Supplementary Material - PhysAvatar: Learning the Physics of Dressed 3D Avatars from Visual Observations

Yang Zheng[1⋆], Qingqing Zhao[1⋆], Guandao Yang[1], Wang Yifan[1], Donglai Xiang[2], Florian Dubost[3], Dmitry Lagun[3], Thabo Beeler[3], Federico Tombari[3,4], Leonidas Guibas[1], and Gordon Wetzstein[1]

[1] Stanford University
[2] Carnegie Mellon University
[3] Google
[4] Technical University of Munich

Appendix

A Implementation Details

A.1 SMPL-X Fitting

We observe that the human pose estimations provided by Actor-HQ [4] are noisy, and failure cases exist in some sequences with challenging poses. Since our method relies on a body collider that provides accurate contact information for garment simulation, an SMPL-X mesh that aligns well with the human pose and shape is necessary. To this end, we fit the SMPL-X parameters from multi-view videos using an algorithm [10, 11] simplified for our setting. Specifically, given image $\mathbf{I}_{i,t}$ captured from camera i at t timestamp, we detect 2D human keypoints $\mathbf{x}_{i,t}$ using state-of-the-art human keypoints detector DWPose [3,9] and calculate the projection loss as follows:

$$\mathcal{L}_{2d} = \left\| \mathbf{x}_{i,t} - \hat{\mathbf{x}}_{i,t} \right\|_2 , \tag{1}$$

where $\hat{\mathbf{x}}_{i,t}$ is the projection of SMPL-X 3D joints under camera i. We also employ a regularization term as:

$$\mathcal{L}_{reg} = \lambda_m \left\| m_h \right\|_2 + \lambda_l \left\| \theta_b \right\|_2 + \lambda_f \left\| \theta_f \right\|_2 + \lambda_\beta \left\| \beta \right\|_2 + \lambda_\psi \left\| \psi \right\|_2 , \tag{2}$$

where m_h, β_b, θ_f denote the hand, body, and facial pose latent parameters, β refers to body shape parameters, and ψ is the facial expression parameters, as introduced in SMPL-X [8]. The full loss used to optimize SMPL-X parameters is formally defined as:

$$\mathcal{L}_{fitting} = \lambda_b \mathcal{L}_{b2d} + \lambda_h \mathcal{L}_{h2d} + \mathcal{L}_{reg}, \tag{3}$$

⋆ Equal Contribution

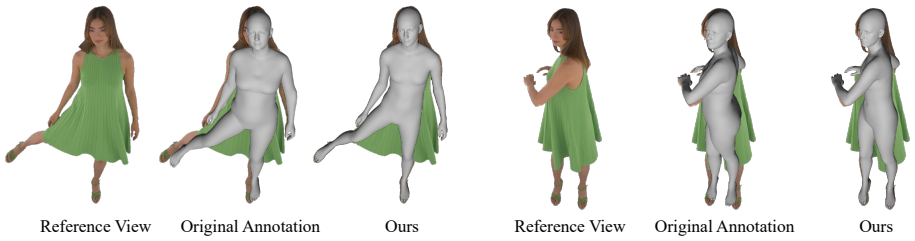

Fig. 1: Our SMPL-X [8] fitting results show better accuracy compared to the original annotations from Actor-HQ [4] dataset.

where $\mathcal{L}_{h2d}$ is the projection loss for hand joints, and $\mathcal{L}_{b2d}$ is the projection loss for the rest body joints. We set $\lambda_b = 1e-3$, $\lambda_h = 1e-3$, $\lambda_m = 1$, $\lambda_l = 1e-3$, $\lambda_f = 1e$, $\lambda_\beta = 3e-2$, $\lambda_\psi = 1e-1$ to balance the loss terms. In order to maintain temporal consistency, we use the optimized SMPL-X parameters from the previous frame as initialization. As demonstrated in Fig. 1, our fitting results are better aligned with the actual human pose.

A.2 Skin Weight Transfer

We animate the body part of the mesh (non-garment region) using Linear Blend Skinning (LBS), employing a robust skin weight transfer algorithm through weight inpainting as introduced by Abdrashitov *et al.* [1]. This method effectively transfers skin weights from the SMPL-X [8] mesh (source) to the human mesh (target) in two stages. Initially, we directly transfer skinning weights to target vertices that meet the closest point matching criteria, thoroughly discussed in Section 3.1 of Abdrashitov *et al.* [1]. Subsequent to this direct transfer, for vertices that remain without skinning weights, we apply weight inpainting. This step involves extrapolating the missing weights to ensure a seamless and natural skin deformation across the mesh, treated as an optimization problem and described in Section 3.2 of Abdrashitov *et al.* [1]. We refer the reader to Abdrashitov *et al.* [1] for a detailed explanation, where the processes and their applications are comprehensively described.

A.3 Garment Physics Parameter Estimation

We outline the pseudo-code for estimating garment physics parameters in Algorithm 1 and provide details of the simulator and boundary node specifications below.

Simulator For simulating garment dynamics, we adopt the open-source C-IPC simulator [5], available at https://github.com/ipc-sim/Codim-IPC. In the task of garment material estimation, we opt for a sequence length of 24 frames, balancing speed with the need for sufficient dynamic information. A longer sequence, while

containing more dynamic information, demands prohibitive simulation times, making it impractical for optimization tasks. Conversely, a shorter sequence might fail to capture adequate dynamics, offering insufficient data for material estimation. The garment simulation runs around $0.05 - 1$ fps on the CPU.

Boundary Nodes Specification In our setup, occlusions within the garment mesh—such as areas near the shoulders obscured by hair—require the designation of specific garment nodes as boundary points, also known as Dirichlet boundary conditions. During simulation, the garment is driven by these boundary nodes in conjunction with the underlying body collider. To identify these boundary points, we focus on regions with minimal relative movement between the garment and the body, such as those near the shoulder. The dynamics of these selected boundary points are then calculated using Linear Blend Skinning (LBS), utilizing the nearest corresponding SMPL skinning weights, to obtain $\mathbf{V}^b_{1:T}$.

Hyperparameters We configure the simulation with a step size of $\delta t = 0.04$ and optimize the garment physics parameters over 100 iterations using the Adam optimizer. For finite difference-based gradient estimation, we set $\delta\rho = 5$, $\delta\kappa_s = 0.05$, and $\delta\kappa_b = 0.05$, with ρ ranging from 200 to 640, κ_b and κ_s each ranging from 0.1 to 8. The scaling for these parameters is detailed in the C-IPC codebase.

A.4 Appearance Modeling

Lighting With the ActorHQ dataset [4], we assume that lighting conditions can be approximated by constant environmental lighting, represented by a global ambient light parameterized by a global L_a. We initialize L_a to have a unit power per unit area per unit steradian across all three color channels.

Rendering We utilize the differentiable path tracer Mitsuba 3 [7], leveraging GPU backends for accelerated rendering. During optimization, we apply 8 samples per pixel (spp) for efficiency. For final rendering, this is increased to 128 spp for enhanced quality. Relighting and additional rendering tasks are conducted in Blender [2].

Optimization Details For inverse rendering, we employ the Adam optimizer, conducting optimization for 2000 iterations for each character. At each gradient step, a timestep and camera view are randomly selected. Supervision is applied within the masked region only for each character. Given the imperfection of the ground-truth masks, we also erode the ground-truth mask by 2 pixels to mitigate potential artifacts. We use $\text{lr} = 0.01$ and decay the learning rate by 0.5 every 500 iterations. We clip the optimized texture after each gradient update to ensure legal color values.

ALGORITHM 1: Garment Physics Parameter Estimation

Input: Tracked mesh sequences $\{\mathbf{V}_{1:T}, \mathbf{F}\}$, Garment Boundary condition $\{\mathbf{V}^b_{1:T}\}$, Body Collider $\{\mathbf{V}^C_{1:T}, \mathbf{F}^C\}$, initial garment density ρ, initial garment membrane stiffness κ_s, initial bending stiffness κ_b, step size δt, density step size δ_ρ, bending stiffness step size δ_{κ_b}, membrane stiffness step size δ_{κ_s}, the optimizer Adam$(\cdot)$, and the cloth simulator $f(\cdot)$.

Output: ρ, κ_s, κ_b

Function `get_loss`$(\rho, \kappa_s, \kappa_b)$**:**

$\mathcal{L}_{\text{sim}} = 0$
$\dot{\mathbf{V}}^g_0 = 0$
$\mathbf{V}^g_0 = \mathbf{V}_0$
for $0 \leq t \leq T$ **do**
$\mathbf{V}^g_{t+1} = f(\mathbf{V}^g_t, \dot{\mathbf{V}}^g_t, \mathbf{V}^b_{t+1}, \mathbf{V}^C_{t+1}, \rho, \kappa_s, \kappa_b, \Delta t)$
$\mathcal{L}_{\text{sim}} = \mathcal{L}_{\text{sim}} + \mathbf{MSE}(\mathbf{V}^g_{t+1}, \mathbf{V}^g_t)$
end
return $\mathcal{L}_{\text{sim}}$

for $0 \leq i \leq itermax$ **do**

#Apprximate Gradient using finite difference
$\Delta\rho = (\texttt{get\_loss}(\rho + \delta_\rho, \kappa_s, \kappa_b) - \texttt{get\_loss}(\rho, \kappa_s, \kappa_b))\,/\delta_\rho$
$\Delta\kappa_s = (\texttt{get\_loss}(\rho, \kappa_s + \delta_{\kappa_s}, \kappa_b) - \texttt{get\_loss}(\rho, \kappa_s, \kappa_b))\,/\delta_{\kappa_s}.$
$\Delta\kappa_b = (\texttt{get\_loss}(\rho, \kappa_s, \kappa_b + \delta_{\kappa_b}) - \texttt{get\_loss}(\rho, \kappa_s, \kappa_b))\,/\delta_{\kappa_b}.$

#Update the parameter
$\rho = \text{Adam}_\rho(\Delta\rho)$
$\kappa_s = \text{Adam}_{\kappa_s}(\Delta\kappa_s)$
$\kappa_b = \text{Adam}_{\kappa_b}(\Delta\kappa_b)$
end

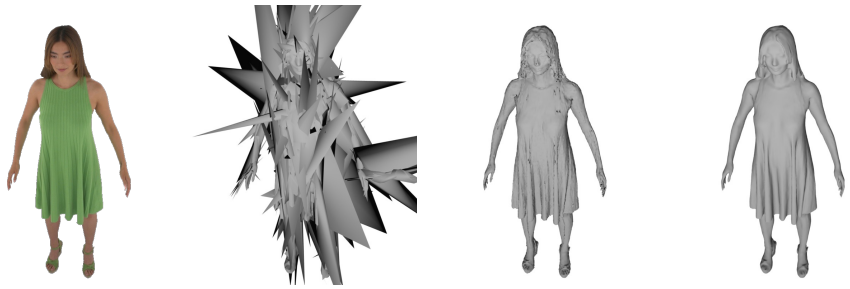

Fig. 2: Ablation Study on loss terms used in mesh tracking. Without the ISO loss, the mesh topology gradually becomes chaotic (W/o ISO Loss). Without the normal loss, the mesh surface becomes noisy during optimization (W/o Normal Loss). With the full loss terms, we can obtain a smooth and robust mesh tracking result.

B Additional Results

B.1 Ablation Study On Mesh Tracking

As introduced in Sec. 3 of the main paper, we propose a robust mesh tracking method to capture accurate geometry correspondences, which is crucial for the subsequent physical parameter estimations. To validate the effectiveness of our approach, we conduct an ablation study focusing on the various loss terms employed during the mesh tracking process. As demonstrated in Fig. 2, the ISO loss is important for preserving the local mesh topology. This ensures that the mesh deformation remains realistic by maintaining edge lengths, thereby preventing distortions. Additionally, our introduced normal loss is a key component to generating smooth and robust tracking results. Moreover, we don't observe significant contributions for the regularization losses aside from the ISO loss introduced in Dynamic 3D Gaussians [6] (e.g., rigidity loss and rotation loss), which might not be necessary in our setting.

B.2 Additional Animation Results

We show additional animation results on our website: `https://qingqing-zhao.github.io/PhysAvatar`. Please refer to it for better visualization.

References

1. Abdrashitov, R., Raichstat, K., Monsen, J., Hill, D.: Robust skin weights transfer via weight inpainting. In: SIGGRAPH Asia 2023 Technical Communications, pp. 1–4 (2023)
2. Blender Online Coummunity: Blender, blender Foundation, https://www.blender.org/

3. Contributors, M.: Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose (2020)
4. Işık, M., Rünz, M., Georgopoulos, M., Khakhulin, T., Starck, J., Agapito, L., Nießner, M.: Humanrf: High-fidelity neural radiance fields for humans in motion. ACM Transactions on Graphics (TOG) **42**(4), 1–12 (2023). https://doi.org/10.1145/3592415, https://doi.org/10.1145/3592415
5. Li, M., Kaufman, D.M., Jiang, C.: Codimensional incremental potential contact. ACM Transactions on Graphics (TOG) **40**(4), 1–24 (2021)
6. Luiten, J., Kopanas, G., Leibe, B., Ramanan, D.: Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. arXiv preprint arXiv:2308.09713 (2023)
7. Nimier-David, M., Vicini, D., Zeltner, T., Jakob, W.: Mitsuba 2: A retargetable forward and inverse renderer. ACM Transactions on Graphics (TOG) **38**(6), 1–17 (2019)
8. Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A.A.A., Tzionas, D., Black, M.J.: Expressive body capture: 3D hands, face, and body from a single image. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 10975–10985 (2019)
9. Yang, Z., Zeng, A., Yuan, C., Li, Y.: Effective whole-body pose estimation with two-stages distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4210–4220 (2023)
10. Zhang, Y., Li, Z., An, L., Li, M., Yu, T., Liu, Y.: Lightweight multi-person total motion capture using sparse multi-view cameras. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5560–5569 (2021)
11. Zheng, Y., Shao, R., Zhang, Y., Yu, T., Zheng, Z., Dai, Q., Liu, Y.: Deepmulticap: Performance capture of multiple characters using sparse multiview cameras. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6239–6249 (2021)