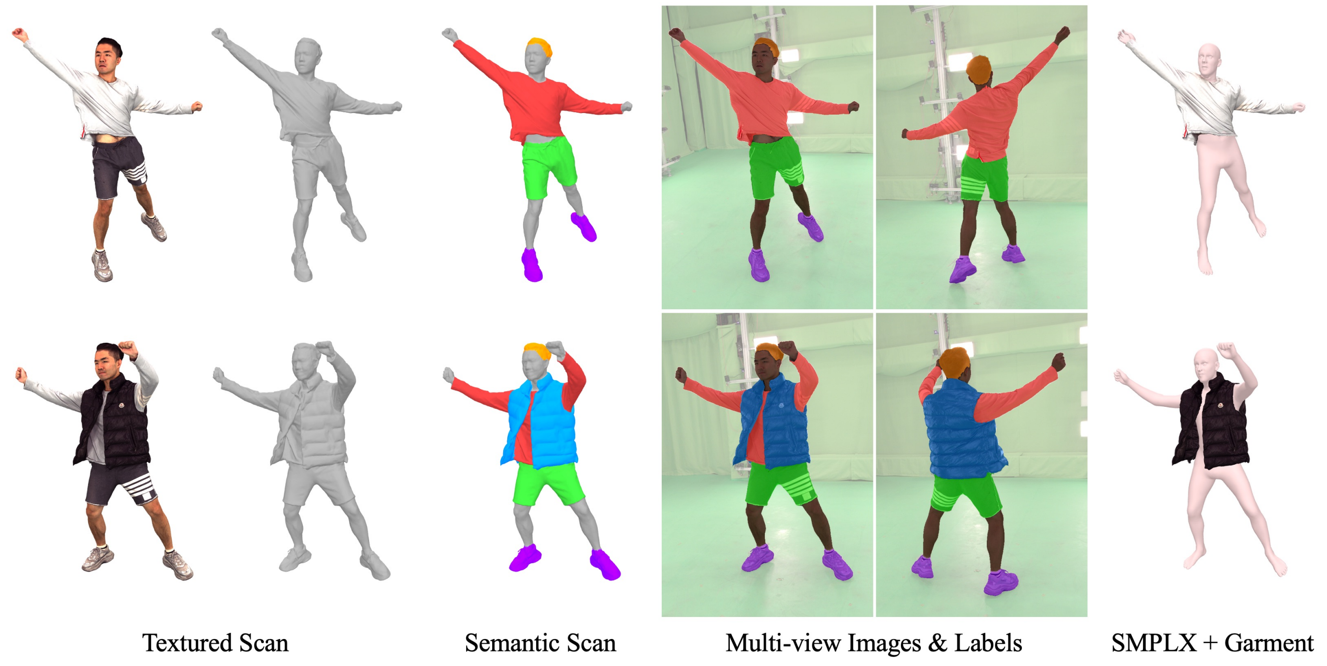

Figure 5. **Data provided in the 4D-DRESS dataset.** We provide high-quality 4D textured scans. For each scan, we annotate vertex-level semantic labels, thereby obtaining the corresponding garment meshes and fitted SMPL(-X) body meshes.

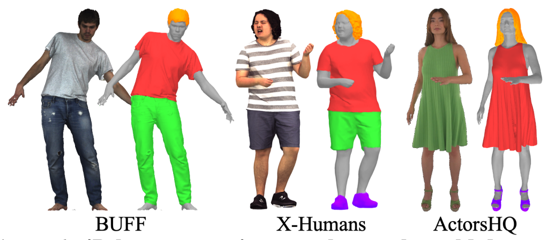

Figure 6. **4D human parsing on other real-world datasets.**

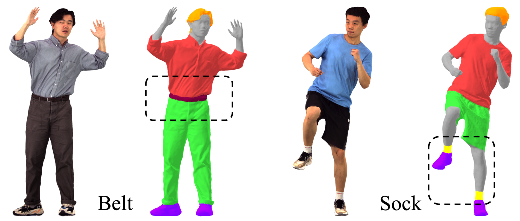

Figure 7. **4D human parsing with new labels.**

X-Humans, and ActorsHQ, as shown in Fig. 6.

2.3. 4D Parsing with New Labels

The six classes in our 4D-DRESS are strategically defined to ensure a consistent benchmark evaluation for clothing simulation and reconstruction. We showcase the generalization ability of our parsing method with new labels in Fig. 7, by effectively distinguishing a belt from pants and socks from shoes. Initiated during the first-frame initialization, these new labels can integrate into the 4D parsing pipeline. However, refining labels for these smaller clothes and objects may entail additional manual efforts for rectification.

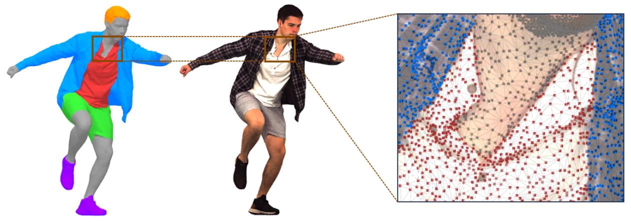

Figure 8. **Vertex-level semantic annotations.** Our dataset contained precise vertex-level semantic labels of clothing categories.

3. Additional Dataset Description

3.1. Data Capturing Steup

We captured our dataset with a volumetric capture system [5] equipped with 106 synchronized cameras (53 RGB and 53 IR cameras). The sequences are filmed at 12 MP, 30 FPS, and within an effective capture volume of 2.8 m in diameter and 3 m in height. Each frame consists of a mesh with 80k faces and a texture map.

3.2. Dataset Contents

Our 4D-DRESS dataset provides the following data, examples are shown in Fig. 5:

- **4D textures scans.** High-quality 4D textured scans of 32 subjects, 64 human outfits (32 Inner and 32 Outer), with 520 motion sequences and 78k frames in total.
- **Vertex-level annotations.** We offer accurate vertex-level annotations through our 4D human parsing pipeline. An

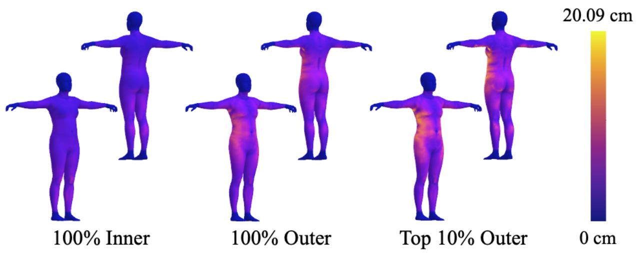

Figure 9. **Visualization of 4D-DRESS outfits distance**. The mean distance distribution from garment outfits to SMPL bodies.

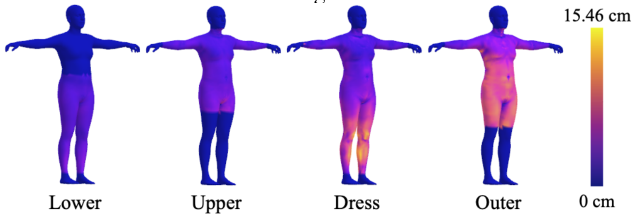

Figure 10. **Visualization of 4D-DRESS outfits distance**. The mean distance distribution from garment meshes to SMPL bodies.

example of our label quality is shown in Fig. 8. Using these labels, we also provide multi-view images with semantic labels in 2D.

- **Parametric body models.** We register precise SMPL and SMPL-X body models for each frame.
- **Garment meshes**. We extract 3D garment meshes based on the vertex labels.

3.3. Clothing Distribution

We compute the mean distances from the outfits to the registered SMPL body surfaces. The inner and outer outfits exhibit distance ranges of up to 7.12 cm and 14.76 cm, respectively, over all frames. The distribution of the distance on the SMPL body is shown in Fig. 9. In the 10% most challenging frames that have a larger Chamfer distance between scan mesh and SMPL mesh, the distance range increases to 20.09 cm for outer outfits. We further visualize the mean distances of each garment category, as shown in Fig. 10. The average Chamfer distance between the clothed human scans and SMPL body meshes are 3.30 cm and 5.28 cm for the inner and outer outfits in our 4D-DRESS dataset, and 2.21 cm in the X-Humans dataset [15].

4. Additional Evaluation Benchmarks

4.1. Clothing Simulation

4D-Dress provides diverse garments and challenging human pose sequences, which serves as a great asset for future research in clothing simulation. Unlike the synthesized garment templates with smooth surfaces and simple topologies, we provide templates extracted from scans, with realistic wrinkles and complex structures. Using these templates, we evaluated the performance of recent unsupervised cloth simulators, including PBNS [1], Neural Cloth Simulator (NCS) [2] and HOOD [8], and a baseline method, linear blend-skinning. We quantitatively and qualitatively compared the generated garments with our scanned garments. We also demonstrated the potential of HOOD by simply optimizing the material parameters, which again confirmed the value of our dataset. In the following sections, we elaborate on each step of our experiments.

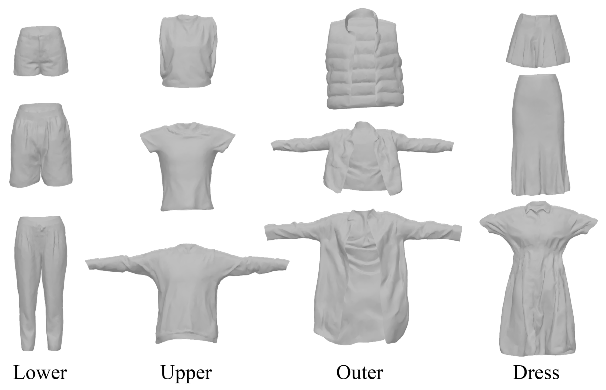

Figure 11. **Garment templates used for clothing simulation.** We extract four types of Garment templates from T-pose scans.

4.2. Template Extraction

Current clothing simulation algorithms rely on a predefined garment template, deforming it to generate realistic simulations under various poses. They typically utilized synthesized garment templates, with unnaturally smooth surfaces and basic topologies. In our work, we provided templates directly extracted from real-world scans, offering a more realistic foundation for deformation.

Firstly, we select from pose sequences the frames closest to the canonical pose, in other words, "T-pose". We also make sure that the body in this frame is static and garments are in rest status. Then we apply inverse LBS to convert the scans into exact canonical pose. After extracting garment meshes from the unposed scans, we made some manual efforts to recover the garment shape in Blender [6]. Specifically, we erased unwanted faces, solved penetrations between clothing and body, and smoothed rigid wrinkles and coarse boundaries. Synthesized templates used by current simulators usually have 4-5k vertices. We observed in experiments that too many vertices in the template are computationally expensive for simulation and may erode performance. Therefore, we downsampled each template to 30-50%, which now has 3-8k vertices in total depending on each garment's surface area, while keeping them in their original shapes. To use lower garments in simulators, like pants and lower skirts, pinned vertices are compulsory for them to stay on the body. We extract the loop around the waist as pinned vertices and provide their indexes.

4.3. Evaluation Details

In the clothing simulation benchmark, we compared four different clothing simulators: LBS, PBNS [1], NCS [2], and HOOD [8]. The training and evaluation of each method were conducted using the SMPLX model, which provides more details in visualization. The final evaluation is done on four types of garments(Upper, Outer, Dress, and Lower), with each having 2 garments and 6 sequences in total. For qualitative evaluation, we employed Chamfer distance and stretching energy, scaling vertex positions by a factor of 100 to use centimeters as the unit.

The Chamfer distance, shown in equation 1, is computed by summing the squared distances between nearest neighbor correspondences of two-point clouds. We denote the sampled points on simulation and ground-truth meshes as X and Y, respectively, with N_* representing the amount of sampled points, set to 100,000 in our experiment.

$$d_{CD} = \frac{1}{N_x}\sum_{x\in X}\underset{y\in Y}{min}\|x-y\|_2^2 + \frac{1}{N_y}\sum_{y\in Y}\underset{x\in X}{min}\|x-y\|_2^2 \quad (1)$$

The stretching energy, widely used in mass-spring-based simulators, is computed as equation 2, where N_e is the total number of edges, e_i and $\bar{e}_i$ are the lengths of the edge i in the current frame and the template respectively.

$$E_{str} = \frac{1}{N_e}\sum_i \|e_i - \bar{e}_i\|^2 \quad (2)$$

We provide more details on implementing each method: **LBS** blends joint transforms with skinning-weights. For each garment template, we find the nearest body node on the canonical SMPLX human, and get the skinning weights on this point. Then, we follow the same forward LBS process in SMPLX to get deformed template meshes.

PBNS and **NCS**, both are deformation-based methods, predict vertex-wise deformation on the template and employ LBS to transform the deformed garment into desired poses. Given their "One model for one garment" nature, we trained each garment from scratch. We also used identical AMASS sequences mentioned in the NCS paper to ensure fairness. As both PBNS and NCS developed using SMPL, we made slight adjustments to the data-loading pipeline to ensure their compatibility with SMPLX. And we assigned zero poses to joints that are exclusive in SMPLX.

Meanwhile, we also kept the same training settings used in their original papers. For PBNS, default parameters were used, and each garment underwent training for 20-50 epochs to ensure convergence. For NCS, a batch size of 2048 was employed across all training instances, as suggested in their paper. In the case of tight garments, default parameters were maintained with a temporal window size of 0.5 and 10 iterations for blend weights smoothing. In the case of loose garments like outerwear and dresses, we made slight parameter adjustments for stable training, typically using a temporal window size of 0.75 and 1, with 50 iterations for blend weights smoothing, as suggested by the author in a GitHub issue.

HOOD, as a simulation-based method, predicts physically realistic fabric dynamics and is agnostic to garment topology. Hence, we directly used a pre-trained publicly available model to evaluate our garments. Unlike the deformation-based methods, which convert the template in canonical pose to any pose instantly, HOOD predicts garment motion frame by frame. Therefore, to apply our canonical template for simulating each sequence, we have to convert the template into the pose of the first frame. In the HOOD paper, they used LBS to convert templates, which works for tight synthesized garments. However, for our real-world garments, it usually results in large stretching on mesh, especially around joint areas. Therefore, alternatively, we insert extra frames from the canonical pose to the first frame and simulate the prolonged sequence to get a natural transform from the canonical pose. The first poses for all sequences in our dataset are in A-pose. Generally, we insert 30 frames to transfer from canonical to A-pose, which makes it slow enough for the garment to stay in rest status with minimum dynamics.

4.4. HOOD*: Material Optimization

HOOD provides 4 local material parameters for each vertex, including μ and λ evaluating the ability of stretching and area preservation, mass m computed from the fabric density, and the bending coefficient $k_{bending}$ penalizing folding and wrinkles. For each edge, there are three material parameters, including μ, λ, and $k_{bending}$. Assuming we have v vertices, e edges, and coarse edges in total, we define the material parameters as $\mathcal{M} \in \mathbb{R}^{4v+3e}$.

In the fine-tuning process, we freeze the pre-trained HOOD model $\mathcal{H}$ and only update material parameters $\mathcal{M}$. Using all 6 sequences of each garment for training, we feed them into model f to get simulated outputs. Then, with Ground Truth garment mesh G, we compute Chamfer distance and stretching energy, as described in equation 3.

$$\mathcal{L} = \mathcal{L}_{CD}(f(\mathcal{M},\mathcal{H}),G) + w\mathcal{L}_{Estr}(f(\mathcal{M},\mathcal{H}),G) \quad (3)$$

We used the stretching energy from HOOD and set w as 1 in our experiments. Chamfer distance $\mathcal{L}_{CD}$ is described in equation 4, measuring the average distance between simulation and ground-truth garment. We use V_*, $(* \in [s,g])$ to represent the simulated and ground truth vertices and use N_* as the total number of vertices.

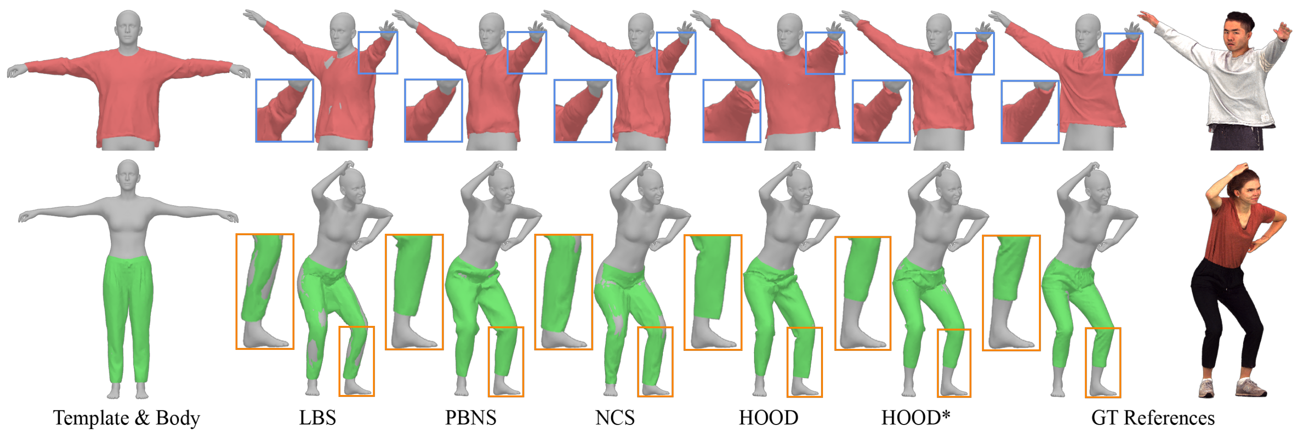

Figure 12. **Additional qualitative results for clothing simulation.** Left are templates used for simulations. Right are simulations and ground-truth scans. HOOD presents more dynamic while getting overly stretched. HOOD* matches well with ground truth.

	Inner		Outer	
Method	mAcc.↑	mIoU↑	mAcc.↑	mIoU↑
SCHP [10]	0.908	0.832	0.863	0.768
CDGNet [11]	0.922	0.853	0.887	0.790
Graphonomy [7]	0.968	0.859	0.915	0.810

Table 2. **Image-based human parsing**. Results of image-based human parsers on 4D-DRESS.

$$\mathcal{L}_{CD} = \frac{1}{N_s}\sum_{x\in V_s}\underset{y\in V_g}{min}\|x-y\|^2 + \frac{1}{N_g}\sum_{y\in V_g}\underset{x\in V_s}{min}\|x-y\|^2 \tag{4}$$

For each garment, we trained with Adam Optimizer with a learning rate of 5e-4. And it usually takes 50 epochs to converge. Generally, HOOD* gets a much lower distance compared to ground truth mesh quantitatively, and also performs more natural fabric dynamics qualitatively.

4.5. Clothed Human Parsing

We design a benchmark for the image-based human parser. Concretely, we project each scan frame's vertex labels to the multi-view captured images using corresponding camera parameters and rasterizer, which provide the ground-truth pixel labels for evaluating the image-based human parsing methods: SCHP [10], CDGNet [11], and Graphonomy [7]. In Tab. 2, we report the mean Pixel Accuracy (**mAcc**.) and mean Intersection over Union (**mIOU**) between the prediction and the ground-truth labels. We conducted our human image parsing experiments on one subset of our 4D-DRESS dataset, which contains 128 sequences of 64 outfits (2 sequences for each of the inner and outer outfits). The qualitative parsing results are shown in Fig. 13.

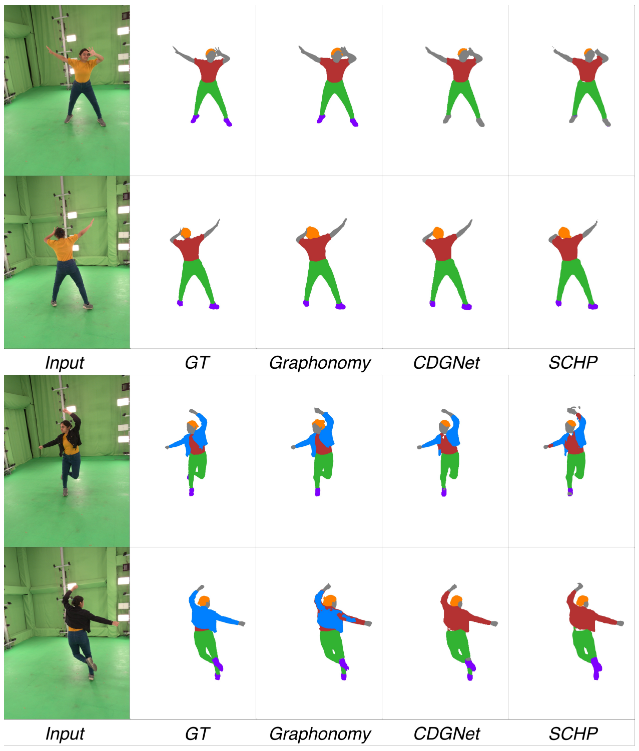

Figure 13. **Human parsing comparison.** We use the ground-truth semantic labels to evaluate state-of-the-art human parsing models. These methods generally failed to predict correct clothing labels from different view angles.

4.6. Human Representation Learning

We design a new benchmark for evaluating the human representation learning task. Unlike physics-based methods, this line of work directly takes 3D human scans as training input and obtains an animation-ready human avatar. We follow the split strategy mentioned before and evaluate prior

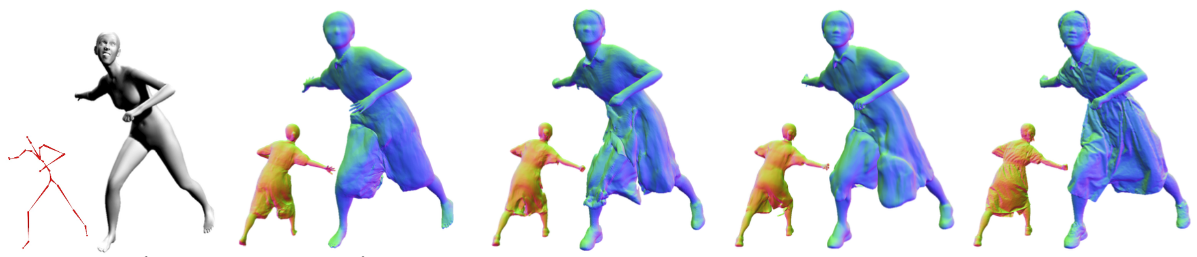

Figure 14. **Human representation learning.** Qualitative results of the novel pose synthesis of state-of-the-art human representation learning approaches together with the GT of 4D-DRESS. All Baseline methods fail to learn the large non-rigid surface deformations and are bounded by the skeletal deformations.

	Inner			Outer		
Method	CD↓	NC↑	IoU↑	CD↓	NC↑	IoU↑
SCANimate [14]	0.965	0.854	0.918	1.237	0.828	0.912
SNARF [4]	1.158	0.843	0.907	1.248	0.827	0.930
X-Avatar [15]	1.008	0.861	0.954	1.177	0.841	0.946

Table 3. **Human representation learning**. Results of human representation learning approaches on 4D-DRESS.

works, SCANimate [14], SNARF [4], X-Avatar [15] on the novel-pose synthesis. Fig. 14 shows that state-of-the-art human representation learning approaches cannot correctly learn the large non-rigid surface deformations (e.g., folded skirt) due to the strong skeletal dependency and the lack of modeling for temporal dynamics. This effect can also be reflected in Tab. 3 quantitatively where all baseline methods produce higher errors on the split of more challenging garments (outer outfits).

References

[1] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Pbns: Physically based neural simulation for unsupervised garment pose space deformation. *ACM Transactions on Graphics (TOG)*, 40(6), 2021. 5, 6

[2] Hugo Bertiche, Meysam Madadi, and Sergio Escalera. Neural cloth simulation. *ACM Transactions on Graphics (TOG)*, 41(6):1–14, 2022. 5, 6

[3] Michael J. Black, Priyanka Patel, Joachim Tesch, and Jinlong Yang. BEDLAM: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8726–8737, 2023. 3

[4] Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021. 8

[5] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (TOG)*, 34(4):1–13, 2015. 4

[6] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 5

[7] Ke Gong, Yiming Gao, Xiaodan Liang, Xiaohui Shen, Meng Wang, and Liang Lin. Graphonomy: Universal human parsing via graph transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 7

[8] Artur Grigorev, Bernhard Thomaszewski, Michael J. Black, and Otmar Hilliges. Hood: Hierarchical graphs for generalized modelling of clothing dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16965–16974, 2023. 5, 6

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4015–4026, 2023. 1

[10] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 7

[11] Kunliang Liu, Ouk Choi, Jianming Wang, and Wonjun Hwang. Cdgnet: Class distribution guided network for human parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4473–4482, 2022. 7

[12] Gerard Pons-Moll, Sergi Pujades, Sonny Hu, and Michael J. Black. Clothcap: Seamless 4d clothing capture and retargeting. *ACM Transactions on Graphics (TOG)*, 36(4), 2017. 3

[13] 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. 1

[14] Shunsuke Saito, Jinlong Yang, Qianli Ma, and Michael J. Black. SCANimate: Weakly supervised learning of skinned clothed avatar networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8

[15] Kaiyue Shen, Chen Guo, Manuel Kaufmann, Juan Zarate, Julien Valentin, Jie Song, and Otmar Hilliges. X-avatar: Expressive human avatars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 5, 8

[16] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 402–419. Springer, 2020. 1

[17] Xingxing Zou, Xintong Han, and Waikeung Wong. Cloth4d: A dataset for clothed human reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12847–12857, 2023. 3