Supplementary Material

A. Implementation Details

We preprocess the dataset following the ARAH[1]. During optimization, we follow the same strategy from [4] to densify and prune the 3D Gaussians, using the view-space position gradients derived from the transformed Gaussians in the observation space as the criterion for densification.

Our model is trained for a total of $15k$ iterations on the ZJU-MoCap [5] dataset and $12k$ iterations on H36M [2] on a single NVIDIA RTX 3090 GPU. We use Adam to optimize our model and the per-frame latent codes with hyperparameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate of 3D Gaussians is exactly the same as the original implementation from [4]. We set the learning rate for forward skinning network θ_r to 1×10^{-4}, 5×10^{-4} for 3D sparse U-Net, and 1×10^{-3} for all the others. An exponential learning rate scheduler is employed to gradually decrease the learning rate by a factor of 0.1 on neural networks. We also apply a weight decay with a weight of 0.05 to the per-frame latent codes.

Following prior works [6, 8], we split the training stage and learn the whole model in a coarse-to-fine manner. In the first $1k$ iterations, we freeze everything except the forward skinning network f_{θ_r} to learn a coarse skinning field with $\mathcal{L}_{skin}$. We then enable optimization on the 3D Gaussians after $1k$ steps. To decouple rigid and non-rigid motion, we start to optimize the non-rigid deformation network $f_{\theta_{nr}}$ after $3k$ iterations. Lastly, we turn on Geometric and Semantic Feature Learning after $5k$ iterations.

B. Implementation Details for Baselines

In this section, we elaborate on the implementation details of baselines used for comparison to our proposed method, *i.e.* NeuralBody [5], HumanNeRF [8], MonoHuman [9] and InstantAvatar [3].

For NeuralBody [5], HumanNeRF [8], MonoHuman [9] and InstantAvatar [3], we use the results of them reported in 3DGS-Avatar [6] which follow the same data split.

For 3DGS-Avatar [6], we train the models using the code from official code repository[2]. For GauHuman [1], we train the models using the code from official code repository[3] for 15000 epochs. For GoMAvatar [7], we train the models using the code from official code repository[4]. All other hyperparameters remain unchanged. The trained models are then used for qualitative evaluation and out-of-distribution pose animation.

[1]https://github.com/taconite/arah-release
[2]https://github.com/mikeqzy/3dgs-avatar-release
[3]https://github.com/skhu101/GauHuman

C. Loss Definition

In the main paper we describe our loss term which can be formulated as follows:

$$\begin{aligned}\mathcal{L}_{reconstruct} = &\mathcal{L}_{rgb} + \lambda_1 \mathcal{L}_{mask} + \lambda_2 \mathcal{L}_{SSIM} + \lambda_3 \mathcal{L}_{LPIPS} \\ &+ \lambda_4 \mathcal{L}_{skin} + \lambda_5 \mathcal{L}_{isopos},\end{aligned} \tag{1}$$

$$\mathcal{L} = \mathcal{L}_{reconstruct} + \lambda_6 \mathcal{L}_{semantic} + \lambda_7 \mathcal{L}_{neighborhood}. \tag{2}$$

We describe how each loss term is defined below:

RGB Loss. We employ an $l1$ loss for pixel-wise error and a perceptual loss for robustness against local misalignments, crucial in monocular setups.

Mask Loss. To boost the convergence of 3D Gaussian positions, we use an explicit mask loss. For each pixel p, we compute the opacity value O_p by summing up the sample weights in the rendering equation in the main paper:

$$O_p = \sum_i \alpha'_i \prod_{j=1}^{i-1} (1 - \alpha'_j). \tag{3}$$

We thus supervise it with the ground truth foreground mask via an $l1$ loss. Experiments show that the $l1$ loss provides faster convergence than the Binary Cross Entropy (BCE) loss.

SSIM Loss. We further employ SSIM to ensure the structural similarity between ground truth target image C and synthesized images $\hat{C}$:

$$\mathcal{L}_{SSIM} = SSIM(\hat{C}, C). \tag{4}$$

[4]https://github.com/wenj/GoMAvatar

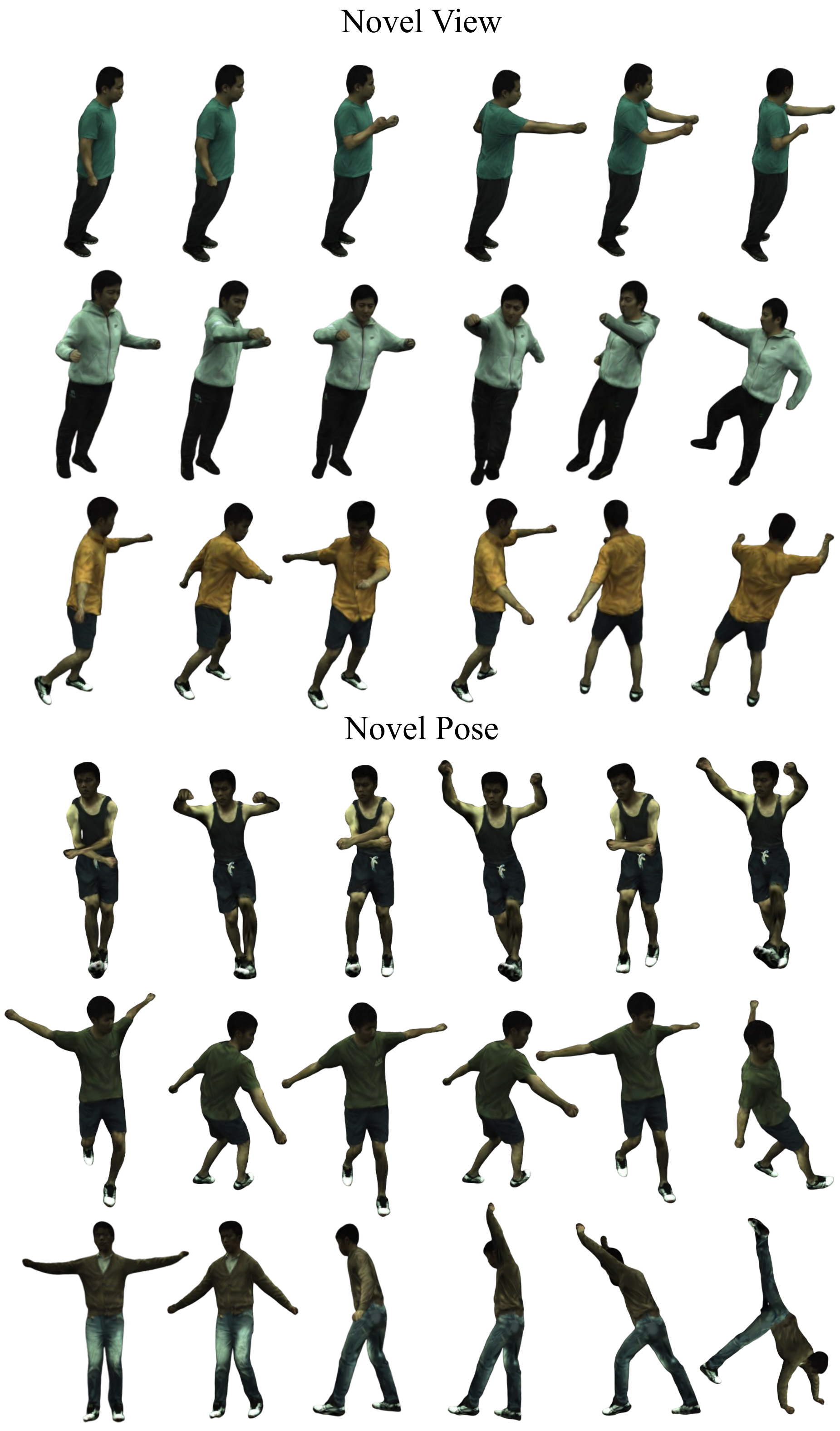

Figure 1. More visualization results on novel view and novel pose.

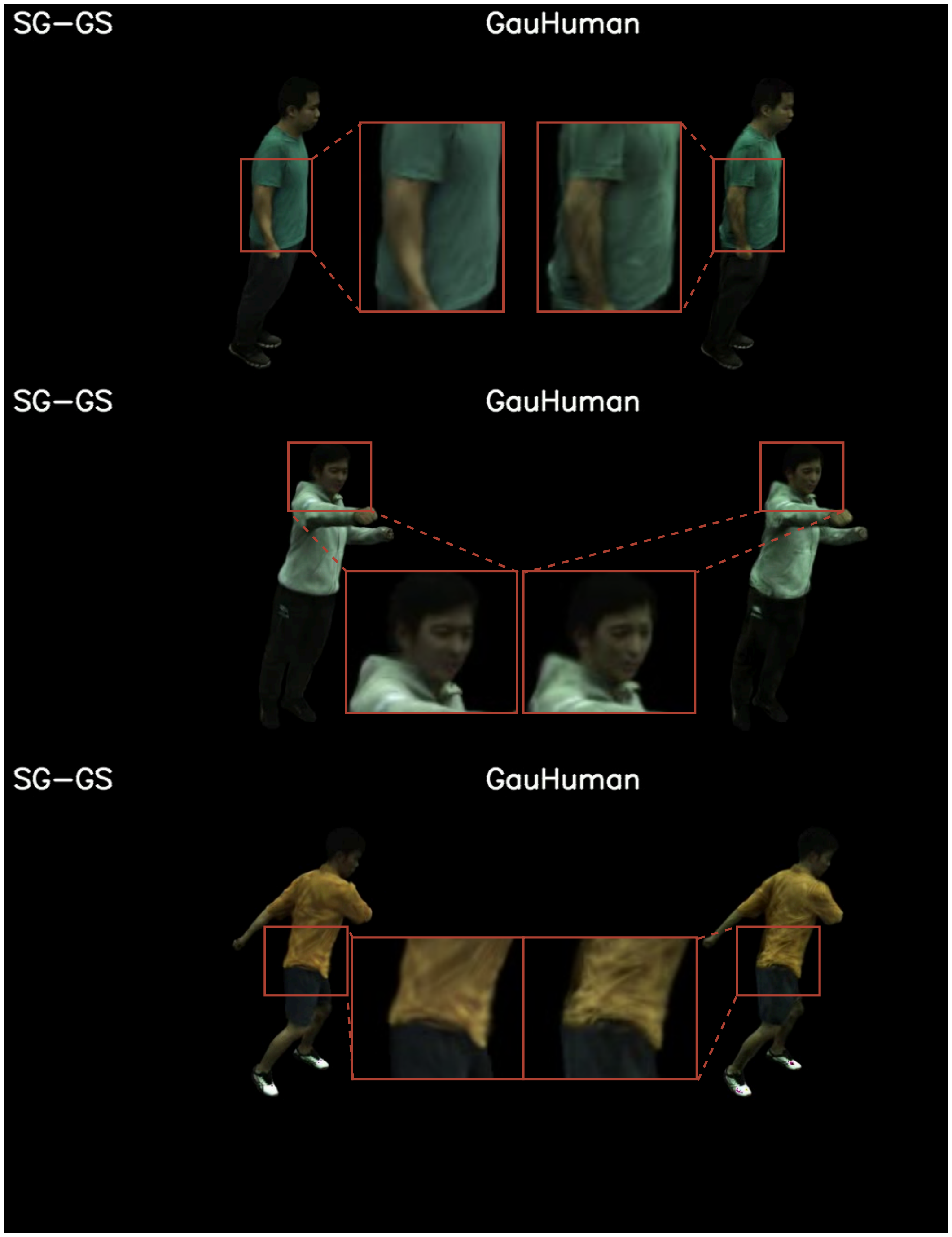

Figure 2. More comparison results on novel view with GauHuman [1].

LPIPS Loss. Following [8], we use VGG-based LPIPS as the perceptual loss. Unlike NeRF methods, we render full images via rasterization, eliminating the need for patch sampling. For efficiency, we compute LPIPS on cropped bounding boxes using ground truth masks:

$$\mathcal{L}_{LPIPS} = LPIPS(\hat{C}, C). \quad (5)$$

Skinning Loss: We leverage SMPL prior by sampling 1024 points $\mathbf{X}_{skin}$ on the surface of the canonical SMPL mesh and regularizing the forward skinning network with corresponding skinning weights $\mathbf{w}$ interpolated with barycentric coordinates.

$$\mathcal{L}_{skin} = \frac{1}{|\mathbf{X}_{skin}|} \sum_{\mathbf{x}_{skin} \in \mathbf{X}_{skin}} ||f_{\theta_r}(\mathbf{x}_{skin}) - \mathbf{w}||^2. \quad (6)$$

We set $\lambda_1 = 0.5, \lambda_2 = 0.05, \lambda_3 = 0.01, \lambda_4 = 0.1, \lambda_5 = 1, \lambda_6 = 0.1, \lambda_7 = 0.1$ in all experiments. For λ_4, we set it to 10 for the first $1k$ iterations for fast convergence to a reasonable skinning field, then decreased to 0.1 for soft regularization.

D. More Visualization Results

We also provide extended visualization results to further illustrate the robustness and versatility of our model for 3D avatars generation, as shown in Fig. 1. Specifically, these include examples of avatars rendered from novel views and with novel poses. These visualizations showcase the model's capacity to generalize effectively to new perspectives and body configurations, highlighting its potential in generating realistic and adaptable avatars across varied viewing conditions and postures. We provide generated videos in the project page[5].

We also present additional results in comparison with GauHuman [1], in Fig.2, demonstrating that our method achieves significantly better rendering quality.

References

[1] Shoukang Hu et al. GauHuman: Articulated gaussian splatting from monocular human videos. In *cvpr*, pages 20418–20431, 2024. 1, 3, 4

[2] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6M: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Trans. on Pattern Anal. and Mach. Intell.*, 36(7):1325–1339, 2013. 1

[3] Tianjian Jiang, Xu Chen, Jie Song, and Otmar Hilliges. InstantAvatar: Learning avatars from monocular video in 60 seconds. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 16922–16932, 2023. 1

[4] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. 1

[5] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 9054–9063, 2021. 1

[6] Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3DGS-Avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 5020–5030, 2024. 1

[7] Jing Wen, Xiaoming Zhao, Zhongzheng Ren, Alexander G Schwing, and Shenlong Wang. GomavAtar: Efficient animatable human modeling from monocular video using gaussians-on-mesh. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2059–2069, 2024. 1

[8] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. HumanNeRF: Free-viewpoint rendering of moving people from monocular video. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 16210–16220, 2022. 1, 3

[9] Zhengming Yu, Wei Cheng, xian Liu, Wayne Wu, and Kwan-Yee Lin. MonoHuman: Animatable human neural field from monocular video. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, pages 16943–16953, 2023. 1

[5] https://sggs-projectpage.github.io/