Real-time High-fidelity Gaussian Human Avatars with Position-based Interpolation of Spatially Distributed MLPs

Supplementary Material

1. More Implementation Details

Template Mesh. For the canonical template mesh used in our method, we use SMPL-X [2] mesh as the template mesh for avatars wearing tight clothing, and follow Animatable-Gaussians [1] to obtain template mesh for avatars wearing loose clothing.

Spatially Distributed MLPs. The spatially distributed MLP contains four hidden layers with 512, 256, 256, 256 neurons respectively. Each spatially distributed MLP takes only pose vector as input and outputs coefficients of length $2B = 30$. Half of the coefficients are interpolated by Gaussians to obtain Gaussian coefficients, while the other half are interpolated by control points to obtain control point coefficients.

For the implementation details, we use the new vmap function in Pytorch 2.0 for the parallel processing of multiple spatially distributed MLPs. The vmap function allows running models (e.g., MLPs and CNNs) with the same architecture in parallel. Specifically, we convert the MLP to a function (line 6 in Code 1), and then vmap runs the function with the batched weights and tensors as input (line 8 in Code 1). Please refer to the official document pytorch.org/tutorials/intermediate/ensembling.html for more details.

We also compare the forward time of vmap with group 1D convolution, which was used in SLRF [3] to run multiple MLPs in parallel. We run 1000 iterations and report the average time. The vmap implementation takes 0.52ms, and is much faster than the group convolution (3.70ms).

Training Details. At the beginning of training, we only optimize Gaussian neutral properties and neural position offsets. After 2K iterations, we optimize property offset basis and position offset basis, as well as spatially distributed MLPs. We also optimize only the zero-order component of the SH coefficients at the beginning. The first-order SH coefficients are optimized after 250K iterations.

```
1 from torch.func import vmap, functional_call,
      stack_module_state
2 models = [MLP(in=63,out=30) for _ in range(300)]
3 weights, _ = stack_module_state(models)
4 base_model = MLP(in=63,out=30).to("meta")
5 def fmodel(param, data):
6   return functional_call(base_model, param, data)
7 input_tensor = torch.tile(pose_vector, [300, 1])
8 output_tensor = vmap(fmodel)(weights, input_tensor)
9 # output tensor shape: (300, 30)
```

Code 1. Sample code for processing multiple MLPs in parallel.

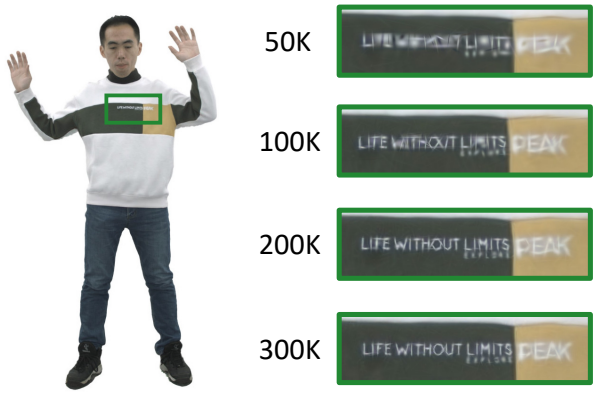

Figure 1. Qualitative comparison of different number of Guassians.

GS number	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	Training ↓	FPS ↑
50K	32.6020	0.9858	0.0244	10.7696	15.0 h	332
100K	32.6833	0.9863	0.0232	10.3303	15.8 h	246
200K	32.7456	0.9868	0.0226	10.1169	17.5 h	166
300K	32.7626	0.9869	0.0224	10.0621	19.0 h	129

Table 1. Quantitative comparison of different number of Gaussians.

2. Experiment Details

In the main paper, the quantitative results in Tab. 1, Tab. 4, Tab. 5, and Tab. 6 are evaluated on avatarrex_zzr sequence from AvatarRex [4] dataset. We use the first 2000 frames for training and calculate metrics on the first 500 frames from "22010710" camera view. The quantitative results in Tab. 2 of the main paper are evaluated on subject00 sequence from THuman4.0 [3] dataset. We use the first 2000 frames for training and calculate metrics on the remaining 500 frames from "cam18" camera view.

3. Ablation Study on Gaussian Number

We conduct experiments on the number of Gaussians used in our method. Quantitative experiments are shown in Tab. 1. Although reducing the number of Gaussians can greatly improve rendering speed, we find that fewer Gaussians make it more difficult to accurately capture details, as shown in Fig. 1. Therefore, we empirically choose 200K Gaussians in our method.

4. Ablation Study on PCA Components

We use 20 PCA components during testing. Fig. 2 shows the results of using different numbers of components and no PCA. The left results show that using fewer PCA components yields fewer details. The right results show that artifacts can appear when no PCA is used.

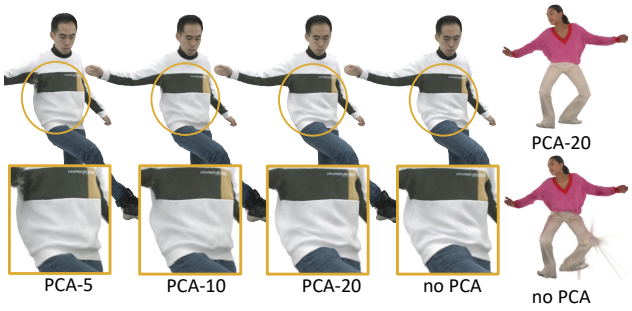

Figure 2. Ablation study on PCA components.

References

[1] Zhe Li, Zerong Zheng, Lizhen Wang, and Yebin Liu. Animatable gaussians: Learning pose-dependent gaussian maps for high-fidelity human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19711–19722, 2024. 1

[2] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed AA Osman, Dimitrios Tzionas, and Michael J Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10975–10985, 2019. 1

[3] Zerong Zheng, Han Huang, Tao Yu, Hongwen Zhang, Yandong Guo, and Yebin Liu. Structured local radiance fields for human avatar modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15893–15903, 2022. 1

[4] Zerong Zheng, Xiaochen Zhao, Hongwen Zhang, Boning Liu, and Yebin Liu. Avatarrex: Real-time expressive full-body avatars. *ACM Transactions on Graphics (TOG)*, 42(4):1–19, 2023. 1