Real-time Free-view Human Rendering from Sparse-view RGB Videos using Double Unprojected Textures – Supplemental Material –

Guoxing Sun[1], Rishabh Dabral[1,2], Heming Zhu[1], Pascal Fua[3], Christian Theobalt[1,2], Marc Habermann[1,2]
[1]Max Planck Institute for Informatics, Saarland Informatics Campus [2]VIA Research Center [3]EPFL
{gsun, rdabral, hezhu, theobalt, mhaberma}@mpi-inf.mpg.de pascal.fua@epfl.ch
https://vcai.mpi-inf.mpg.de/projects/DUT/

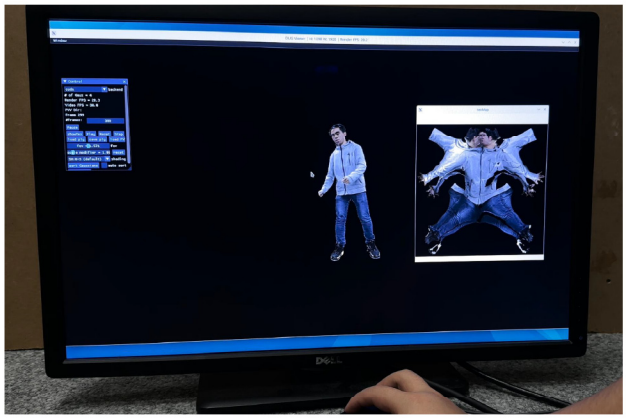


Figure 1. Given four-view image streams and body motions from the disk, our method generates photorealistic human renderings at an interactive speed.

A. Overview

We present more details and experimental results of DUT towards a more clear understanding and in-depth analysis. We first show an interactive demo with our method (Sec. B). Then, we offer additional details about the visibility computation (Sec. C), loss function (Sec. D), implementation details (Sec. E), runtime analysis (Sec. F), undeformed texture maps (Sec. L) and limitation discussions (Sec. M). We also provide additional experimental results about influence with respect to motion capture qualities (Sec. G), out-of-distribution motions (Sec. H), motion sensitivity analysis (Sec. I), performance on a loose and long hair subject (Sec. J), novel lighting (Sec. K). Tab. 4 lists notations and symbols used in the main paper, and demonstrates their descriptions.

B. Interactive Demo

To validate the potential of our method towards live telepresence, we built an interactive system to run our method in an end-to-end manner. As illustrated in Fig. 1 and also in the Supplementary Video, we load images and body motions from the disk of PC and perform live free-viewpoint rendering controlled by a mouse.

C. Visibility Computation

For the methods [3, 10, 11] with texture unprojection, the computation of visibility map is a crucial step, as the goal is to preserve as much information as possible while correctly unprojecting the pixels into the texel space. Here, we provide additional details about the visibility computation. As presented in the Eq. 3, we use normal difference $\mathcal{T}_{\text{v}}^{\text{angle},i}$, depth difference $\mathcal{T}_{\text{v}}^{\text{depth},i}$ and segmentation masks $\mathcal{T}_{\text{v}}^{\text{mask},i}$ as the visibility computation criteria.

The normal difference verification checks if an angle between the surface normal of the world space template and the inverse ray direction from the cameras are roughly parallel to each other. Similar to HoloChar [11], we render the normal texture maps $\mathcal{T}_{\text{N}}^{i}$ and position texture maps $\mathcal{T}_{\text{P}}^{i}$ of the world space template in texel space. The normal difference visibility is computed as:

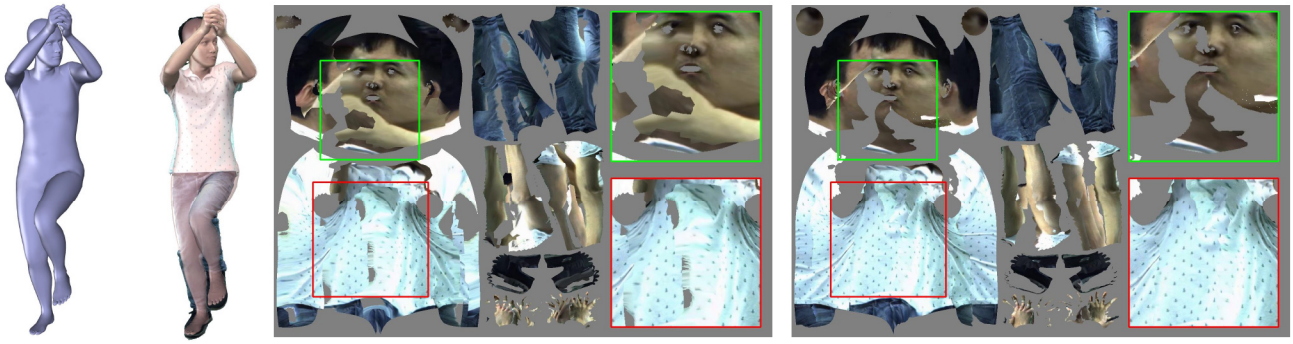
$$\mathcal{T}_{\text{ray}}^{i} = \mathcal{T}_{\text{P}}^{i} - \mathbf{o}^{i} \tag{1}$$

$$\mathcal{T}_{\text{v}}^{\text{angle},i} = cos(-\mathcal{T}_{\text{ray}}^{i}, \mathcal{T}_{\text{P}}^{i}) > \delta \tag{2}$$

where $\mathbf{o}^{i}$ denotes the optical center of camera i and $\mathcal{T}_{\text{ray}}^{i}$ is the texture map of rays. In practice, we set $\delta = 0.17$ in all the experiments.

We also perform depth difference verification to remove the outlier points that are still on the same ray but on the backside of the template. Following prior works [10, 11], we render depth texture maps $\mathcal{T}_{\text{D}}^{i}$ and image coordinate texture maps $\mathcal{T}_{\text{xy}}^{i}$ of the world space template in the texel space and also depth maps $\mathcal{I}_{\text{D}}^{i}$ in the image space. Then, we compute depth difference visibility as :

$$\mathcal{T}_{\text{v}}^{\text{depth},i} = \left| \pi_{uv}(\mathcal{T}_{\text{xy}}^{i}, \mathcal{T}_{\text{D}}^{i}) - \mathcal{I}_{\text{D}}^{i} \right| < \epsilon \tag{3}$$



Posed SMPLX Template | w/ Visibility Map from GHG | w/ Visibility Map from Ours

Figure 2. Compared to the visibility map computed by GHG [3], the visibility map computed by our method is more robust and accurate, generating better unprojected texture map.

In practice, we set $\epsilon = 0.02$ in all the experiments.

Apart from the geometry guidance, we can also utilize segmentation information to distinguish if points are in the foreground or background. Thus, the segmentation visibility can be computed as:

$$\mathcal{T}_{\mathrm{v}}^{\mathrm{mask},i} = \pi_{uv}(\mathcal{T}_{\mathrm{xy}}^{i}, \mathcal{I}_{\mathrm{M}}^{i}) \tag{4}$$

where $\mathcal{I}_{\mathrm{M}}^{i}$ represents the segmentation of image $\mathcal{I}^{i}$.

Instead of geometry clues and segmentation clues, GHG [3] employ the rasterized faces to check the visibility. Specially, they first rasterize face indices to image planes, all the rasterized faces' visibility are tagged as 1.0. Next, they render the face visibility into the texel space to get the final visibility maps. However, we found this routine is not stable, leading to incorrectly unprojected textures (Fig. 2). Thus, to fully explore the upper bound of their method, we use our visibility computation method when re-implementing their method.

D. Loss Function

In this section, we discuss more details about loss functions and how they are computed.

Chamfer Distance. We use Chamfer distance to evaluate the similarity between the vertices of posed deformed template $\hat{\mathbf{P}} = \mathbf{V}(\mathcal{M}, \mathcal{D})$ and ground truth point-clouds $\mathbf{P}_{\mathrm{GT}}$:

$$\mathcal{L}_{\mathrm{Chamf}} = \frac{1}{N_{\mathrm{V}}} \sum_{p_1 \in \hat{P}} \min_{p_2 \in \mathbf{P}_{\mathrm{GT}}} \|p_1 - p_2\|_2^2 \tag{5}$$

$$+ \frac{1}{|\mathbf{P}_{\mathrm{GT}}|} \sum_{p_2 \in \mathbf{P}_{\mathrm{GT}}} \min_{p_1 \in \hat{\mathbf{P}}} \|p_1 - p_2\|_2^2 \tag{6}$$

where N_{V} is the vertex number of human template and $|\mathbf{P}_{\mathrm{GT}}|$ is the vertex number of $\mathbf{P}_{\mathrm{GT}}$.

Laplacian Loss. We apply a Laplacian loss on the posed deformed template $\mathbf{V}(\mathcal{M}, \mathcal{D})$ to ensure the smoothness

$$\mathcal{L}_{\mathrm{Lap}} = \frac{1}{N_{\mathrm{V}}} \sum \|\Delta(N_{\mathrm{V}}, \mathbf{E})\mathbf{V}(\mathcal{M}, \mathcal{D})\|_2^2, \tag{7}$$

where $\mathbf{E}$ represents the edges of the human template and Δ is the uniform Laplacian operator.

Isometry Loss. To prevent severe stretching of template edges, we constrain the edge length of the deformed template $\mathbf{T}_{\mathrm{D}}(\mathcal{D}, \bar{\mathbf{V}})$

$$\mathcal{L}_{\mathrm{Iso}} = \frac{1}{N_{\mathrm{V}}} \sum_{i=1}^{N_{\mathrm{V}}} \frac{1}{|\mathcal{E}_i|} \sum_{e_{i,j} \in \mathcal{E}_i} \|\bar{\mathbf{V}}(e_{i,j}) - \mathbf{T}_{\mathrm{D}}(\mathcal{D}, \bar{\mathbf{V}})(e_{i,j})\|_2^2, \tag{8}$$

where $\mathcal{E}_i$ is the edge set of vertex i, $|\mathcal{E}_i|$ is the edge number of $\mathcal{E}_i$. $\bar{\mathbf{V}}(e_{i,j})$ represents indexing the vertices of edge $e_{i,j}$ from $\bar{\mathbf{V}}$.

Normal Consistency Loss: A normal consistency loss is used to improve the consistency between the normal of a vertex and nearby vertices

$$\mathcal{L}_{\mathrm{Nc}} = \frac{1}{N_{\mathrm{V}}} \sum_{i=1}^{N_{\mathrm{V}}} \frac{1}{|\mathcal{N}_i|} \sum_{v_{i,j} \in \mathcal{N}_i} \|1 - cos(\mathbf{N}_{\mathrm{D}}(v_i), \mathbf{N}_{\mathrm{D}}(v_{i,j}))\|_2^2, \tag{9}$$

where $\mathcal{N}_i$ is the set of nearby vertices of vertex i, $|\mathcal{N}_i|$ is the number of $\mathcal{N}_i$, and $\mathbf{N}_{\mathrm{D}}(v_i)$ denotes the normal of vertex i on the deformed template $\mathbf{T}_{\mathrm{D}}(\mathcal{D}, \bar{\mathbf{V}})$.

L1 Loss: We compute the L1 loss between rendered image $\hat{\mathcal{I}}'$ and ground truth image $\mathcal{I}$ as:

$$\mathcal{L}_{\mathrm{L1}} = \frac{1}{|\mathcal{I}|} \left| \hat{\mathcal{I}}' - \mathcal{I} \right| \tag{10}$$

SSIM Loss: SSIM loss is mainly used to maintain the structure similarity between rendered image $\hat{\mathcal{I}}'$ and ground truth image $\mathcal{I}$:

$$\mathcal{L}_{\mathrm{SSIM}} = 1 - \mathbf{SSIM}(\hat{\mathcal{I}}', \mathcal{I}) \tag{11}$$

IDMRF Loss: We additionally use IDMRF Loss [15] for the perceptual regularization and encouraging high-frequency details.

E. Implementation Details

Motion Capture and Template Tracking. We use a marker-less motion capture approach [12, 14] to recover the body motions for our method and HoloChar [11], which takes images from 34 cameras as inputs. DVA [10] and GHG [3] require SMPLX [9] as the human template. To eliminate the influence of motion capture accuracy, we first transform motions from Captury format to SMPLX format and refine the shape parameters and body parameters with ground truth point-clouds. Notably, our method does not need ground truth point-clouds for motion refinement and can still work with sparse-view motion capture results (Sec. G).

Training and Evaluation. The training views, condition views and evaluation views do not overlap, expect for S2618. The training sequences and evaluation sequences do not overlap, but they share similar types of motion. For the new collected subject, we also have novel action types that are totally out of the distribution of action types in the training set (Tab. 2).

Our method. We implement all the modules of our method in Pytorch [8]. To accelerate the computation of the texture unprojection, we use nvdiffrast [4] for parallel rendering and implement forward kinematics [7] and camera projection with the extension of Pytorch. The visualizer of our interactive demo is built upon 3DGStream [13].

GHG. GHG [3] is a method for generalizable human rendering, we use the offical code and official checkpoints of inpainting network. With our tracked SMPLX and refined visibility map, we freeze the weights of inpainting network and train the Gaussian regressor for each subject respectively.

HoloChar. For each subject, we first train the geometry module [2] of HoloChar [11] with both point-clouds and distance transformation images [1], and then use the official code of HoloChar [11] to train the texture network and the super resolution network.

DVA. With the official code, we train DVA [10] with our tracked SMPLX and same training image splits.

ENeRF. Though ENeRF [5] is not designed for human renderings, however, they have the generalization ability based on the design. For each subject, we give same conditions (four-view images) as our methods and finetune the provided official checkpoint.

F. Runtime Analysis

In this section, we discuss the detailed runtime performance of each methods. Since all methods have different structures of the networks and modeling strategies, we design a runtime evaluation criterion for fair comparisons. More specifically, we neglect the data transportation time and assume an ideal situation that a method receives data, processes data and outputs rendering results. For each method and component, we compute the average time of 100 frames to obtain a stable runtime estimate, and perform it for three times to obtain the final time.

ENeRF. After receiving conditioned images and novel camera parameters, ENeRF [5] can be split into two stages, ray sampling and model inference (including rendering). For 1K resolution, the time of ray sampling is 7.84ms and the time of model inference is 30.83ms, the total time is 38.68 ms and 25.85 FPS. For 4K resolution, the time of ray sampling is 158.68ms and the time of model inference is 351.03ms, the total time is 509.72ms and 1.96 FPS.

DVA. We can also divide DVA [10] into two stages, model inference and rendering. For 1K resolution, the time of model inference is 33.90ms and the time of rendering is 7.53ms, the total time is 41.43ms and 24.13 FPS. For 4K resolution, the time of model inference is 178.45ms and the time of rendering is 7.62 ms, the total time is 186.07ms and 5.37 FPS.

HoloChar. HoloChar [11] includes three stages, geometry inference, texture unprojection and texture inference. Their 1K rendering and 4K rendering are performed together, thus we only report the time for once. The time of geometry inference is 9.3499 ms, the time of texture unprojection is 28.3590 ms and the time of texture inference is 35.7566 ms, the total time is 73.4655 ms and 13.6118 FPS. Notably, we perform all the modules on a workstation with a single GPU and obtain slightly better inference speed, compared to their evaluation.

GHG. GHG [3] has four stages, position map rendering, visibility map rendering, network inference and rendering. Though, in the experiments, we use our visibility computation to obtain better performance of GHG (Sec. C). Here, we only report the time performance of their original implementation. The time of position map rendering is 922.10ms and the time of visibility map rendering is 2754.64ms. For 1K resolution, the time of model inference is 246.83ms and the time of rendering is 214.37ms, the total time is 4137.96ms and 0.2416 FPS. For 4K resolution, the time of model inference is 247.1433 ms and the time of rendering is 213.87ms, the total time is 4137.77ms and 0.2416 FPS.

Ours. Our method contains two main stages, geometry stage and appearance stage. Towards a more comprehensive runtime analysis, we will split each stage into fine-grained components and report the accumulated time to each component. As illustrated in Tab. 1, our method finishes all the operations within 30 ms. The rendering resolution does affect the speed of our method. Besides, we found our method has the potential for improvement when given more power-

GPU	Rnd Res	Tex Res	Stage I			Stage II			FPS
			Forward Kinematic	Obtaining First Unprojected Map	GeoNet Inference	Obtaining Second Unprojected Map	GauNet Inference	Rendering	
3090	1 K	256	3.8723	7.6027	11.4430	14.6424	21.3143	23.5674	42.43
3090	4 K	256	3.8751	7.6097	11.5076	14.7454	21.4354	23.7146	42.17
3090	1 K	512	3.8669	7.5927	16.9868	20.3320	27.0508	29.4855	33.91
3090	4 K	512	3.8601	7.5768	16.9993	20.3178	27.0191	29.3762	34.04
H100	1 K	256	3.8130	6.9313	10.7382	13.3484	16.5124	18.7570	53.31
H100	4 K	256	3.8489	6.9685	10.7392	13.3512	16.4705	18.7303	53.39
H100	1 K	512	3.7821	6.8754	12.9675	15.6070	18.7620	21.0133	47.59
H100	4 K	512	3.8300	6.8982	12.9877	15.6178	18.7669	21.0022	47.61

Table 1. **Quantitative Ablation.** Here, we demonstrate detailed runtime ablation of our methods with different texture resolutions (Tex Res), rendering resolutions (Rnd Res) and GPUs. All the time in the table is the accumulated time from the beginning and their units are milliseconds.

Training & Testing Action			
Jogging	Walking	Looking	Picking Up
Talking	Waving	Celebrating	Jumping
Baseball Throwing	Baseball Swing	Boxing	Goalkeeping
Penalty Kick	Golf	Archery	Weight Lifting
Squating	Jumping Jack	Playing Instrument	Dancing
Opening and Pushing Door	Playing Table Tennis	Playing Badminton	Giving Presentation
Drinking	Using Phone	Petting Animal	Playing Hockey
Playing Tug of War	Juggle Balls	Playing Hula Hoop	Bowling
Playing Volleyball	Wrestling	Stretching	Mopping Floor
Digging	Typing	Cooking	Using Spray
Applying Makeup			
Out of Distribution Action			
Shooting	Surrender	Fishing	Standing long jump
Single-leg hop	Frog Jumping	Crawling	Rolling on the Ground
Sleeping	Ultraman		

Table 2. The action types of body motions in the training split, testing split and out-of-distribution split.

Methods	Motion	**Tex Res**	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
Ours-Large	Sparse	512	30.0309	0.8722	0.1322
Ours-Large	Dense	512	30.0311	0.8722	0.1322

Table 3. **Quantitative Ablation.** We evaluate the influence from mocap quality. Our method still produces good results with sparse-view captured motions.

ful GPU, i.e. H100.

G. Ablation on Mocap Quality

In Fig. 3, we evaluate how does the motion capture quality affect the performance of our method on S3. We use the same mocap method [12, 14] but only using 4 condition views as the inputs, obtaining the sparse mocap results. There is a small performance drop when replacing dense motion capture with sparse motion capture on the PSNR metric. We believe this is due to the fact that current motion capture methods could produce reasonable results with sparse-view cameras, while our method could compensate for motion capture quality to some extent.

H. Additional Comparisons on OOD Motions

In Fig. 3, we show additional quantitative comparisons between our method and HoloChar [11] on the sequences with out-of-distribution motions. Our method produces consistently better results, and also slower standard deviations. In terms of PSNR, our method's standard deviation is **1.411**, while HoloChar gets a standard deviation of 1.8912, which reveals the robustness of our method to the OOD motions.

I. Sensitivity Analysis to Motion Errors

We show a sensitivity analysis by manually adding linear motion capture errors to left elbow from 0° to $60°13'$ in Fig. 4. Surprisingly, DUT produces good results within $7°46'$ error (0°, $3°43'$, $7°46'$) and reasonable results even

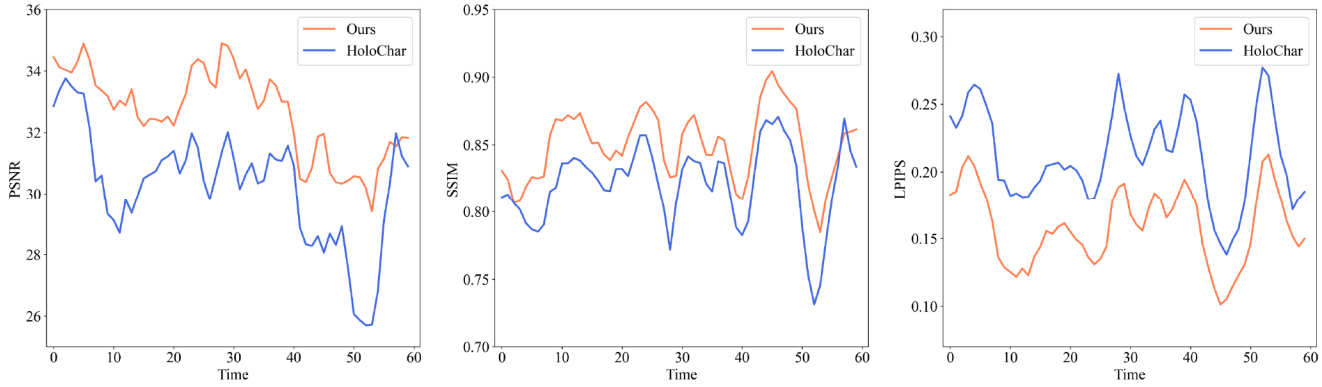


Figure 3. **Quantitative Comparison.** We quantitatively compare the rendering results of our method and HoloChar [11] on out-of-distribution human motions. Our method outputs consistently better rendering results.

Figure 4. **Motion Sensitivity Analysis to Increasing Errors**. Our method still outputs reasonable results to different level of motion capture errors at inference.

with $60°13'$ error.

J. Performance on Loose Clothing and Long Hair Subject

In Fig. 5, we show a subject with dynamic long-hair wearing a dress. Our geometry module recovers the coarse geometry of hair dynamics and dress. After which, our rendering module faithfully generates high fidelity renderings.

K. Results under Novel Lighting Environment

Towards realistic application scenarios, once our method is trained, we may run it with novel cameras under novel lighting environments. We can either perform color augmentation to the input views during training [16] or finetune the model on the sparse input views captured under the new illumination. We provide a preliminary result for this in Fig. 6 where we finetuned the model on the input views and **isolated frames (not used in testing)** in a new lighting condition. After finetuning, DUT still runs in feed-forward manner. Our method can be easily adopted to the novel lighting and produce reasonable rendering results.

L. Additional Discussions about Undeformed (First) Texture Map

Our GeoNet Φ_{Geo} estimates template deformations from undeformed texture map $\mathcal{T}_{\text{c,1st}}$ and non-root normal map $\mathcal{T}_{\bar{\text{N}}}$. Similar to the normalized skeletal motion in DDC [2], $\mathcal{T}_{\bar{\text{N}}}$ provides the pose information. However, dynamic geometries of a moving human body are not completely determined by the skeletal pose at that moment, it leads to one-to-many mapping issue [6]. Our undeformed texture map $\mathcal{T}_{\text{c,1st}}$ offers additional information about the degree of deformations. As shown in Fig. 7, under the same body pose, the degree of deformations can be reflected by the distortions of $\mathcal{T}_{\text{c,1st}}$.

M. More Discussions about Limitations

Color Fluctuation. Though simple and efficient, our method suffers from a certain extent color fluctuation of some frames as shown in the video. We found that this could be attributed to the predicted Gaussians are trying to overfit the uneven lightings of studio and shadows on the body, which are challenging for such simple representation. Integrating ray tracing or physically based rendering may

Figure 5. **Qualitative results.** We perform our method on a loose and long-hair subject, it manages to capture the coarse deformations of hair and dress and produces faithful rendering results.

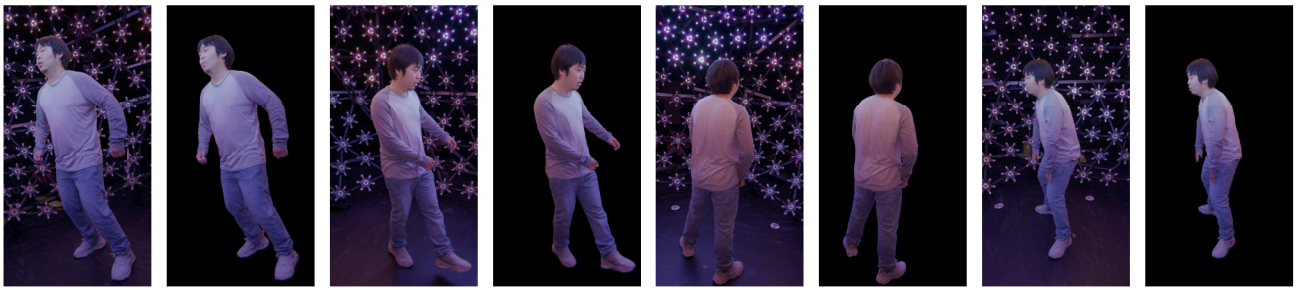


Figure 6. **Qualitative results**. Results with finetuned model under novel lighting. After finetuning, DUT still runs in feed-forward manner.

reduce the color fluctuation.

Topology Change. In Fig. 8, we show results on a sequence where the subject is taking clothes off. While results look reasonable, still the quality degrades. Though a fixed template contributes a lot, it will be an interesting direction to investigate how to introduce dynamic template update into such task, especially with only RGB inputs.

References

[1] Gunilla Borgefors. Distance transformations in digital images. Computer vision, graphics, and image processing, 34 (3):344–371, 1986. 3

[2] Marc Habermann, Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Gerard Pons-Moll, and Christian Theobalt. Real-time deep dynamic characters. ACM Transactions on Graphics (ToG), 40(4):1–16, 2021. 3, 5

[3] Youngjoong Kwon, Baole Fang, Yixing Lu, Haoye Dong, Cheng Zhang, Francisco Vicente Carrasco, Albert Mosella-Montoro, Jianjin Xu, Shingo Takagi, Daeil Kim, et al. Generalizable human gaussians for sparse view synthesis. ECCV, 2024. 1, 2, 3

[4] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics, 39(6), 2020. 3

[5] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In SIGGRAPH Asia Conference Proceedings, 2022. 3

[6] Lingjie Liu, Marc Habermann, Viktor Rudnev, Kripasindhu Sarkar, Jiatao Gu, and Christian Theobalt. Neural actor: Neural free-view synthesis of human actors with pose control. ACM Trans. Graph.(ACM SIGGRAPH Asia), 2021. 5

[7] Haokai Pang, Heming Zhu, Adam Kortylewski, Christian Theobalt, and Marc Habermann. Ash: Animatable gaussian splats for efficient and photoreal human rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 1165–1175, 2024. 3

[8] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 3

[9] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), pages 10975–10985, 2019. 3

[10] Edoardo Remelli, Timur Bagautdinov, Shunsuke Saito, Chenglei Wu, Tomas Simon, Shih-En Wei, Kaiwen Guo, Zhe Cao, Fabian Prada, Jason Saragih, et al. Drivable volumetric avatars using texel-aligned features. In ACM SIGGRAPH 2022 Conference Proceedings, pages 1–9, 2022. 1, 3

[11] Ashwath Shetty, Marc Habermann, Guoxing Sun, Diogo Luvizon, Vladislav Golyanik, and Christian Theobalt. Holoported characters: Real-time free-viewpoint rendering of humans from sparse rgb cameras. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1206–1215, 2024. 1, 3, 4, 5

[12] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In 2011 International

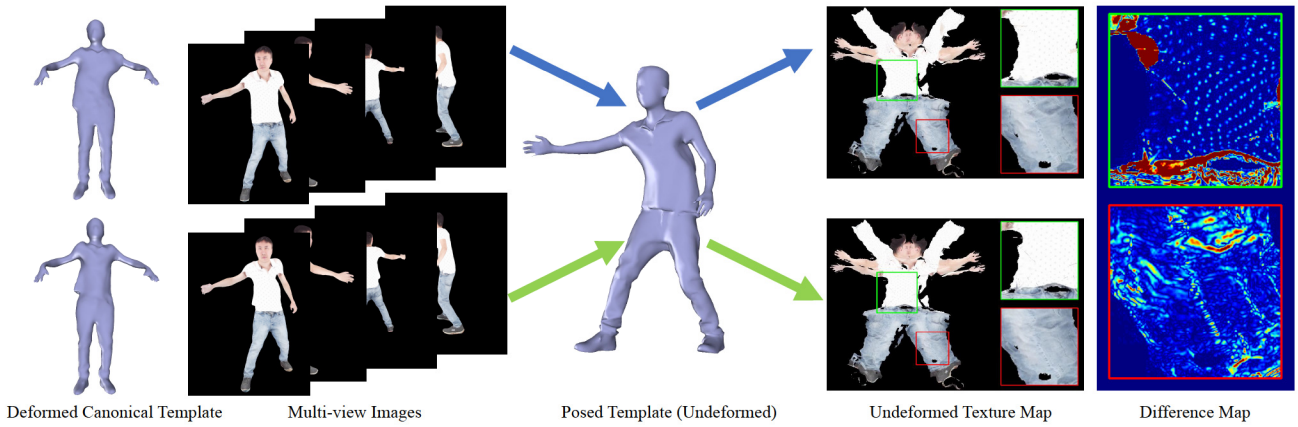


Figure 7. **Illustration of Undeformed Texture Map**. The distortions of undeformed (first) texture maps are directly related with deformations on the canonical template.

Figure 8. **Topology Change**. Results of taking off cloth.

Conference on Computer Vision, pages 951–958. IEEE, 2011. 3, 4

[13] Jiakai Sun, Han Jiao, Guangyuan Li, Zhanjie Zhang, Lei Zhao, and Wei Xing. 3dgstream: On-the-fly training of 3d gaussians for efficient streaming of photo-realistic free-viewpoint videos. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 20675–20685, 2024. 3

[14] TheCaptury. The Captury. http://www.thecaptury.com/, 2023. 3, 4

[15] Yi Wang, Xin Tao, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. Image inpainting via generative multi-column convolutional neural networks. In Advances in Neural Information Processing Systems, pages 331–340, 2018. 3

[16] Donglai Xiang, Fabian Prada, Zhe Cao, Kaiwen Guo, Chenglei Wu, Jessica Hodgins, and Timur Bagautdinov. Drivable avatar clothing: Faithful full-body telepresence with dynamic clothing driven by sparse rgb-d input. In SIGGRAPH Asia 2023 Conference Papers, pages 1–11, 2023. 5

Symbol	Description
$\mathcal{D}$	Deformation parameters
$\mathcal{M}$	Body motion parameters
$\bar{\mathbf{V}}$	Vertices of human template mesh in the canonical body pose
N_{V}	Number of vertices on human template mesh
$\mathbf{T}_{\mathrm{LBS}}(\cdot)$	LBS transformation function
$\mathbf{T}_{\mathrm{D}}(\cdot)$	Mesh deformation function
$\mathbf{H}(\cdot)$	A function that converts spherical harmonics coefficients $\mathbf{h}$ into RGB colors $\mathbf{s}$
x	3D positions in the Euclidean space
$\mathbf{R}$	3D rotation matrix
$\mathbf{S}$	3D scale matrix
$\mathbf{J}$	Jacobian of the affine approximation of the projective transformation
$\mathbf{W}$	3D view matrix
$\mathbf{G}$	A set of Gaussian parameters, including $\mathbf{p}$, $\mathbf{r}$, $\mathbf{h}$, $\mathbf{s}$, $\boldsymbol{\alpha}$
$\mathbf{p}$	Positions of 3D Gaussians in world space
$\mathbf{r}$	Rotations of 3D Gaussians
$\mathbf{h}$	Spherical harmonics of 3D Gaussians
$\mathbf{s}$	Scales of 3D Gaussians
$\boldsymbol{\alpha}$	Density values of 3D Gaussians
$\boldsymbol{\alpha}^{'}$	Density values of 3D Gaussians in 2D
$\mathbf{c}$	Color values of 3D Gaussians
$\mathcal{T}_{\mathrm{c}}$	Unprojected texture map in texel space
$\mathcal{T}_{\mathrm{v}}$	Visibility map in texel space
$\mathcal{T}_{\mathrm{v}}^{\mathrm{angle}}$	Visibility map computed by normal difference in texel space
$\mathcal{T}_{\mathrm{v}}^{\mathrm{depth}}$	Visibility map computed by depth difference in texel space
$\mathcal{T}_{\mathrm{v}}^{\mathrm{mask}}$	Visibility map computed by segmentation mask in texel space
Φ_{Geo}	Geometry network that estimates deformations of human body template in texel space
Φ_{Gau}	Gaussian network that estimates Gaussian parameters in texel space
$\mathcal{T}_{\bar{\mathrm{N}}}$	Normal map of posed template but without root rotation in texel space
$\mathcal{T}_{\mathrm{c,1st}}$	First unprojected texture map in texel space
$\mathcal{T}_{\mathrm{c,2nd}}$	Second unprojected texture map in texel space
$\boldsymbol{\mathcal{G}}$	A set of modified Gaussian parameters, including $\mathbf{d}$, $\mathbf{r}$, $\mathbf{h}$, $\mathbf{s}$, $\boldsymbol{\alpha}$
$\mathcal{M}_{\mathcal{G}}$	Mask map of valid Gaussians in texel space
$R(\cdot)$	Gaussian renderer
$\mathcal{T}_{\mathrm{LBS}}$	LBS transformations map in texel space
$\mathcal{T}_{\mathbf{T}_{\mathrm{D}}(\mathcal{D},\bar{\mathbf{V}})}$	Deformed base geometry map in texel space
$\mathcal{T}_{\mathcal{G}}$	Modified Gaussian map $\boldsymbol{\mathcal{G}}$ in texel space
$\mathcal{T}_{\boldsymbol{\mathcal{G}},\chi}$	Feature map χ of $\mathcal{T}_{\mathcal{G}}$ in texel space, $\chi \in \{\mathbf{d}, \mathbf{h}, \mathbf{s}, \mathbf{r}, \boldsymbol{\alpha}\}$
$\pi_{uv}(\cdot)$	The function that indexes the features in the texel space.
$\hat{\mathcal{I}}'$	Rendered image with Gaussian scale refinement
$\mathcal{T}_{s'}$	Refining scale map in texel space
$\mathcal{L}_{\mathrm{Chamf}}$	Chamfer distance
$\mathcal{L}_{\mathrm{Lap}}$	Laplacian loss
$\mathcal{L}_{\mathrm{Iso}}$	Isometry loss
$\mathcal{L}_{\mathrm{Nc}}$	Normal consistency loss
$\mathcal{L}_{\mathrm{L1}}$	L1 loss
$\mathcal{L}_{\mathrm{SSIM}}$	SSIM loss
$\mathcal{L}_{\mathrm{IDMRF}}$	IDMRF loss
$\mathcal{L}_{\mathrm{Reg}}$	Geometric regularization loss

Table 4. Notations and symbols.